NUREG/CR-4179
UCID-20350
Vol. 2

January 1986

DIGRAPH MATRIX ANALYSIS FOR SYSTEMS INTERACTIONS AT INDIAN POINT UNIT NO. 3

VOLUME II - APPENDIX A

NUREG CR-

UCRL -

DIGRAPH MATRIX ANALYSIS FOR SYSTEMS INTERACTIONS AT INDIAN POINT UNIT 3

Volume 2 - Appendix A

Overview of Digraph Matrix Analysis

H. P. Alesso. T. J. Altenbach, P. Prassinos,
D. Lappa, C. Kimura, C. Patenaude,

Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, California 94550

I. J. Sacks, B. C. Ashmore, D. C. Fromme, M. Hershberger

Analytic Information Processing, Inc.

C. F. Smith, W. J. Williams

Science Applications, Inc.

Prepared for
U. S. Nuclear Regulatory Commission

bmc
July, 1984
1207b

APPENDIX A

Table of Contents

List of Figures

APPENDIX A

OVERVIEW OF DIGRAPH MATRIX ANALYSIS

A.1 Introduction and Purpose

Digraph Matrix Analysis (DMA) is an approach to the assessment of the safety of large complex systems. The approach is based on the use of directed graphs (digraphs) containing logic connectives to represent the propagation of failure through a physical network of interrelated systems, subsystems and components.

The purpose of this Appendix is two-fold. First, a discussion of the application of the DMA approach to a single system is presented. This discussion is intended to provide a basic understanding of the conventions and procedures used in modeling and analyzing individual systems. The second purpose is to describe the overall procedure of applying DMA to a nuclear power reactor (i.e., a complex group of interacting systems). In addition to these two purposes, later sections of this appendix address additional topics of relevance to digraph modeling and analysis.

A.2 The Basics of DMA System Analysis

A.2.1 Terminology and Approach

There are three major steps in a DMA failure assessment of a system; these are:

- o Construction of the system digraph model based on plant schematics, piping and instrumentation diagrams, operational procedures, and other relevant documentation.

- o The processing of the digraph model using a graph-based "reachability" code to determine failure paths through the model and subsequently minimal cut-sets.
- o Subsequent expansion of the digraph model through the use of unit models to incorporate additional detail regarding support component dependency.

These steps are carried out on an iterative basis to provide an analysis that is ultimately sufficiently detailed to identify potential interactions that may be subtle in nature. Figure A-1 presents a block diagram description of the major steps in the DMA procedure for modeling a given system.

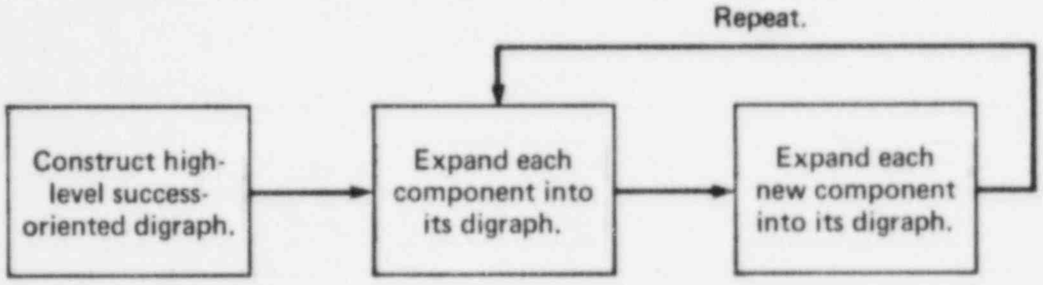

Repeat.
Construct high-level success-oriented digraph.
Expand each component into its digraph.
Expand each new component into its digraph.

Figure A-1. The DMA Procedure

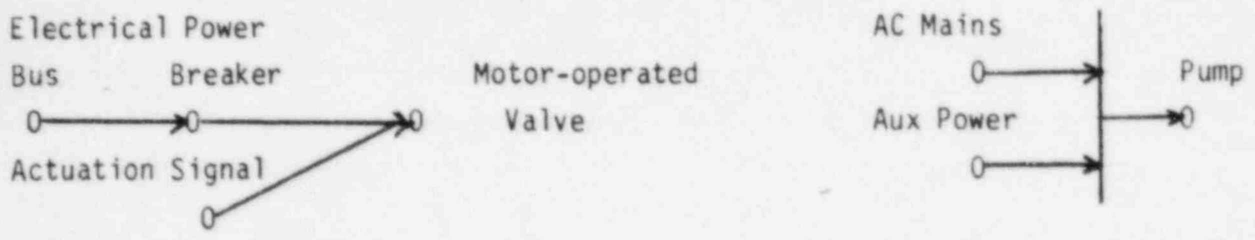
The first of these steps begins with the identification of all of the components directly necessary for successful system operation. These components are represented by nodes in a digraph model which is developed to closely resemble the physical layout of the piping and instrumentation drawing (P&ID) or other schematic drawings. To this layout of nodes representing necessary components directional connectives and logic gates are added. The resulting directed graph with logic connectives, or digraph, is a system model in which the logic relationships between components required for system functioning have been explicitly incorporated.

