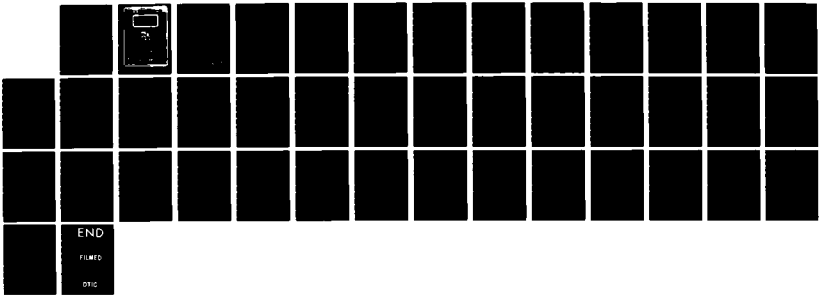
AD-A160 313 PERFORMANCE AND RELIABILITY ANALYSIS USING DIRECTED ACYCLIC GRAPHS(U) DUKE UNIV DURHAM NC DEPT OF COMPUTER SCIENCE R A SAHNER ET AL. 04 APR 85 CS-1985-9 1/1

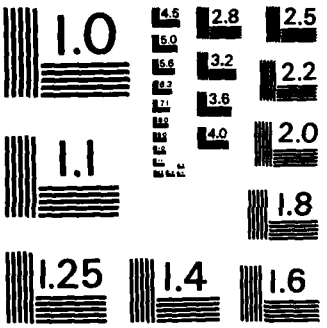
UNCLASSIFIED AFOSR-TR-85-0808 AFOSR-84-0132 F/G 9/2 NL



END

FILMED

DTIC

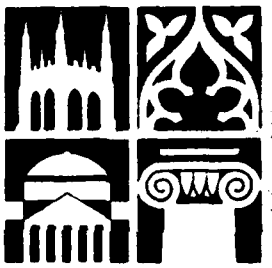


CS-1985-9

Performance and Reliability Analysis Using Directed Acyclic Graphs

Robin A. Sahner and Kishor S. Trivedi

Department of Computer Science
Duke University
Durham, N. C. 27706



DEPARTMENT OF COMPUTER SCIENCE

DUKE UNIVERSITY

(2)

CS-1985-9

Performance and Reliability Analysis Using Directed Acyclic Graphs

Robin A. Sahner and Kishor S. Trivedi

Department of Computer Science
Duke University
Durham, N. C. 27706

Performance and Reliability Analysis Using Directed Acyclic Graphs

Robin A. Sahner and Kishor S. Trivedi

Department of Computer Science
Duke University
Durham, N. C. 27706

This work was supported in part by the Air Force Office of Scientific Research under grant AFOSR-84-0132, by the Army Research Office under contract DAAG29-84-0045 and by the National Science Foundation under grant MCS-830200.

Abstract

A powerful model for the stochastic analysis of directed acyclic graphs is developed. These graphs represent event-precedence networks where events may occur serially, probabilistically, or concurrently. When a set of events occurs concurrently, the condition for the set of events to complete is that any specified number of the events must complete. This includes the special cases that one or all of the events complete. The distribution function associated with an event is assumed to have exponential polynomial form. Further generality is obtained by allowing these distributions to have a mass at the origin and/or at infinity. The distribution function for the time taken to complete the entire graph is computed in a semi-symbolic form. Applications of the model for the evaluation of concurrent program execution time and to the reliability analysis of fault-tolerant systems are discussed.

Additional keywords: fault tree nodes; SPADE computer program; SPADE (Series Parallel Directed Acyclic Graph Evaluator).

1. INTRODUCTION

Many interesting problems can be modeled by directed graphs whose nodes represent events and whose edges represent a precedence relation between the events. In this paper we consider the analysis of event-precedence graphs that are series-parallel. Each node in the graph is assigned a completion time distribution that has exponential polynomial form. This form is quite general, and is closely related to Neuts' phase-type distributions[20]. We allow distributions that have a mass at the origin and/or a mass at infinity.

If a graph is composed of serial subgraphs, then all events in the first subgraph must be completed before the first events in the second subgraph may begin. The division of a graph into parallel subgraphs can be interpreted as either probabilistic or concurrent. In the probabilistic case, the events of exactly one of the subgraphs will occur in any given traversal of the overall graph. In the concurrent case, the traversal of all of the subgraphs begins in parallel, but they need not all complete. The overall graph is considered to be completed when some specified number k of the n subgraphs have completed.

Given a graph and probability distributions $F(t)$ for the event completion time of individual nodes, we compute the distribution function of the completion time of the entire graph in terms of the time parameter t. We also can compute the completion time distribution for any particular path through the graph. A program called SPADE (*Series-PArallel Directed* acyclic graph *Evaluator*) has been written to compute these distributions.

The power of our model and solution method comes from several of its features.

1) The distributions of individual nodes are drawn from a large class of distributions, including

defective distributions . (A distribution function $F(t)$ is defective if $\lim_{t\to\infty} < 1$).

2) The distribution function for the completion of a graph is computed symbolically in t .

3) The interpretation of parallel subgraphs is chosen from a general and useful group of alternatives. Parallel subgraphs are not required to have identical distributions.

4) We make no assumptions about the nature of the events except that they are statistically independent.

Because of the above features, our model can be applied to many different kinds of problems, including performance analysis of concurrent programs and reliability/availability analysis of fault-tolerant systems.

A great deal of study has been done of event-precedence graphs which represent concurrent programs or transactions. Generally, it is assumed that all events must be completed; in some cases probabilistic branching is allowed. Extensive study of the analysis and scheduling of such graphs in the case that the event times are constant has been done[3,22]. When the event times are allowed to be random[7,9], the analysis of a graph becomes much more difficult. Two approaches which have been used are Markov chain techniques and path analysis.

The first approach converts the graph into a continuous-time Markov chain [14,16,26,27]. This approach restricts the node times to be exponentially distributed, and also quickly leads to an explosion in the state-space of the Markov chain. The path analysis technique involves computing the distribution of the time to traverse each path through the graph. For complex graphs the number of paths can be rather large, making the technique computationally expensive. In the general case, overlapping paths exist and hence one can only obtain an approximation (or bounds) for the overall execution time distribution[9].

If the shape of the graph is restricted to series-parallel, the overall execution time distribution can be obtained exactly by combining the distribution function of the individual nodes using multiplication and convolution. This is the approach taken by Robinson [24] and Kleinoder [13] in using directed graphs for the performance analysis of concurrent programs. Their graph model is a subset of ours, and is solved numerically starting from empirical distributions for the nodes. We allow a more general interpretation of parallel paths, and therefore can model more general programs, including those containing probabilistic branching and the implementation of non-deterministic algorithms. Furthermore, we allow the node exe-

cution times to be defective random variables. This further widens the class of programs we can analyze. For example, we can use our model to analyze the performance of programs in the presence of hardware or software failures[15,18].

In addition to performance analysis, the SPADE model can be applied to reliability and availability analysis. Approaches to reliability and availability analysis include the use of Markov and semi-Markov models [27,28], fault trees [2], and reliability block diagrams [8,19]. Generally, the analysis of these models has been done numerically. Fault trees and reliability block diagrams are easily transformed into our graph model, and then are solved symbolically. In general, Markov and semi-Markov chains are not series-parallel, but any acyclic chain can be transformed into an equivalent series-parallel graph and solved by SPADE.

In the next section, we describe the SPADE model. In section 3 we briefly describe the program SPADE. Section 4 gives examples illustrating the use of our approach. The examples chosen are simple, for the purpose of exposition. More complex problems can be and have been solved by the model.

2. THE SPADE MODEL

This section presents a definition of "series-parallel" graphs, describes how such graphs can be interpreted as representing events under precedence constraints and presents an algorithm for computing the distribution function for the time needed to complete the specified events.

2.1. Series-Parallel Graphs

Acyclic directed graphs are useful for modeling events that are bound by precedence constraints. Many problems that are NP-complete for arbitrary acyclic digraphs become tractable when restricted to the class of series-parallel graphs. Examples are the job sequencing problems discussed in [17] and [1].

There are many definitions for the term "series-parallel", and many different names for the series-parallel structure. In [13], such graphs are called "simple"; in [16] they are called "standard". We begin by defining a finite linear graph to be an ordered quadruple $G=(N,A,S,T)$ where

a) N is a finite set of elements called nodes

b) A is a subset of N x N, called the set of edges

c) S is the subset of N containing those nodes that are not the second member of any edge in A (these are the entrance or start nodes).

d) T is the subset of N containing those nodes that are not the first member of any edge in A (these are the exit or terminal nodes).

Suppose $G_1=(N_1, A_1, S_1, T_1)$ and $G_2=(N_2, A_2, S_2, T_2)$ are nonintersecting graphs. A graph $G=(N, A, S, T)$ is the series connection of G_1 and G_2 if and only if

a) At least one of $|T_1|$ and $|S_2|$ is 1. That is, at least one of the two sets is a singleton.

b) $N=N_1 \cup N_2$

c) $A=A_1 \cup A_2 \cup (T_1 \times S_2)$.

d) $S=S_1$, $T=T_2$

A graph G is the parallel combination of G_1 and G_2 iff

a) $N=N_1 \cup N_2$

b) $A=A_1 \cup A_2$

c) $S=S_1 \cup S_2$, $T=T_1 \cup T_2$

The class of series-parallel graphs is the smallest class of graphs containing the unit graphs (graphs consisting of one node) and having the property that whenever G is the series or parallel connection of two graphs in the class, then G is in the class. Note that a series-parallel graph is by definition acyclic and contains no redundant edges.

This definition is more restrictive than the definition of "minimal series parallel" given in [29], and less restrictive than the definition given in [1]. As pointed out in [1], any graph that is minimal series

parallel can be rewritten (with the addition of suitable intermediate nodes) as a graph that is series-parallel according to our definition.

Figure 1 shows several examples of series-parallel graphs. In this and all subsequent figures, the direction of the edges is not shown explicitly; it is assumed that an edge points downward. Figure 2 shows two graphs that are *not* series-parallel. Graph G1 in figure 2 is not series-parallel under any definition. Graph G2 is "transitive series-parallel" [29], but is not series-parallel because of the redundant arc from A to C.

Although the definition of series-parallel is binary in nature, it is convenient to think of each series or parallel combination as being built from n subgraphs, rather than two. The parallel (series) combination of n subgraphs is defined to be the sequence of $n-1$ binary parallel (series) combinations of the subgraphs.

2.2. Graph Traversal Time

The nodes in a graph represent events that take time; with each node is associated a distribution function. We define the traversal time distribution of a graph recursively.

If G consists of a single node, the traversal time distribution is given by the distribution function associated with the node.

Suppose G was formed by combining the graphs $G_1, G_2, \cdots, G_n$ having independent distribution functions $F_1, F_2, \cdots, F_n$. If the graphs were combined in series, then the graphs are traversed one at a time, and the distribution function for the graph G is given by

$$1)\quad F_{ser}(t) = \bigotimes_{i=1}^{n} F_i(t)$$

where the symbol $\otimes$ represents convolution. The convolution of two CDFs F_j and F_k is defined by

$$F_{j \otimes k}(t) = \int_0^t F_k(t-x)\, dF_j(x)$$

Note that the order of traversal of the subgraphs does not matter.

If the graphs G_i were combined in parallel, we allow the parallelism to be interpreted as either probabilistic or concurrent. For probabilistic parallel subgraphs, only one of the subgraphs is actually

traversed. Each subgraph has associated with it the probability that it is chosen for traversal. Suppose the probability that subgraph G_i is chosen is p_i. Then the traversal time distribution for G is given by

$$2)\ F_{prob}(t) = \sum_{i=1}^{n} p_i F_i(t)$$

For concurrent parallel subgraphs, the traversal time for G is given by the k^{th} smallest traversal time of the n subgraphs. Two special cases occur so often that we treat them separately. If k is one, then the traversal time is the minimum of the traversal times of the subgraphs. In that case we have

$$3)\ F_{\min}(t) = 1 - \prod_{i=1}^{n}(1 - F_i(t))$$

If k is n, then the traversal time is the maximum of those of the subgraphs, and we have

$$4)\ F_{\max}(t) = \prod_{i=1}^{n} F_i(t)$$

In the general case, if the n subgraphs have identical distributions F, then the traversal time distribution for G is (omitting the parameter t for better readability)

$$5)\ F_{k/n} = \sum_{i=k}^{n} \binom{n}{i} F^i(1 - F)^{n-i}$$

If the subgraphs do not have identical distributions, the expression for $F_{k/n}$ is (see [5])

$$6)\ F_{k/n} = \sum_{|T| \geq k} \left(\prod_{j \in T} F_j\right)\left(\prod_{j \notin T}(1-F_j)\right)$$

where T is a set of indices ranging over all combinations of indices chosen from $\{1, 2, \cdots, n\}$. That is, T ranges over all choices $\{j_1, j_2, \cdots, j_m\}$ such that $m \leq n$ and $j_1 < j_2 < \cdots < j_m$.

Equation 6 can be written in a form more suitable for mechanical computation. Given a vector $\vec{F} = (F_1, F_2, \cdots, F_n)$, let $S_i(\vec{F})$ be the the elementary symmetric polynomial of degree i in $\vec{F}$. That is,

$$S_i(\vec{F}) = \sum_{|U| = i} \prod_{j \in U} F_j$$

where U ranges over all combinations of i indices chosen from $\{1, 2, \cdots, n\}$. Then we have the following lemma.

Lemma: If n subgraphs have distributions given by the elements of the vector $\vec{F}$, the traversal time distribution for k out of the n subgraphs is given by

6′) $F_{k/n}(\vec{F}) = \sum_{i=k}^{n} (-1)^{i-k} \binom{i-1}{k-1} S_i(\vec{F})$

Proof :

First, we expand the second product in equation 6, so that we have

$$F_{k/n}(\vec{F}) = \sum_{|T| \geq k} \left(\prod_{j \in T} F_j\right)\left(\sum_{V \subseteq T^c} (-1)^{|V|} \prod_{j \in V} F_j\right)$$

$$= \sum_{\substack{|T| \geq k, \\ V \subseteq T^c}} \sum_{j \in T \cup V} \left((-1)^{|V|} \prod F_j\right)$$

Now we are interested in obtaining the coefficient for a particular product $F_{j_1} \bullet F_{j_2} \bullet \cdots \bullet F_{j_i}$ where $i \geq k$. That product appears whenever $T \subseteq \{j_1, \cdots, j_i\}$ and $|T| \geq k$. For each $m \geq k$, there are $\binom{i}{m}$ ways of choosing T. Once T is chosen, V is determined and $|V| = i - m$. Therefore the coefficient of interest is (see [23])

$$\sum_{m=k}^{i} (-1)^{i-m} \binom{i}{m} = \sum_{m=0}^{i-k} (-1)^m \binom{i}{m} = \sum_{m=0}^{i-k} (-1)^m \left[\binom{i-1}{m} + \binom{i-1}{m-1}\right] = (-1)^{i-k} \binom{i-1}{k-1}$$

Noting that this coefficient is independent of the particular i-tuple chosen, and that the sum of all such i-tuples is exactly the elementary symmetric polynomial $S_i(\vec{F})$, we have the result desired.

Formula 6′ is much more computationally tractable than formula 6, because the elementary symmetric polynomials can be computed efficiently. Let us write $S_i(j)$ for the elementary symmetric polynomial of degree i chosen out of a vector $\vec{F}$ with j components. We can compute the polynomials $S_i(j)$ as follows.

a) $S_1(1) = F_1$

b) $S_1(j) = S_1(j-1) + F_j$ for $j > 1$

c) $S_j(j) = S_{j-1}(j-1) \bullet F_j$ for $j > 1$

d) $S_i(j) = S_{i-1}(j) + (F_j \bullet S_i(j-1))$ for $1 < i < j$

This method is shown graphically for $j = 4$ in figure 3. Note that we do not need to compute all of the terms $S_i(j)$, but only the last $n - k + 1$ terms in each row. Using this method, the number of multiplications of CDFs is $O(n(n-k))$.

3. Graph Interpretation

Suppose we are modeling program execution. Clearly, series subgraphs represent serial statement ecution. The interpretation of parallel subgraphs depends on the type assigned to the parallelism. The pe *maximum* corresponds to the kind of concurrency considered in [3] and by Robinson[24] and leinoder[13]. If parallel subgraphs are probabilistic, they can be interpreted as the alternatives in a contional statement. Minimum parallel subgraphs will model the parallel execution of a non-deterministic gorithm [25] in which the verification of all guessed solutions is attempted concurrently, and the first uess to be verified provides a solution to the whole problem. We can also consider the unreliability of asks by representing each task by a parallel combination of the task execution and the failure process of he task. We can thus model software reliability as proposed by Littlewood[18].

SPADE graphs can also be used to model the lifetime of closed (non-repairable) fault-tolerant systems with permanent faults. Such systems are defined in [21], where they are analyzed by Markov chain echniques. We should note that our graphs do allow more general distributions of subsystem or component lifetimes than those allowed by the Markov chain techniques. A system consisting of a series combination of components is modeled by parallel graph nodes with type minimum; a parallel combination is nodeled by parallel graph nodes with type maximum. Systems with redundant components which require ome minimum number of the components to function can be modeled by k out of n parallel subgraphs.

We can also model the point (instantaneous) availability for the restricted class of repairable systems where each component has an exponentially distributed lifetime and an independent repair facility.

3.4. Graph Analysis

The algorithm for computing the traversal time distribution for a series-parallel graph has two parts. First we decompose the graph into a tree, such that the nodes of the graph appear as leaves of the tree, and the sequence of series and parallel combinations that form the graph appear as internal tree nodes. Figure 4 shows a graph and its tree decomposition. When the decomposition is parallel, we label the nternal node with the particular interpretation (maximum, minimum, probabilistic, or k out of n) placed on the traversal of the parallel subgraphs. It is possible to carry out this decomposition in time proportional to the number of nodes in the resulting tree. For a description of such a linear algorithm for

mposing any transitive series-parallel graph into a binary tree, see [29].

Every subgraph of the series-parallel graph corresponds to some subtree of the decomposition tree. lefine the distribution of a tree node T to be the traversal time distribution of the subgraph to which subtree rooted at T corresponds. The distribution of the root node of the decomposition tree is the ersal time distribution of the entire graph. Thus, the second part of the algorithm consists of comput- the distribution for each node of the decomposition tree. We visit all of the nodes of the tree in pos- ler. If a node is a leaf, then its distribution is exactly that of the graph node corresponding to the leaf. node is internal, then we apply one of the formulas 1 through 6′ , depending on the type of the node, he immediate descendents of the node.

If we take as our unit of calculation the multiplication and convolution of distributions, then the ıe needed to calculate the CDF for a tree node of type series, maximum, minimum, probabilistic, or k t of n with identically distributed subgraphs is $O(n)$. If the type is k out of n with non-identically tributed subgraphs, the time is at worst (when k is small) $O(n^2)$.

The overall time complexity of the algorithm depends on the representation of distributions, and on e implementation of the various operations done on distributions.

5. Distribution Functions

Up to this point, we have made no assumptions about the character of the CDFs associated with the ›des of our graphs except that they are statistically independent. For CDFs of any form, it would be ›ssible to compute numerically $F(t)$ for traversal of the entire graph for any particular value of t. If ıe type of the CDF's is restricted to be of exponential polynomial form and the parameters are given, it relatively easy to compute the overall CDF as a function of t.

An exponential polynomial is defined to be an expression of the form

$$\sum_i a_i t^{k_i} e^{b_i t}$$

ote that this form is quite general. In particular, the CDF of each node can be exponential, hyperex- onential, Erlang, or a mixture of Erlang distributions. Because the class of exponential polynomials is losed under the operations of addition, subtraction. multiplication, differentiation and integration, a

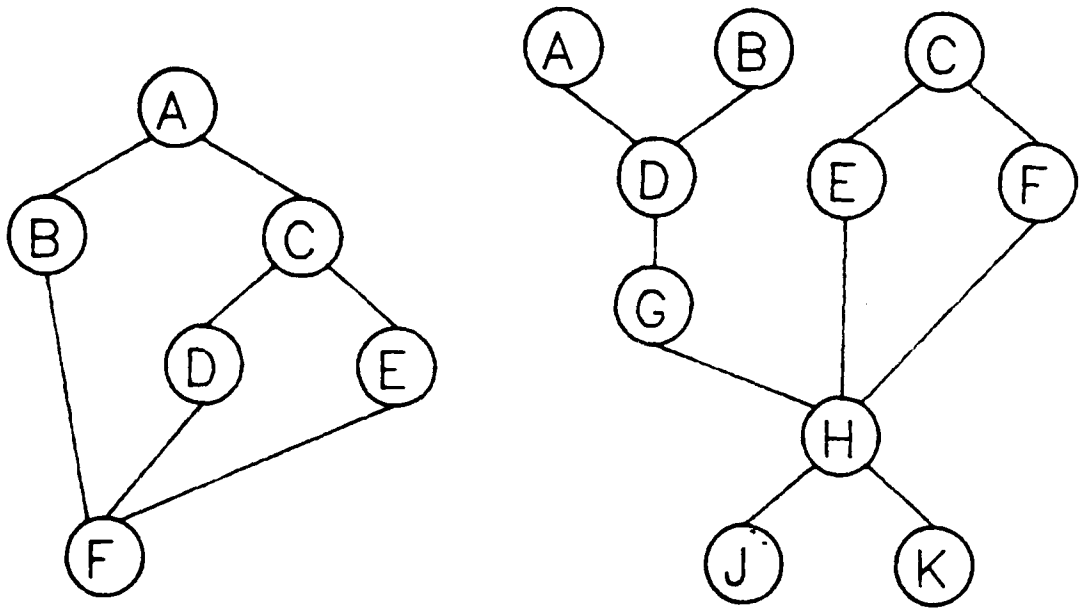


Figure 1. Examples of Series–Parallel Graphs

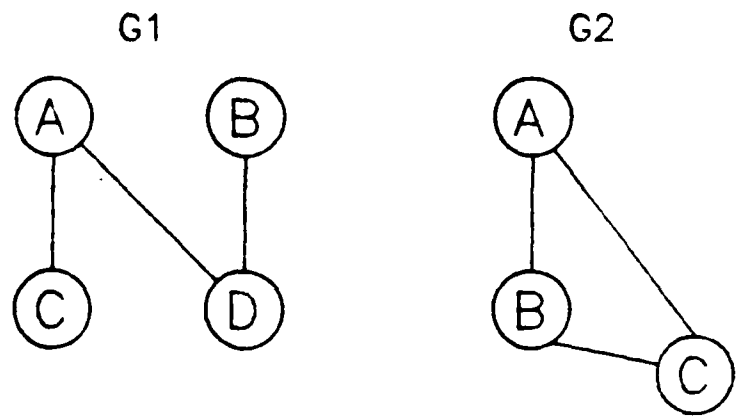


Figure 2. Graphs which are not Series–Parallel

Wei, A.T., and Campbell, R.H., "Construction of a Fault Tolerant Real-Time Software System", Tech Report UIUCDCS-R-80-1042, U. of Illinois, Dec. 1980.