The principal logic conditions incorporated in digraph models are the OR gate and the AND gate. These are discussed in the following paragraphs.

If a component requires the successful operation of two support components (i.e., failure of either of these support components would result in failure of the basic component), then the nodes representing the support components would be connected to the node representing the basic component by an OR gate. For example, a motor-operated valve may require both electrical power and an actuation signal for successful operation. The nodes representing the two support components would be connected to the node representing the valve by an OR gate. Figure A-2a shows the convention used in DMA for representing this example of an OR gate connection. Note that the DMA convention for OR gate representation is multiple inputs into a single node; no other special indication or symbol is used to represent the OR gate.

DMA allows an alternative OR condition. Components in series also indicate an OR condition in that any failure in a chain, fails the connecting component. These two OR configuration facilitate the DMA models' ability to resemble the P&ID.

If a component requires the successful operation of only one of a group of alternative support components, the nodes representing these support components would be connected to the node representing the basic component using an AND gate. For example, a pump might be supplied with electrical power from AC mains or from an auxiliary generator. Failure of the pump would result from failure of both AC main power and the auxiliary generator. The use of the AND gate is shown in Fig. A-2b. The notation used is that of Petri Net theory [A-1].



a) Use of the OR gate b) Use of the AND gate

Figure A-2. Conventions for the Use of AND and OR gates.

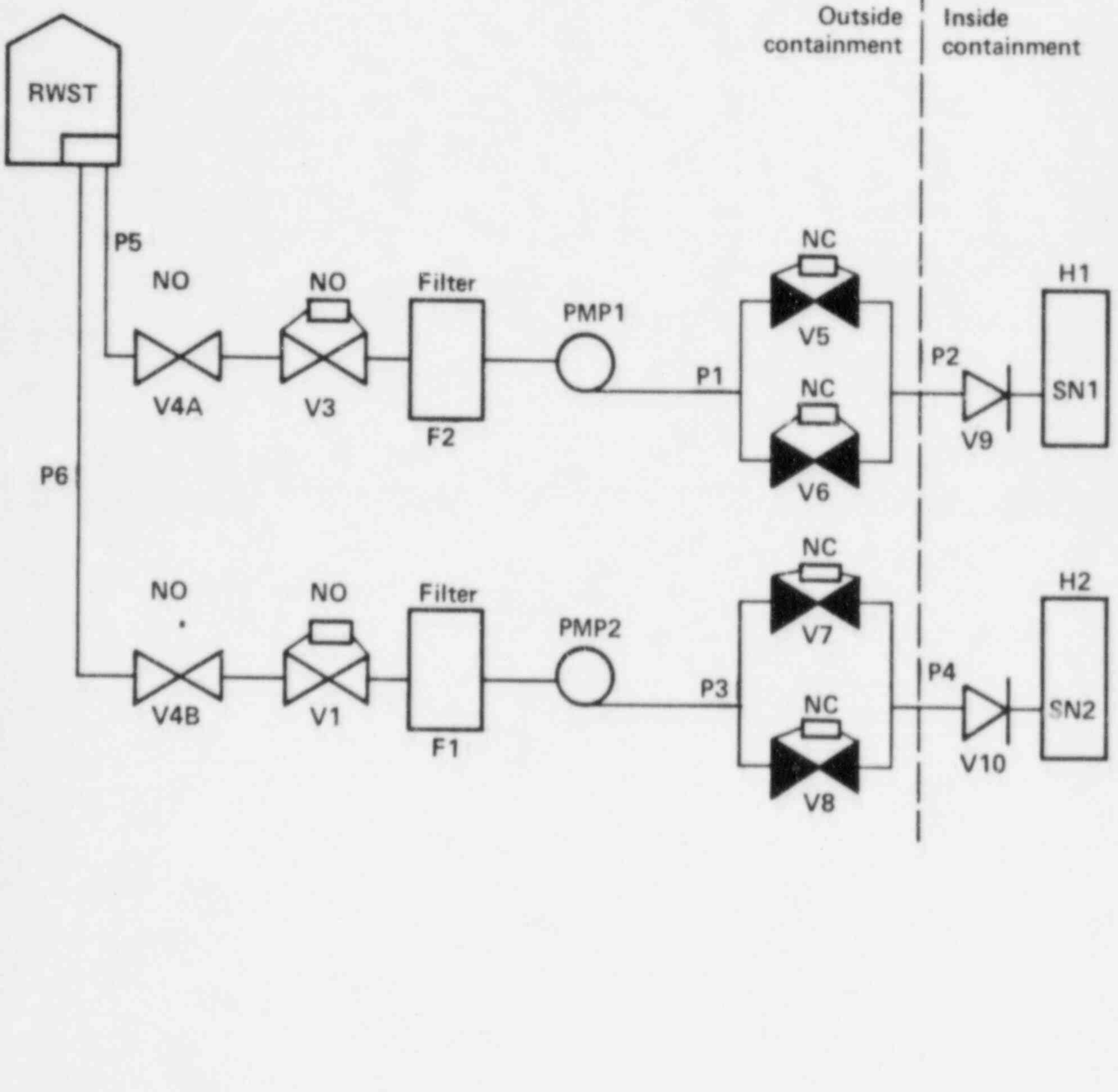
The arrows on the connectives or "edges" between the nodes representing the components in the system indicate the direction of propagation of the <u>effect</u> of information, physical movement, power, etc. among the components. The digraph model thus contains the physical components directly responsible for the functioning of the system along with the logical relationships among the components required for this functioning.