[2] Heidelberger, P. and Trivedi, K.S., "Queueing Network Models for Parallel Processing with Asynchronous Tasks", *IEEE Trans. on Computers*, Vol. C-31 No. 11 (Nov. 1982), 1099-1109.

[3] Kleinoder, W., "Evaluation of Task Structures for a Hierarchical Multiprocessor System", *Proc. Int. Conf. on Modeling Techniques and Tools for Performance Analysis*, Paris, France, May 1984.

[4] Kulkarni, V. and Adlakha, W., "Markov and Markov-Regenerative PERT Networks", Tech. Report, Operations Research and Systems Analysis, Univ. of North Carolina at Chapel Hill, 1984.

[5] Kulkarni, V., Nicola , V. and Trivedi, K., "On Modeling the Performance and Reliability of Multi-Mode Computer Systems," *Proc. Int. Workshop on Modeling and Performance Evaluation of Parallel Systems*, Grenoble , Dec. 1984, (to be published by North-Holland).

[6] Kung, K.C.-Y., "Concurrency in Parallel Processing Systems", Ph.D. Dissertation, UCLA Computer Science Department, 1984.

[7] Lawler, E. L., "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints", *Annals of Discrete Mathematics* 2 (1978) 75-90.

[8] Littlewood, B., "A Semi-Markov Model for Software Reliability with Failure Costs", *Proceedings of the Symposium on Computer Software Engineering*, New York (1976), 281-300.

[9] Modarres, M., "A Method of Predicting Availability Characterstics of Series-Parallel Systems", *IEEE Transactions on Reliavility*, Vol. R-33 No. 4, (Oct 1984), 309-312.

[10] Neuts, M.F., *Matrix-Geometric Solutions in Stochastic Models*, The Johns Hopkins University Press, Baltimore, Md., 1981.

[11] Ng, Y.-W. and Avizienis, A., "A Model for Transient and Permanent Fault Recovery in Closed Fault-Tolerant Systems," *Proc. 1976 Int. Symp. on Fault-Tolerant Computing*, June 1976.

[12] Ramamoorthy, C.V., and Ho, G.S., "Performance Analysis of Asynchronous Concurrent Systems Using Petri Nets", *IEEE Transactions on Software Engineering*, Vol. SE-6 No. 5 (Sept 1980), 440-449.

[13] Riordan, John, *Combinatorial Identities*, John Wiley & Sons, NY., 1968.

[14] Robinson, J.T., "Some Analysis Techniques for Asynchronous Multiprocessor Algorithms", *IEEE Transactions on Software Engineering*, Vol. SE-5 No. 1 (Jan 1979), 24-31.

[15] Sedgewick, R., *Algorithms*, Addison-Wesley, Reading, Mass., 1983.

[16] Towsley, D.F., Browne, J.C. and Chandy, K.M., "Models for Parallel Processing within Programs," *CACM*. October 1978.

[17] Trivedi, K.S., *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[18] Trivedi, K., Dugan, J. B., Geist R., and Smotherman ,M., "Modeling Imperfect Coverage in Faul-Tolerant Systems", *Proc. of the Fourteenth Int. Conf. on Fault-Tolerant Computing (FTCS - 14)*, Orlando, FL., June 1984, 77-82.

[19] Valdes, J., Tarjan, R.E., and Lawler, E.L., "The Recognition of Series-Parallel Digraphs", *Siam J. of Computing*, Vol. 11 no 2 (1982), 298-313.

Parallel subgraphs are allowed to have non-identical (but independent) distributions.

This generality allows us to model a wide-ranging set of applications including the execution time analysis of concurrent programs, program execution in a failure-prone environment, reliability analysis of non-repairable fault-tolerant systems, and availability analysis of a class of repairable systems.

Several generalizations of the techniques discussed in this paper are under investigation. These include the restriction of a limited number of processors and hence the modeling of queuing for limited resources, perhaps in a way analogous to the methods employed in [10,11]. We are also developing SPADE-like methods specialized for solving acyclic Markov and semi-Markov chains.

REFERENCES

[1] Abdel-Wahab, H.M. and Kameda, T., "Scheduling to Minimize Maximum Cumulative Cost Subject to Series-Parallel Precedence Constraints", *Operations Research*, Vol 26 No. 1, 141-158.

[2] Bavuso, S., Peterson, P., and Rose, D., "Care III Model Overview and User's Guide", NASA Technical Memorandum Number 85810, June 1984.

[3] Coffman, E.G., *Computer and Job/Shop Scheduling*, John Wiley & Sons, NY., 1976.

[4] Coffman, E. G., and Denning, P. J., *Operating Systems Theory*", Prentice-Hall, Inc., New Jersey, 1973.

[5] David, H.E., *Order Statistics*, John Wiley & Sons, NY., 1981.

[6] Dugan, J.B., Trivedi, K.S., Geist, R.M. and Nicola, V.F., "Extended Stochastic Petri Nets: Analysis and Applications", *PERFORMANCE '84*, Paris, Dec 1984 (proceedings to be published by North-Holland).

[7] Fix, W. and Neumann, K., "Project Scheduling by Special GERT Networks," *Computing* 23, 3 (1979), 299-308.

[8] Fleming, J. L., "Relcomp: A Computer Program for Calculating System Reliability and MTBF", *IEEE Transactions on Reliability*, Vol. R-20 No. 3 (Aug 1971), 102-107.

[9] Gaul, W., "On Stochastic Analysis of Project Networks," in M.A.H. Dempter et al. (eds.), *Deterministic and Stochastic Scheduling*, D. Reidel Publishing Co., 1982.

[10] Haase, V. R., "Real-Time Behavior of Programs", *IEEE-TSE*, Vol SE-7 No. 5 (Sept 1981), 494-501.

[11] Heidelberger, P. and Trivedi, K.S., "Analytic Queueing Models for Programs with Internal Concurrency", *IEEE Trans. on Computers*, Vol. C-32, No. 1 (Jan. 1983), 73-82.

system is able to recover from the fault. The second way of leaving state B is for an error (owing to the fault) to occur. The error rate is ρ. Once an error occurs, the system must detect the error; the error detection rate is ϵ. A detected error is covered with probability c and is not covered with probability 1-c.

This Markov chain can be written as a graph suitable for input to SPADE, as shown in figure 12b. Node A represents the time it takes for a fault to occur. Node B represents the time that passes before either the fault is detected or the fault causes an error. With probability $\delta/(\delta+\rho)$, the fault is detected. In that case we go to node C1, which represents a state in which the system is able to recover. Otherwise, the fault causes an error. That leads to state D, during which the system tries to recover from the error. Once the recovery attempt is finished, we exit either through node C2 if the error was covered, or node F if the error was not covered. Nodes C1, C2 and F are *zero* nodes. They are required in order to express the different ways of exiting from the graph.

Figure 12c shows the results obtained when SPADE is asked to analyze each path through the graph. For each path, SPADE prints

(1) the names of one or more nodes that uniquely identify the path

(2) the probability of taking the path

(3) the conditional distribution for the path

SPADE also prints the unconditional CDF for the graph, computed as if the graph had a dummy exit node collecting nodes C1, C2 and F.

The probability that the system recovers from a fault is given by the sum of the probabilities of traversing the paths that go through BC and DC. The probability that the system does not recover is the probability of traversing the path through DF.

5. CONCLUSION AND FUTURE WORK

We have developed a model for the execution time of stochastic activity networks of series-parallel type. We allow node execution times to have quite general exponential polynomial forms and allow these distributions to have a mass at origin and a mass at infinity. We further allow several interpretations of parallel subgraphs, including the possibilities of required completion of one, all or k of n subgraphs.

approximation given in [19] is in error. The approximation as computed using the method in [19] is $7.1021 \bullet 10^{-4}$.

The bottom part of figure 10c shows the results obtained when all of the μ_i are set to zero, thus these results are the mean, variance, and values of the CDF for the time to failure of the system when components are not repaired.

4.5. Example 5 - Reliability of an Aircraft Flight Control System

To illustrate the use of *k out of n parallelism*, we consider example problem 7 in appendix G of [2]. This problem models an aircraft flight control system. The system contains three inertial reference sensors (IRS) and three pitch rate sensors (PRS), that monitor the status of the aircraft. All of the sensors are connected to each of four computer systems (CS). The computer systems independently collect information from the sensors and process the information. The computers are connected to each other and to three secondary actuators (SA) through four identical bus systems (BS).

In order for the entire system to function (so that the aircraft remains airborne) at least two of each type of component must be functioning. A graph that expresses a reliability model of the overall system is given in figure 11a.

Note that the subgraph representing the computer systems and secondary actuators are *3 out of 4* systems. That is because the nodes represent *times to failure*. If the subsystems operate as long as 2 out of the 4 components function, then they fail when 3 out of the 4 have failed.

Figure 11b shows an input file for the graph and figure 11c shows the results. Note the use of the shorthand method for specifying *k* out of *n* identical single-node subgraphs.

4.6. Example 6 - A Fault Handling Model

This example is also taken from [2]. Consider the Markov chain in figure 12a. This is a model of the sequence of events that follows the occurrence of a fault in a system that monitors itself periodically. When the system is in state A, it is functioning properly, and the fault rate is λ. In state B, a fault has occurred. At this point, one of two things might happen. The first is that the system may detect the fault itself. This happens with rate δ. If the fault is detected, the system goes to state C, in which the

system of components pictured in figure 10a. This is the example presented in [19], where an approximation method is given for computing the steady state unavailability of a series-parallel system. Using our model, we can compute the steady state unavailability exactly, and in addition we compute the transient unavailabilities.

Assume that each component is subject to failure, and has its own independent repair facility. If the time to failure of component i is exponentially distributed with failure rate λ_i and the time to repair is exponentially distributed with repair rate μ_i, then the instantaneous availability is [27]

$$A_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i} e^{-(\lambda_i + \mu_i)t}$$

Note that as $t \to \infty$, $A_i(t)$ approaches the steady-state availability. If $\mu_i = 0$ (no repair), $A_i(t)$ reduces to the reliability (as a function of time) of the component.

Let the distribution function associated with the graph node representing component i be $1 - A_i(t)$. This distribution represents the unavailability of the component, and is in SPADE form with a mass at infinity. We can use SPADE to compute the instantaneous unavailability for the system as a whole. For subsystems in parallel, we must take the product of the component unavailabilities (the system is unavailable only when all parallel subsystems are unavailable). This is the "maximum" combination. For a series of components, the availability is the product of the component availabilities (the system is available only when all subsystems are available). Thus the unavailability of the system is exactly the "minimum" combination of the components.

When the components are combined in this way, the "traversal time" of the overall graph will be the overall system unavailability $1 - A(t)$. By taking the limit of $A(t)$ we obtain the steady-state system availability, and by setting all $\mu_i = 0$ we obtain system reliability as a function of the mission time t.

Figure 10b shows how the series-parallel system can be expressed in SPADE-form. We first analyze the model with the same parameters as used in [19]. Those parameters are $\lambda_5 = \lambda_6 = .005$, $\lambda_1 = .001$, $\lambda_i = .01$ for all other i, $\mu_5 = \mu_6 = 1/6$, $\mu_1 = \mu_{10} = \mu_{11} = 1/5$, and $\mu_i = 1/7.5$ for all other i. The results (with distributions functions deleted) are shown in the top part of figure 10c. The continuous probability is $\lim_{t \to \infty} 1\text{-}A(t)$, and hence is the exact steady state unavailability. It should be noted that the

of three sequential phases[4]. The first phase, corresponding to seek time, is assumed to be exponentially distributed with a mass at the origin:

$$F_{seek}(t) = p_{noseek} + (1 - p_{noseek})(1 - e^{-\lambda_{seek} t})$$

The second phase is the rotational latency phase, and is assumed to exponentially distributed. The third phase, the transfer phase, is also assumed to be exponentially distributed.

Figure 8d shows an input file and results for this modifed model. We have used several convenient features of SPADE to specify the model. We have defined an exponential polynomial called *hyper*, with two parameters, to define the hyperexponential distribution. When we later assign distributions to the nodes *CPU* 1 and *CPU* 2, we simply invoke that polynomial definition with appropriate arguments. We have also defined a subgraph called *io*, consisting of three nodes in series, to represent the three I/O phases. Later, when we assign distributions to the nodes *IO* 1 and *IO* 2, we say that they have the same distribution as the subgraph *io*. In addition to being convenient, the use of the subgraph facility makes the program execution time shorter, since each subgraph will be evaluated only once.

4.3. Example 3 - Program Execution with a Possibility of Failure

To see how SPADE can be used to analyze the finishing time of a program which is subject to software or hardware failure, we consider an example taken from Wei and Campbell[30]. In figure 9a, the nodes in the graph represent segments of a process. Associated with each segment is the probability that a failure occurs before execution of the segment is complete. In [30], a formula is given for approximating the overall failure probability. SPADE computes the failure probability exactly, and in addition computes the CDF of the process completion time if a failure does not occur.

Figure 9b shows the results for the process graph. Figure 9c compares the exact results from SPADE with the approximations obtained by the method used in [30]. As expected, the approximation is better when the individual failure probabilities are smaller.

4.4. Example 4 - Instantaneous Availability

If a system is composed of components that each have an independent repair facility, the SPADE model can be used to compute the instantaneous availability of the system. Consider the series-parallel

The time taken for a message to be consumed is assumed to be exponentially distributed with parameter λ. The distribution of the time taken for a message to be produced is given by $F(t) = p - p * e^{-\mu t}$.

If p is less than one, the distribution for the amount of time it takes to produce a message does not reach one in the limit. We can interpret this as meaning that there is a chance that the message is never produced, and that $(1-p)$ is the probability that the message is never produced. The distribution $F(t)$ is translated by SPADE into the mixture distribution $(1-p)\,I + p\,(1-e^{-\mu t})$.

Figures 7b and 7c show an input file and the results obtained from SPADE. Because the overall CDF does not reach one in the limit, SPADE shows the probability that the overall graph traversal takes no time, the probability that it takes infinite time, and the probability that the graph is traversed in finite nonzero time. In this example, the "infinite" probability is the probability that the number of messages actually produced is less than two. The CDF given is conditional on the traversal time being finite.

4.2. Example 2 - CPU-I/O Overlap

Figure 8a shows a SPADE graph for one iteration of the program with CPU-I/O overlap considered by Towsley, Chandy and Browne [26]. In each iteration of the program there are two stages. The first stage is always a CPU burst. The second stage consists of either pure input/output, or input/output that may be overlapped with a second CPU burst. The probability that the second stage consists of CPU-I/O overlap is given by p.

The use of a "zero" node allows us to have one branch of the CPU1 node lead to a single node, while the other branch leads to a group of nodes to be executed in parallel. Figures 8b and 8c show a SPADE input file and the results obtained from that file.

We can also used SPADE to carry out the analysis of an iteration of this program assuming that no concurrency is allowed. The resulting mean execution time is .27505. The speedup, defined to be the ratio of the mean sequential execution time to the mean parallel execution time, is 1.21.

To show the versatility of SPADE, we now allow the CPU service time distribution to be a two-stage hyperexponential (with the same mean as before). The I/O service time will be assumed to consist

any combination of variables, constants, and the operators +, -, *, /, exponentiation and parentheses. In addition, SPADE contains a mechanism for specifying two types of functions of any number of parameters. The first type of function is defined to be an arithmetic expression. Once such a function is defined, it can be used in other expressions wherever a variable would appear. The second type is defined to be an exponential polynomial, and provides a convenient way for a user to pre-define commonly used distribution functions. The overall CDF of a graph is computed in the single time parameter t; the user must bind any variable names used to particular values before the calculation is done.

Once a user has supplied a graph, exit types, probabilities, distributions, and values for variable names, SPADE computes and prints the distribution for the traversal time of the graph and the mean and variance of the traversal time. The user may request that the distributions be evaluated over specified intervals of values.

When a graph has probabilistic parallel subgraphs, the user may be interested in knowing the probability that a particular node or subgraph was chosen, and the conditional distribution of the graph traversal time given that particular choice. SPADE allows the user to request that traversal times be computed for each possible path through the graph.

4. EXAMPLES

This section contains examples of models that can be analyzed using SPADE.

4.1. Example 1 - Producer-Consumer Problem

Consider the producer-consumer problem, where the number of messages produced (and consumed) is two. The process of the production and consumption of the two messages is shown by the graph in Figure 7a. The nodes P1 and P2 represent the production of the first and second messages; C1 and C2 represent the consumption of the messages. Obviously, each message cannot be consumed until it is produced, but it is possible for message one to be consumed while message two is being produced. Haase [10] explores this problem when the production and consumption times are deterministic, with the objective of discovering whether both messages are consumed by a given deadline. We assume the times to be probabilistic, and use SPADE to compute the distribution of the time taken to consume both messages.

in the form of a simple input language.

A SPADE user must first specify a series-parallel event-precedence graph. This is done by entering the node pairs which define the edges in the graph. SPADE allows the degenerate case where an edge is specified by a single node. This allows for the specification of a node which is not part of any edge. Because it is often the case that a graph contains many subgraphs which are identical copies of each other both in shape and distribution, SPADE has a macro facility for the specification of subgraphs.

Because of our definition of series-parallel, every set of parallel subgraphs is immediately preceded by some single graph node. (If a graph or subgraph as specified by the user has multiple entrance nodes, SPADE provides a dummy single entrance node.) Therefore, it is convenient for the user to specify the type of parallelism of subgraphs by assigning an "exit type" to that single node. The possible exit types are probabilistic, maximum, minimum, and k out of n. In the case of k out of n parallelism, the user must specify values for k and n.

SPADE provides a shorthand form of specifying k out of n parallel subgraphs when the subgraphs have identical distributions. It allows the user to say that the exit type of a node with only one outgoing edge is k out of n. When that happens, SPADE assumes that there are n identical copies of the immediate descent of the node. To use this shorthand feature when the identical subgraphs consist of more than a single node, one would use the macro subgraph facility.

For each edge leaving a node with probabilistic exit type, the user must specify the probability that the edge is traversed. It is possible for more than one edge to enter a single probabilistic subgraph. In that case, the interpretation is that the probability of entering that subgraph is the sum of the probabilities on all of the edges entering the subgraph. Later when the subgraph is itself decomposed, the multiple edges will be split into more probabilistic subgraphs.

Every node must be assigned a probability distribution. SPADE provides a shorthand for the user to specify the exponential distribution and the distributions Z and I. If any other distribution is desired, the user must specify each term of the desired exponential polynomial.

Values for probabilities, k and n for k out of n exits, and the parameters a_i, k_i and b_i for exponential polynomials may all be specified in the form of symbolic expressions. The expressions may contain

Thus the values for p_z , p_∞, p_c and F^c in $F_{\max}$ are

$$p_z = p_{z_1} p_{z_2}$$

$$p_\infty = p_{\infty_1} + p_{\infty_2}(p_{z_1}+p_{c_1})$$

$$p_c = p_{z_1} p_{c_2} + p_{c_1} p_{z_2} + p_{c_1} p_{c_2}$$

$$F^c = \frac{p_{z_1} p_{c_2} F_2 + p_{c_1} p_{z_2} F_1 + p_{c_1} p_{c_2} F_1 F_2}{p_{z_1} p_{c_2} + p_{c_1} p_{z_2} + p_{c_1} p_{c_2}}$$

The formulas for probabilistic, serial and minimum combinations are similar. If $G_1, G_2, \cdots G_n$ are k out of n parallel subgraphs having identical distributions $p_z Z + p_\infty I + p_c F$ we have the following.

$$5' \;) \; F_{k/n} = \sum_{m=0}^{n-k} \sum_{i=0}^{k-1} \binom{n}{m} \binom{n-m}{i} p_\infty^m \; p_z^i \; p_c^{n-m-i} \sum_{j=k-i}^{n-m-i} \left\{ \frac{n-m-i}{j} \right\} F^j \; (1-F)^{(n-m-i)-j}$$

Note that if p_z and p_∞ are zero, this reduces to equation 5. The product $p_\infty^m \; p_z^i \; p_c^{n-m-i}$ is the probability that the traversal time is infinite for m of the subgraphs and zero for i of the subgraphs. The rest of the formula represents the time it takes to traverse $k - i$ of the remaining pool of $n - m - i$ subgraphs that have finite nonzero traversal time.