A.2.2 <u>A Simplified System Example</u>

A simplified system schematic is shown in Fig. A-3A. In this example, water from the refueling water storage tank (RWST) flows through two parallel paths to the spray into containment (CONT). Figure A-3B shows the corresponding digraph model which includes nodes representing not only the hardware of the original schematic, but also instrumentation (TS1, TS2, C1 and C2) that is required for proper system operation. Either pump (PMP1 or PMP2) will fail if its supply of water or its control signal fails, thus there is an OR gate that joins the filter (F2) and controller (C1) to the pump (PMP1). In addition, spray into containment will fail only if spray from both paths fail; thus the spray nozzles are joined to CONT by an AND gate to express this criterion.

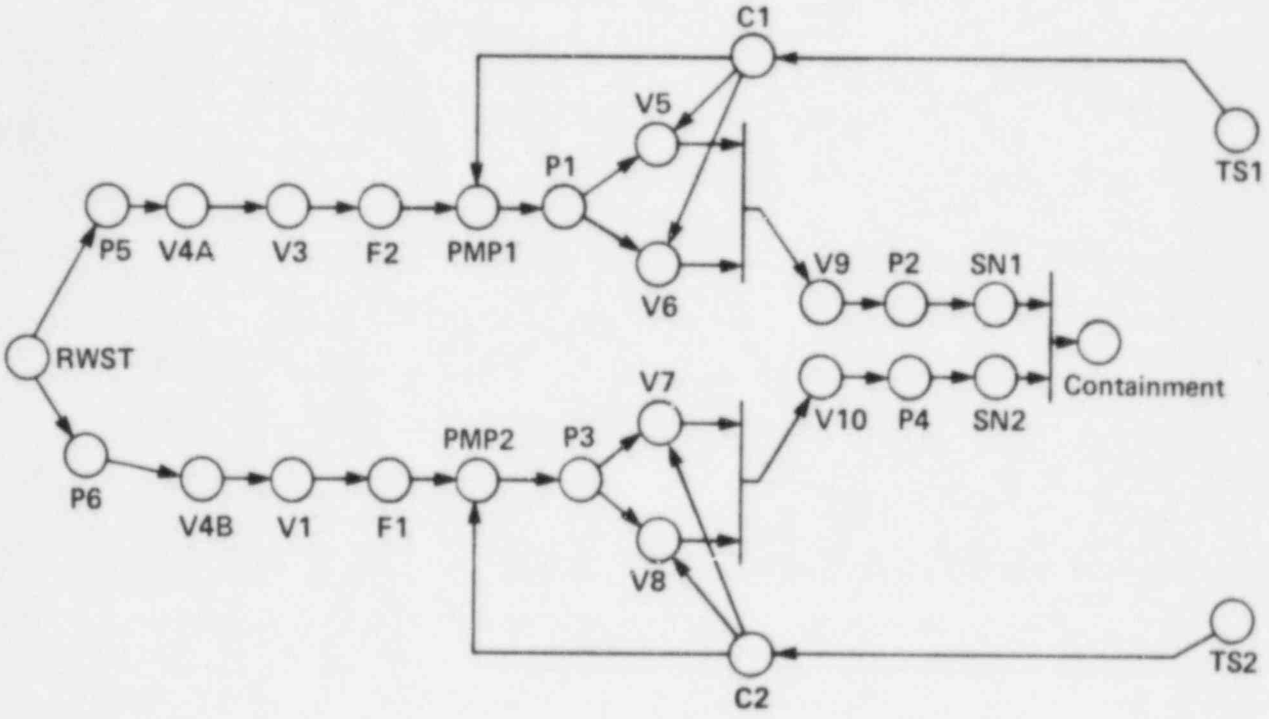
Figure 3A

Simplified Spray System



Abbrev.	Component Name
RWST	Refueling Water Storage Tank
P5	Pipe 5
V4A	Valve 4A
V3	Valve 3
F2	Filter 2
PMP1	Pump 1
P1	Pipe 1
V5	Valve 5
V6	Valve 6
V9	Valve 9
P2	Pipe 2
SN1	Spray Nozzle 1
CONT	Spray into containment
SN2	Spray Nozzle 2
TS1	Temperature Sensor 1
C1	Controller 1
P6	Pipe 6
V4B	Valve 4B
V1	Valve 1
F1	Filter 1
PMP2	Pump 2
P3	Pipe 3
V7	Valve 7
V8	Valve 8
V10	Valve 10
P4	Pipe 4
TS2	Temperature Sensor 2
C2	Controller 2

Figure A-3b. Digraph of Simplified Spray System



The sets of single component failures (Singletons) and sets of double component failures (Doubletons) that result in overall system failure for the example can be determined by inspection of Fig A-3b. For example, RWST is a Singleton since it supplies both parallel flow paths. Pairs of components taken from alternate flow paths form Doubleton pairs except for those involving V5, V6, V7, and V8. This is because these components are included in the system in a double parallel configuration. The computer algorithms that solve this problem, however, are capable of evaluating graphs with thousands of components.

A.2.3 Expansion Via Unit Models

The basic digraph is expanded by replacing components with their appropriate models. These unit models incorporate the direct dependence of a given component on support components, and their inclusion in the system digraph is intended to allow the analyst to uncover additional failures which are introduced by support components. The expansion of the components in the digraph using unit models can lead to the identification of common cause failures between