If the subgraphs do not have identical distributions, we have

$$6' \;' \;) \; F_{k/n}(\vec{F}) = \sum_{\substack{|T_I| \le n-k \\ T_Z \subseteq T_I' \\ |T_Z| \le k-1}} \prod_{m \in T_I} p_{z_m} \prod_{i \in T_Z} p_{\infty_i} \prod_{j \in (T_I \cup T_Z)^c} p_{c_j} \; F_{(k-|T_I|)/(n-|T_I|-|T_Z|)}(\vec{F_j'})$$

Here T_I is any permutation chosen from $\{1, \cdots, n\}$ and T_Z is a permutation chosen from the remaining indices. The products of probabilities represent the probability that the traversal time is infinite for $|T_I|$ of the subgraphs and zero for $|T_Z|$ of the subgraphs. Then equation 6' is used to calculate the time it takes to traverse the remaining pool of subgraphs that have finite nonzero traversal time.

3. THE SPADE PROGRAM

SPADE is a program which implements the analysis of series-parallel event-precedence graphs. It is written in C, and consists of about 2800 lines of code. SPADE may be used either interactively or in batch mode. The data which must be supplied by the user is the same in either case. When used interactively, SPADE prompts the user for data entry, allowing retry whenever possible if invalid data is entered and ensuring that all required data is entered. In batch mode, the user creates a file which contains data

precedence relations that otherwise would not adhere to our definition of series-parallel. For example, the non series-parallel graph G2 in figure 2 can be written equivalently as the series-parallel graph in figure 6.

The limit of a distribution at infinity represents the probability that an event ever finishes. If an event represents part of a numerical algorithm that may not always converge, it is useful to be able to express the probability that the algorithm does not converge. Similarly, if we consider program execution in a failure-prone environment, then we may allow for the possibility of a failure occurring before program completion, so that the program never completes.

The SPADE model allows each node in a graph to have a mixture distribution in the form of equation 11. Note that a node for which $F(0) > 0$ and $\lim_{t \to \infty} F(t) < 1$ can be represented equivalently by three probabilistic parallel nodes, one having distribution Z, one having distribution I, and one having an absolutely continuous CDF satisfying properties 7 through 10. This is illustrated in figure 5.

Equations 1 through 6′ for computing the distributions of combinations of subgraphs apply when the component CDFs are absolutely continuous non-defective distributions. Mixture distributions in the form of equation 11 are also closed under the operations sum, prob, max, min, and k/n. It is convenient to rederive formulas 1 through 6′ for mixture distributions. First we make the following observations.

12) $Z^2 = Z$

13) $I^2 = I$

14) $I\ Z = I$

15) $Z\ F^c = F^c$

16) $I\ F^c = I$

Now suppose that a graph G is composed of two maximum parallel subgraphs G_1 and G_2 having distributions $p_{z_1}Z + p_{\infty_1}I + p_{c_1}F_1$ and $p_{z_2}Z + p_{\infty_2}I + p_{c_2}F_2$. The distribution for the traversal time of G is given by

$$4') \quad F_{\max} = p_{z_1}p_{z_2}Z^2 + p_{z_1}p_{\infty_2}ZI + p_{z_1}p_{c_2}ZF_2 + p_{\infty_1}p_{z_2}IZ + p_{\infty_1}p_{\infty_2}I^2 +$$

$$p_{\infty_1}p_{c_2}IF_2 + p_{c_1}p_{z_2}F_1Z + p_{c_1}p_{\infty_2}F_1I + p_{c_1}p_{c_2}F_1F_2$$

$$= (p_{z_1}p_{z_2})Z + (p_{\infty_1} + p_{\infty_2}(p_{z_1}+p_{c_1}))I + (p_{z_1}p_{c_2}F_2 + p_{c_1}p_{z_2}F_1 + p_{c_1}p_{c_2}F_1F_2)$$

series-parallel graph whose nodes have exponential polynomials for CDF's will have an overall CDF that is also an exponential polynomial.

Of course, not every exponential polynomial is a valid CDF. For a function $F(t)$ to be the CDF of a nonnegative random variable, it must satisfy the following properties:

7) $F(0) = 0$

8) $\lim_{t \to \infty} F(t) = 1$

9) $0 \leq F(t) \leq 1 \ \forall t$

10) F is monotone nondecreasing

It is useful to relax requirements 7 and 8, and require only

7') $F(0) \geq 0$

8') $\lim_{t \to \infty} F(t) \leq 1$

If $F(0) > 0$, then $F(0)$ is a discrete probability mass at the origin. If $\lim_{t \to \infty} F(t) < 1$, then F is a defective distribution. F can be written as a mixture distribution [27] composed of the sum of two discrete parts and a continuous part. Define Z to be the CDF of the discrete distribution with all of its mass at the point 0. Thus $Z(t) = 1$ for $t \geq 0$. Define I to be the CDF of the discrete distribution with all of its mass at infinity. Thus $I(t) = 0$ for all finite $t \geq 0$, and $I(\infty) = 1$. Every distribution F that is exponential polynomial in form and satisfies properties 7' , 8' , 9 and 10 can be written as

11) $F_{mixed}(t) = p_z \bullet Z(t) + p_\infty \bullet I(t) + p_c \bullet F^c(t)$

where $p_z + p_\infty + p_c = 1$ and F^c is an exponential polynomial with $F^c(0) = 0$ and $\lim_{t \to \infty} F^c(t) = 1$. F_{mixed} is obtained from F by setting $p_z = F(0)$, $p_\infty = 1 - \lim_{t \to \infty} F(t)$ and $F^c = \frac{F - p_z}{p_c}$.

$F(0)$ is the probability that the event with distribution F takes no time. If an event represents the failure of a component, it is useful to allow for the possibility that the component is defective to begin with. Also, the distribution of the waiting time in a queuing system usually possesses a mass at the origin.

It is possible to have $F = Z$, in which case the event always takes no time. This is the counterpart of an instantaneous transition in a stochastic Petri net [6]. These "zero" nodes can be used to specify

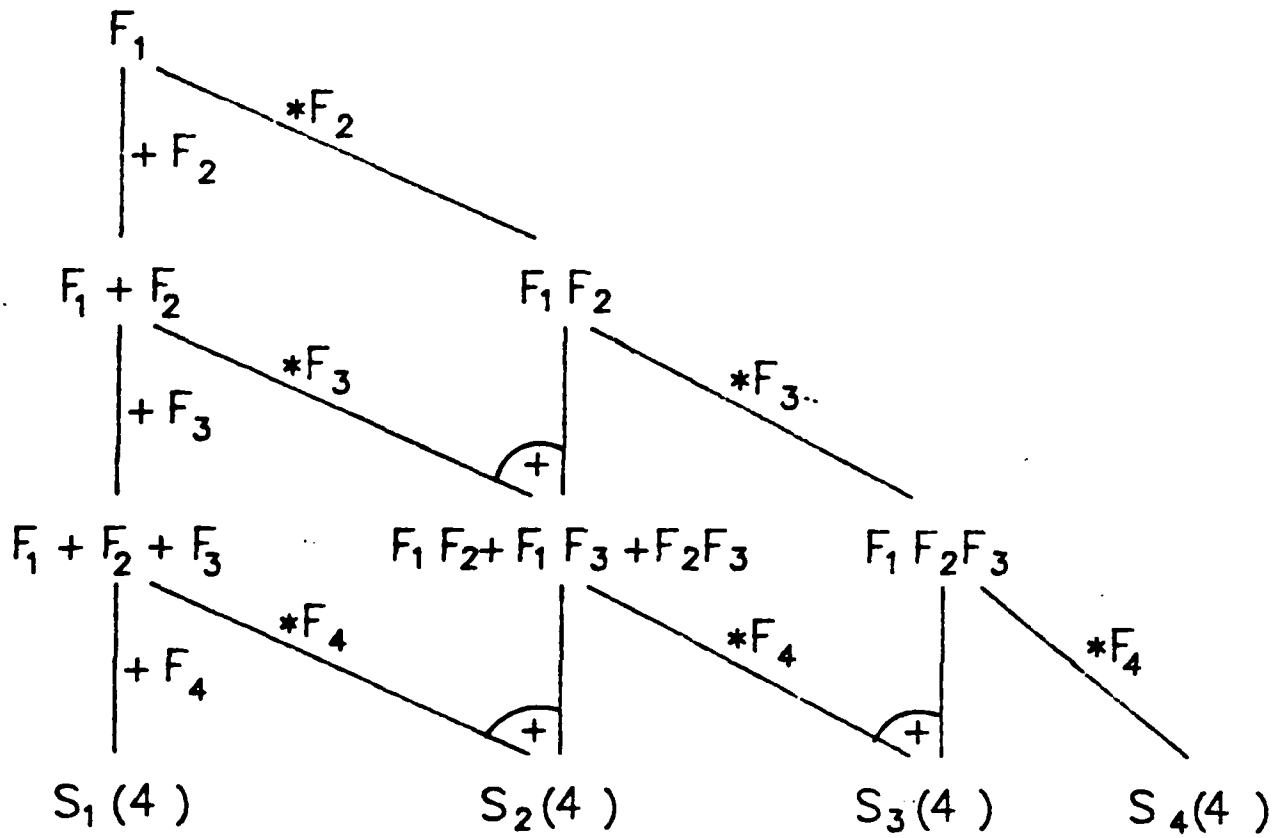


Figure 3. Computation of $S_i(4)$

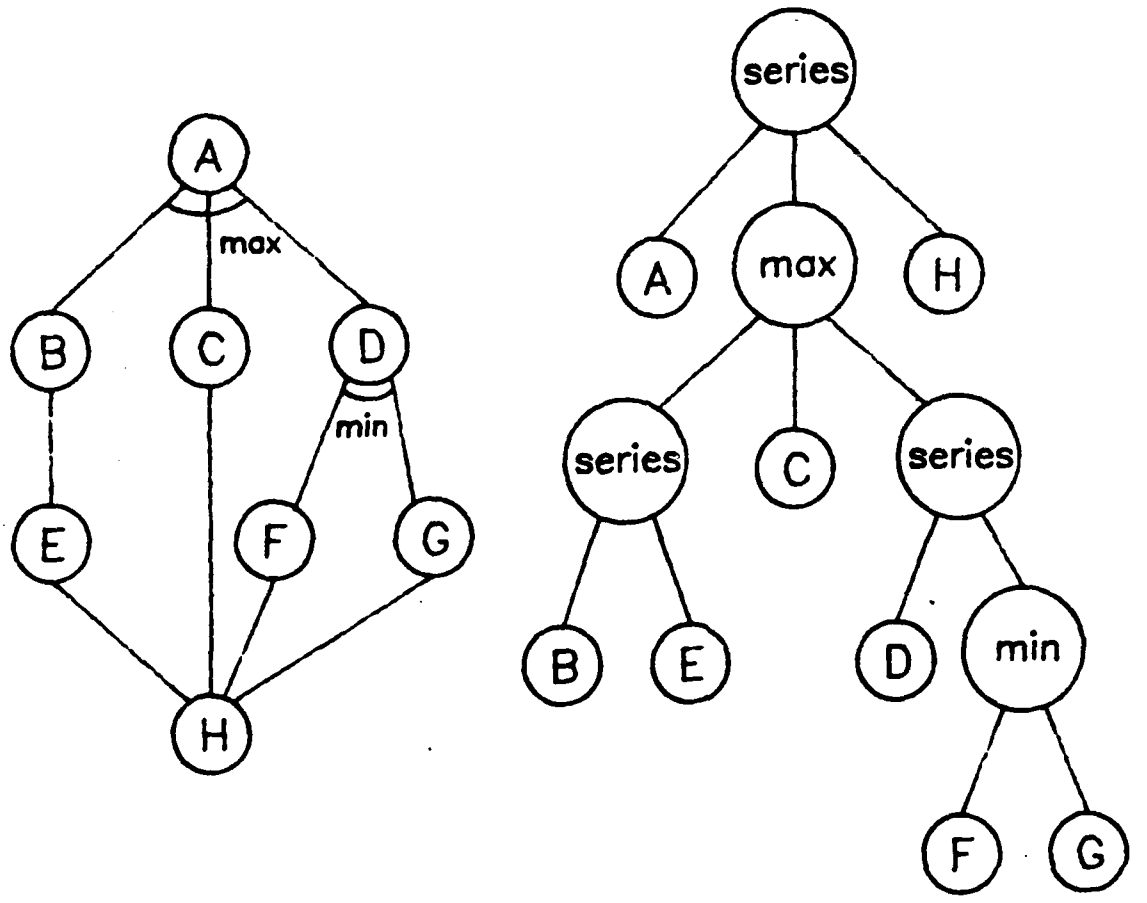


Figure 4. A Series–Parallel graph and its decomposition tree

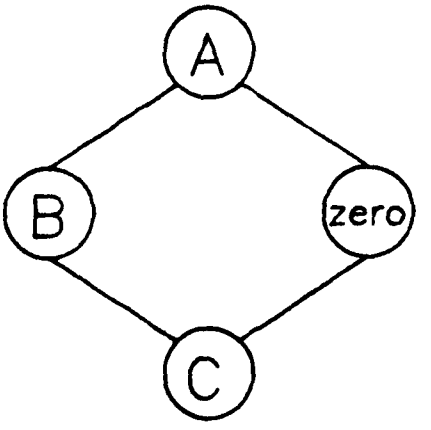


Figure 5. Series–Parallel Equivalent of G2

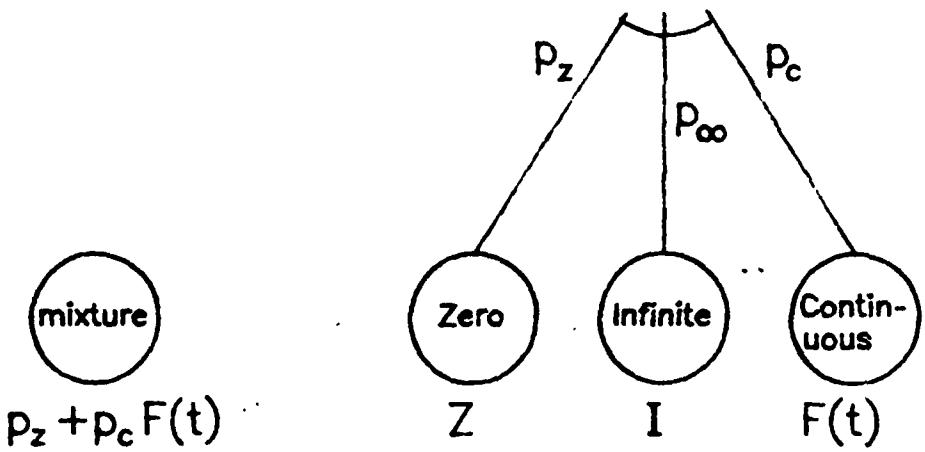


Figure 6. A node with mixture distribution and its three–node equivalent

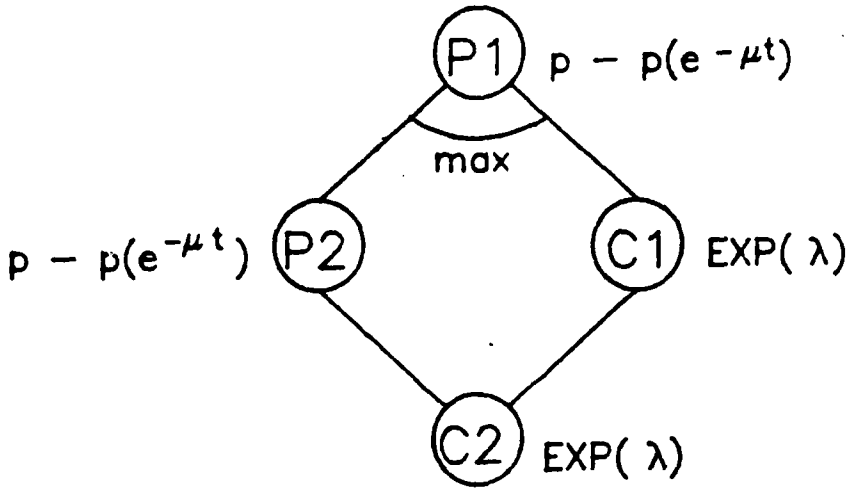


Figure 7a. Producer–Consumer Problem

```
COMM   producer-consumer problem
COMM   with 2 messages

GRAPH
ARC P1 P2
ARC P1 C1
ARC P2 C2
ARC C1 C2
END

EXIT  P1  MAX
DIST  C1  EXP  lambda
DIST  C2  EXP  lambda
DIST  P1  GEN  p, 0, 0\
           -p, 0, -mu
DIST  P2  GEN  p, 0, 0\
           -p, 0, -mu
END

BIND lambda   1 / .2
BIND mu       1 / .5
BIND p        .95
END

EVAL 1 10 2
END
```

Figure 7b. Input File for Example 1

```
probability at 0:                0.0000e+00
probability at infinity:         9.7500e -02
continuous probability:          9.0250e -01

CDF:
      -3.0083e+00 t(  1) exp(-2.0000e+00 t)
 +     3.0083e+00 t(  1) exp(-5.0000e+00 t)
 +     9.0250e -01 t(  0) exp( 0.0000e+00 t)
 +    -9.0250e -01 t(  0) exp(-2.0000e+00 t)
 +    -9.0250e -01 t(  0) exp(-5.0000e+00 t)
 +     9.0250e -01 t(  0) exp(-7.0000e+00 t)

mean and variance are conditional on finite time

mean:  1.2571e+00
variance:  5.1878e-01

     t        F(t)

 1.0000 e+00  3.8824 e-01
 3.0000 e+00  8.7789 e-01
 5.0000 e+00  9.0178 e-01
 7.0000 e+00  9.0248 e-01
 9.0000 e+00  9.0250 e-01
```

Figure 7c. Results for Example 1

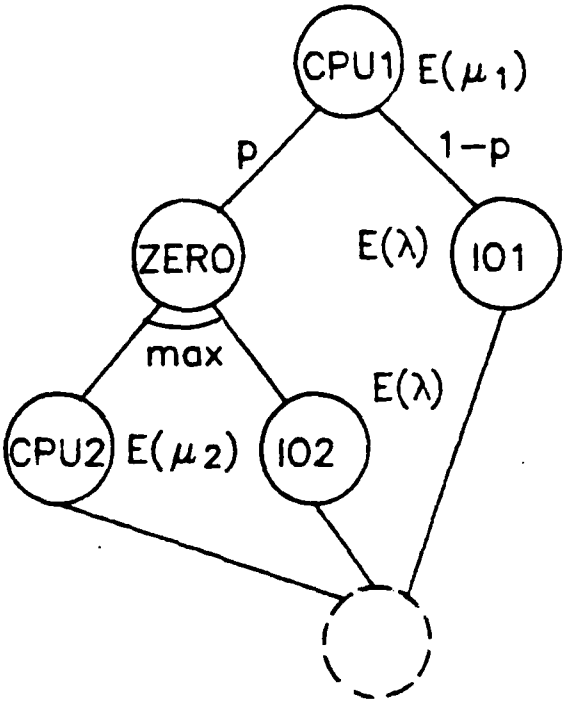


Figure 8a. CPU–IO Overlap

```
COMM     CPU-I/O overlap

GRAPH

ARC      cpu1      zero
ARC      cpu1      io1
ARC      zero      cpu2
ARC      zero      io2
END

EXIT     cpu1      PROB
PROB     cpu1      zero      p
EXIT     zero      MAX
DIST     cpu1      EXP       mu1
DIST     zero      ZERO
DIST     io1       EXP       lambda
DIST     cpu2      EXP       mu2
DIST     io2       EXP       lambda
END

BIND   mu1     1 / 0.0376
BIND   mu2     1 / 0.125
BIND   lambda  1 / 0.14995
BIND   p       .7
END

END
```

Figure 8b. Input File for CPU-I/O Overlap

```
CDF:
        1.0000e+00 t(  0) exp( 0.0000e+00 t)
 +   -1.3347e+00 t(  0) exp(-6.6689e+00 t)
 +   -1.0011e+00 t(  0) exp(-8.0000e+00 t)
 +    1.5609e+00 t(  0) exp(-1.4669e+01 t)
 +   -2.2512e -01 t(  0) exp(-2.6596e+01 t)

mean:  2.2733e-01
variance:  2.5755e-02

     t            F(t)

0.0000 e+00  0.0000 e+00
1.0000 e -01  2.0933 e -01
2.0000 e -01  5.2815 e -01
3.0000 e -01  7.4775 e -01
4.0000 e -01  8.7095 e -01
5.0000 e -01  9.3512 e -01
6.0000 e -01  9.6758 e -01
7.0000 e -01  9.8382 e -01
8.0000 e -01  9.9192 e -01
9.0000 e -01  9.9595 e -01
```

Figure 8c. Results for CPU-I/O Overlap

```
COMM  CPU-I/O overlap
COMM  with 2-stage CPU service
COMM  and 3-stage IO service

SUBGRAPH io
ARC   seek latency
ARC   latency transfer
END

GRAPH
ARC       cpu1       zero
ARC       cpu1       io1
ARC       zero       cpu2
ARC       zero       io2
END

POLY  hyper(x1,x2)\
          1, 0,   0\
    -x2/(x2-x1), 0, -x1\
     x1/(x2-x1), 0, -x2

DIST  seek     gen\
          1, 0, 0\
       -(1-pn), 0, -tseek

DIST    latency     exp      tlatency
DIST    transfer    exp      ttransfer

EXIT    cpu1        prob
PROB    cpu1        zero     p
EXIT    zero        max
DIST    cpu1        hyper (mu1a,mu1b)
DIST    zero        zero
DIST    io1         subgraph io
DIST    cpu2        hyper (mu2a,mu2b)
DIST    io2         subgraph io
END

BIND  mu1a          1 / .0156
BIND  mu1b          1 / .022
BIND  mu2a          1 / 0.0250
BIND  mu2b          1 / 0.1
BIND  p .7
BIND  pn            .001
BIND  tseek         1 / .05
BIND  tlatency      1 / .02
BIND  ttransfer     1 / .08
END

END
```

```
CDF:
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+   -1.4178e+00 t(  0) exp(-1.0000e+01 t)
+   -6.0884e+00 t(  0) exp(-1.2500e+01 t)
+    7.2026e+00 t(  0) exp(-2.0000e+01 t)
+    1.0119e+01 t(  0) exp(-2.2500e+01 t)
+   -1.4319e+01 t(  0) exp(-3.0000e+01 t)
+    5.1714e+00 t(  0) exp(-4.0000e+01 t)
+   -1.9595e+01 t(  0) exp(-4.5455e+01 t)
+    1.0076e+01 t(  0) exp(-5.0000e+01 t)
+    2.9553e+01 t(  0) exp(-5.2500e+01 t)
+   -4.1718e+01 t(  0) exp(-6.0000e+01 t)
+    2.0147e+01 t(  0) exp(-6.4103e+01 t)
+   -1.3064e -01 t(  0) exp(-9.0000e+01 t)

mean:  2.1611e-01
variance:  1.1814e-02

        t          F(t)

0.0000 e+00  0.0000 e+00
1.0000 e -01  1.0188 e -01
2.0000 e -01  5.1780 e -01
3.0000 e -01  8.1418 e -01
4.0000 e -01  9.3659 e -01
5.0000 e -01  9.7915 e -01
6.0000 e -01  9.9318 e -01
7.0000 e -01  9.9775 e -01
8.0000 e -01  9.9925 e -01
9.0000 e -01  9.9975 e -01
```

Figure 8d. Input File and Results for Modified CPU-I/O Overlap Model

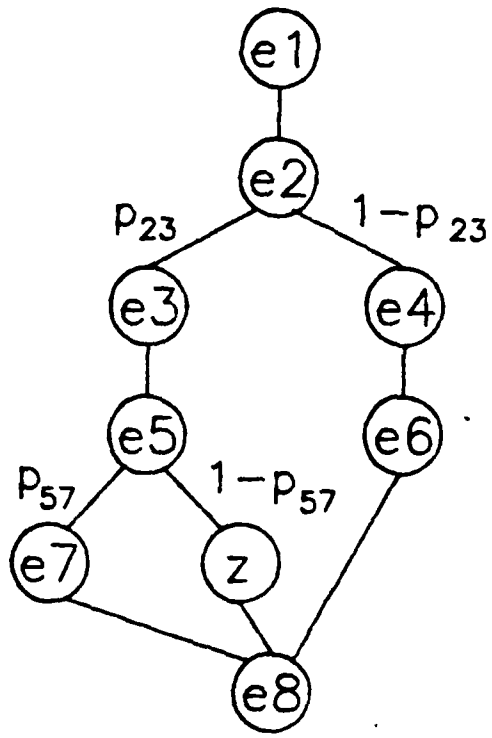


Figure 9a. Modules Which may Fail

probability at 0: 0.0000e+00
probability at infinity: 2.5590e-01
continuous probability: 7.4410e-01

CDF:

-6.9806e -01 t(5) exp(-3.0000e+00 t)
+ -2.5113e+00 t(4) exp(-3.0000e+00 t)
+ -3.3485e+00 t(3) exp(-3.0000e+00 t)
+ -3.3485e+00 t(2) exp(-3.0000e+00 t)
+ -2.2323e+00 t(1) exp(-3.0000e+00 t)
+ 7.4410e -01 t(0) exp(0.0000e+00 t)
+ -7.4410e -01 t(0) exp(-3.0000e+00 t)

mean and variance are conditional on finite time

mean: 1.8211e+00
variance: 6.3466e-01

----------REBIND----------

probability at 0: 0.0000e+00
probability at infinity: 2.8319e-02
continuous probability: 9.7168e-01

CDF:

-9.4129e -01 t(5) exp(-3.0000e+00 t)
+ -3.2794e+00 t(4) exp(-3.0000e+00 t)
+ -4.3726e+00 t(3) exp(-3.0000e+00 t)
+ -4.3726e+00 t(2) exp(-3.0000e+00 t)
+ -2.9150e+00 t(1) exp(-3.0000e+00 t)
+ 9.7168e- 01 t(0) exp(0.0000e+00 t)
+ -9.7168e- 01 t(0) exp(-3.0000e+00 t)

mean and variance are conditional on finite time

mean: 1.8261e+00
variance: 6.3643e-01

Figure 9b. Results for Example 3

	1st dataset	2nd data set
SPADE Result	.2559	.0283
Wei & Campbell approximation	.2864	.02864

Figure 9c. SPADE Results vs. Approximation

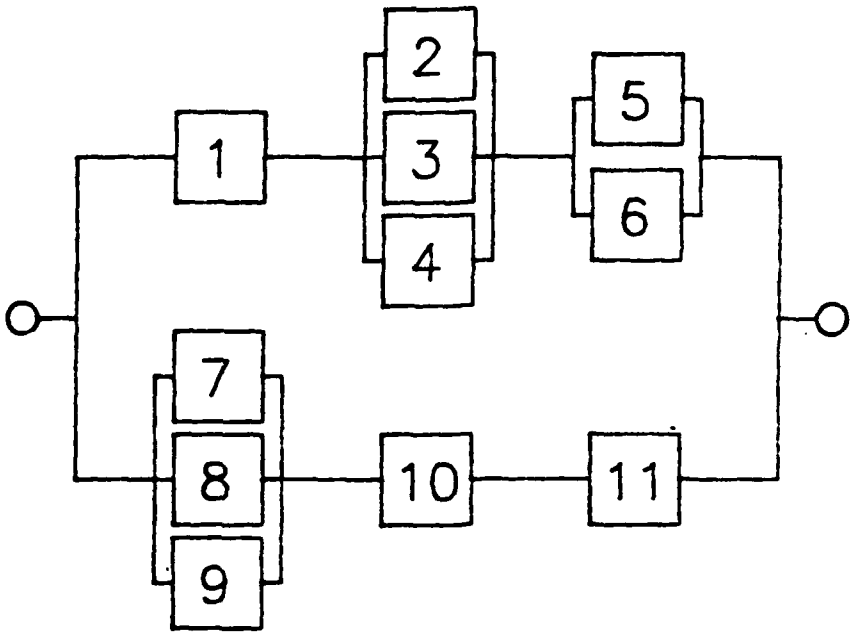
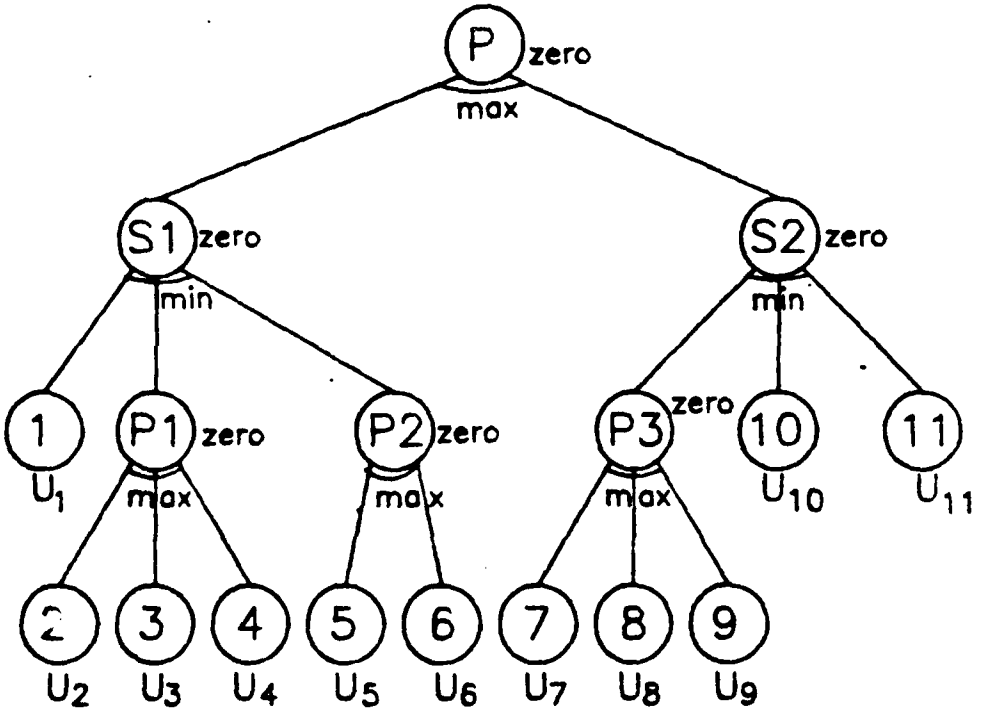


Figure 10a. Series–Parallel System



$$U_i = \frac{\lambda_i}{\lambda_i + \mu_i} - \frac{\lambda_i}{\lambda_i + \mu_i} e^{-(\lambda_i + \mu_i)t}$$

Figure 10b. SPADE Graph for a Series–Parallel System

probability at 0: 0.0000e+00
probability at infinity: 9.9943e-01
continuous probability: 5.7430e-04

t	F(t)
0.0000 e+00	0.0000 e+00
2.0000 e+00	5.5847 e -05
4.0000 e+00	1.5948 e -04
6.0000 e+00	2.6177 e -04
8.0000 e+00	3.4674 e -04
1.0000 e+01	4.1198 e -04
1.2000 e+01	4.5996 e -04
1.4000 e+01	4.9438 e -04
1.6000 e+01	5.1868 e -04
1.8000 e+01	5.3568 e -04
2.0000 e+01	5.4750 e-04

----------REBIND----------

t	F(t)
0.0000 e+00	0.0000 e+00
2.0000 e+00	8.2537 e -05
4.0000 e+00	3.4181 e -04
6.0000 e+00	7.9804 e -04
8.0000 e+00	1.4741 e -03
1.0000 e+01	2.3941 e -03
1.2000 e+01	3.5830 e -03
1.4000 e+01	5.0651 e -03
1.6000 e+01	6.8641 e -03
1.8000 e+01	9.0021 e -03
2.0000 e+01	1.1499 e -02

Figure 10c. Results for Example 4

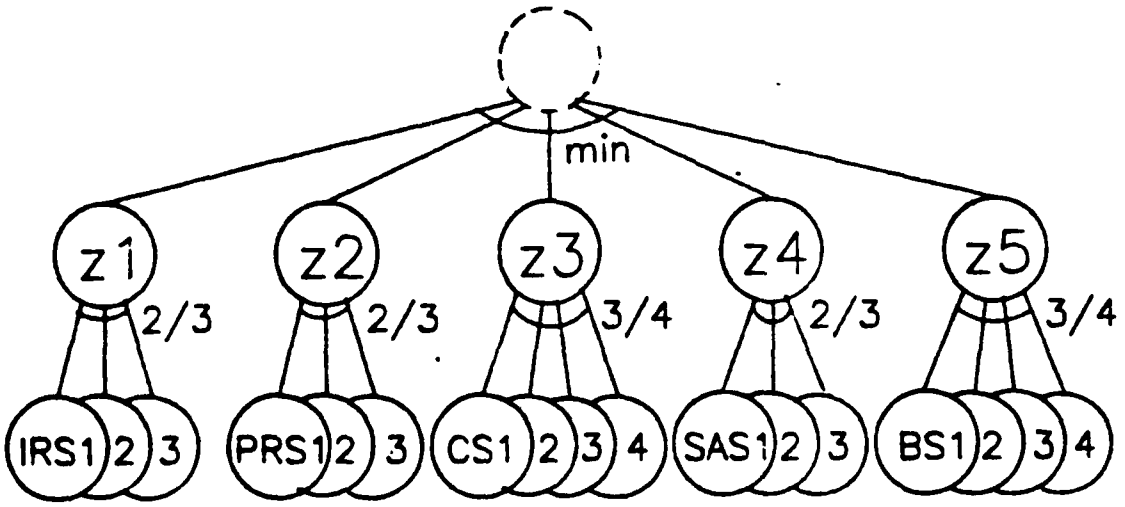


Figure 11a. Aircraft Control System

```
COMM     aircraft flight control system
COMM  (shorthand for k out of n subgraphs)

GRAPH
ARC      z1          IRS
ARC      z2          PRS
ARC      z3          CS
ARC      z4          SAS
ARC      z5          BS
END
EXIT     ENTRANCE    MIN

DIST     z1          ZERO
EXIT     z1          KOFN      2,3

DIST     z2          ZERO
EXIT     z2          KOFN      2,3

DIST     z3          ZERO
EXIT     z3          KOFN      3,4

DIST     z4          ZERO
EXIT     z4          KOFN      2,3

DIST     z5          ZERO
EXIT     z5          KOFN      3,4

DIST     IRS         EXP  0.0002
DIST     PRS         EXP  0.0003
DIST     CS          EXP  0.0004
DIST     SAS         EXP  0.0005
DIST     BS          EXP  0.0006
END

EVAL   10  100  10
END
```

Figure 11b. Input File for Example 5

```
CDF:
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+   -9.7200e+02 t(  0) exp(-4.0000e-03 t)
+    6.4800e+02 t(  0) exp(-4.2000e-03 t)
+    6.4800e+02 t(  0) exp(-4.3000e-03 t)
+    1.2960e+03 t(  0) exp(-4.4000e-03 t)
+    2.1600e+02 t(  0) exp(-4.5000e-03 t)
+    4.3200e+02 t(  0) exp(-4.6000e-03 t)
+   -1.2960e+03 t(  0) exp(-4.7000e-03 t)
+   -1.7820e+03 t(  0) exp(-4.8000e-03 t)
+   -1.1520e+03 t(  0) exp(-4.9000e-03 t)
+   -1.1160e+03 t(  0) exp(-5.0000e-03 t)
+    6.1200e+02 t(  0) exp(-5.1000e-03 t)
+    1.2420e+03 t(  0) exp(-5.2000e-03 t)
+    1.8360e+03 t(  0) exp(-5.3000e-03 t)
+    1.1640e+03 t(  0) exp(-5.4000e-03 t)
+    4.9200e+02 t(  0) exp(-5.5000e-03 t)
+   -3.8400e+02 t(  0) exp(-5.6000e-03 t)
+   -1.0920e+03 t(  0) exp(-5.7000e-03 t)
+   -1.0560e+03 t(  0) exp(-5.8000e-03 t)
+   -7.9200e+02 t(  0) exp(-5.9000e-03 t)
+    5.3000e+01 t(  0) exp(-6.0000e-03 t)
+    1.4400e+02 t(  0) exp(-6.1000e-03 t)
+    5.9400e+02 t(  0) exp(-6.2000e-03 t)
+    4.5000e+02 t(  0) exp(-6.3000e-03 t)
+    9.6000e+01 t(  0) exp(-6.4000e-03 t)
+    5.4000e+01 t(  0) exp(-6.5000e-03 t)
+   -1.9200e+02 t(  0) exp(-6.6000e-03 t)
+   -1.0800e+02 t(  0) exp(-6.7000e-03 t)
+   -1.0800e+02 t(  0) exp(-6.8000e-03 t)
+    7.2000e+01 t(  0) exp(-7.0000e-03 t)

mean:  8.6446e+02
variance:  2.4563e+05

      t            F(t)

 1.0000 e+01  1.1431 e-04
 2.0000 e+01  4.5834 e-04
 3.0000 e+01  1.0335 e-03
 4.0000 e+01  1.8410 e-03
 5.0000 e+01  2.8814 e-03
 6.0000 e+01  4.1553 e-03
 7.0000 e+01  5.6628 e-03
 8.0000 e+01  7.4037 e-03
 9.0000 e+01  9.3775 e-03
 1.0000 e+02  1.1583 e-02
```

Figure 11c. Results for Example 5

```
comm 1st data set
bind      f1        .02
bind      f2        .03
bind      f3        .04
bind      f4        .05
bind      f5        .06
bind      f6        .07
bind      f7        .08
bind      f8        .09
bind      u         3
bind      p23       .6
bind      p57       .8
end
end

comm 2nd data set
bind      f1        .002
bind      f2        .003
bind      f3        .004
bind      f4        .005
bind      f5        .006
bind      f6        .007
bind      f7        .008
bind      f8        .009
bind      u         3
bind      p23       .6
bind      p57       .8
end

end
```

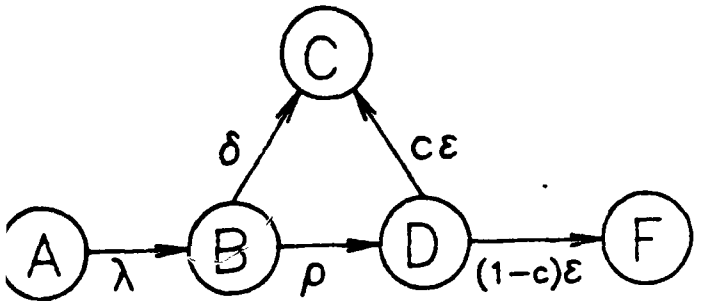


Figure 12a. Markov Chain for Single Fault Handling

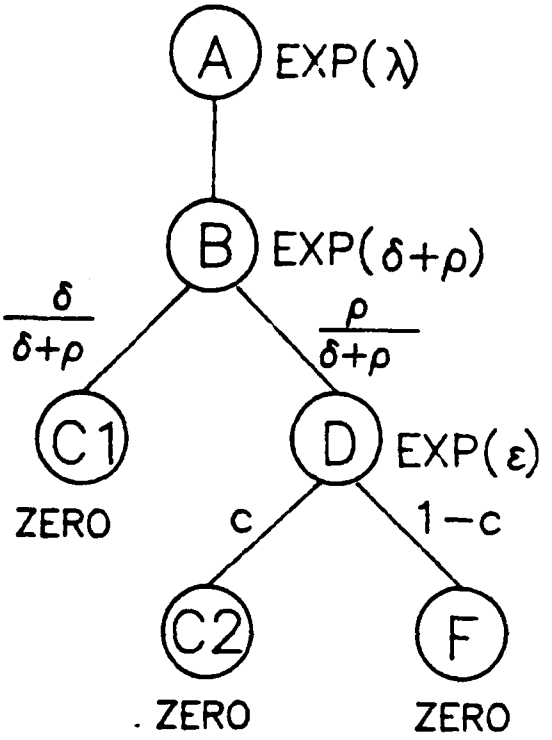


Figure 12b. SPADE Graph Equivalent

PATH: 1

nodes on the path: C1
probability of the path: 0.800

CDF for the path:

```
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+    -1.2500e+00 t(  0) exp(-1.0000e+00 t)
+     2.5000e -01 t(  0) exp(-5.0000e+00 t)
```

mean for the path: 1.2000e+00
variance for the path: 1.0400e+00

PATH: 2

nodes on the path: C2
probability of the path: 0.180

CDF for the path:

```
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+    -1.1338e+00 t(  0) exp(-1.0000e -01 t)
+     1.3889e -01 t(  0) exp(-1.0000e+00 t)
+    -5.1020e -03 t(  0) exp(-5.0000e+00 t)
```

mean for the path: 1.1200e+01
variance for the path: 1.0104e+02

PATH: 3

nodes on the path: F
probability of the path: 0.020

CDF for the path:

```
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+    -1.1338e+00 t(  0) exp(-1.0000e -01 t)
+     1.3889e -01 t(  0) exp(-1.0000e+00 t)
+    -5.1020e -03 t(  0) exp(-5.0000e+00 t)
```

mean for the path: 1.1200e+01
variance for the path: 1.0104e+02

OVERALL

CDF:

```
      1.0000e+00 t(  0) exp( 0.0000e+00 t)
+    -2.2676e -01 t(  0) exp(-1.0000e -01 t)
+    -9.7222e -01 t(  0) exp(-1.0000e+00 t)
+     1.9898e -01 t(  0) exp(-5.0000e+00 t)
```

mean: 3.2000e+00
variance: 3.7040e+01

Figure 12c. Results for Example 6

```
comm  program execution with
comm  a possibility of failure

graph
arc       e1        e2
arc       e2        e3
arc       e2        e4
arc       e3        e5
arc       e5        e7
arc       e5        z
arc       e7        e8
arc       z         e8
arc       e4        e6
arc       e6        e8
end

expo F(f, u) 1-f, 0, 0\
             -(1-f), 0, -u
exit      e2        prob
exit      e5        prob
prob      e2        e3        p23
prob      e5        e7        p57

dist      z         zero
dist      e1        F(f1, u)
dist      e2        F(f2, u)
dist      e3        F(f3, u)
dist      e4        F(f4, u)
dist      e5        F(f5, u)
dist      e6        F(f6, u)
dist      e7        F(f7, u)
dist      e8        F(f8, u)
end
```

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

MASTER COPY - FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER AFOSR-TR- 85-0385	2. GOVT ACCESSION NO. N/A
3. RECIPIENT'S CATALOG NUMBER N/A	
4. TITLE (and Subtitle) Performance and Reliability Analysis Using Directed Acyclic Graphs	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robin A. Sahner and Kishor S. Trivedi	8. CONTRACT OR GRANT NUMBER(s) AFOSR-84-0132 ~~ARO-DAAG29-84-0045~~ ~~NSF-MCS-830200~~
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Duke University Durham, NC 27706	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A5
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research ~~and U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709~~ Bolling HFB DC 20332	12. REPORT DATE April 4, 1985
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Performance evaluation, reliability evaluation, fault trees, series-parallel graphs, reliability block diagrams, stochastic activity networks

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A powerful model for the stochastic analysis of directed acyclic graphs is developed. These graphs represent event-precedence networks where events may occur serially, probabilistically, or concurrently. When a set of events occurs concurrently, the condition for the set of events to complete is that any specified number of the events must complete. This includes the special cases that one or all of the events complete. The distribution function associated with an event is assumed to have exponential polynomial form. Further generality is obtained by allowing these distributions to have a mass at the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract

origin and/or at infinity. The distribution function for the time taken to complete the entire graph is computed in a semi-symbolic form. Applications of the model for the evaluation of concurrent program execution time and to the reliability analysis of fault-tolerant systems are discussed.

END

FILMED

11-85

DTIC