components due to shared support components. A typical unit model for an active component might include electric power, control, lubrication and operator or maintenance inputs. In addition, the location of the component can be represented as an input to the component in order to identify common cause failures that might result from a single initiator affecting multiple components at a common location. A simplified unit model for a pump is shown in Fig. A-4.

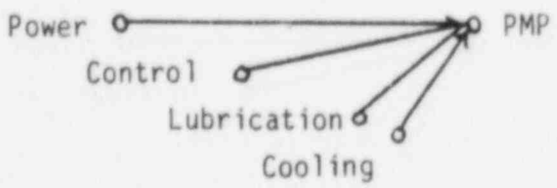


Figure A-4 Unit Model for a Pump

In this mode, failure of control, power, cooling or lubrication will cause the pump to fail. Failure of the pump could also be caused by the propagation of an effect from its location, by an operator action, or by incorrect maintenance practices. The failure due to location could be an external event such as a fire or an internal event such as the explosive failure of another component which shares the location. The discovery of Singletons and Doubletons involving location may be as significant as the discovery of any other component failure sets. There are other possible inputs to the unit model. For example, component manufacturer could be included with a resulting expansion of the failure sets to include common manufacturer. Thus, common mode failure could be included in the analysis.

Most vital components such as pumps, valves, etc. are supplied with redundant power systems. Redundancy in the unit model is represented by connecting the redundant supplies to the component via an AND gate, as shown in Fig. A-5. In this model, the pump is assumed to have redundant power supplies (Main and Auxiliary) as well as redundant controls (AUTOMATIC and OPERATOR). Note that there are two operator inputs (nodes) in this model representing both:operator takes a wrong action (OPW) and failure of operator to do the right action (OPR). Thus, the operator could mistakenly turn off the pump (OPW); or, the operator could fail to override a control failure (OPR).

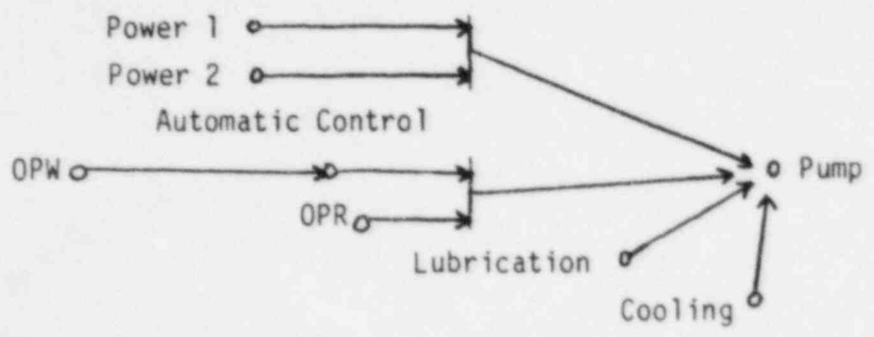


Figure A-5. Pump with Redundant Power and Control

Each of the components identified in the initial system digraph can thus be expanded by unit models. In this process, generic unit models (to be used for like-component types) can be developed for efficient modeling. These generic unit models can be used repetitively for similar components (e.g., motor operated valves), with appropriate adjustment of the unit model inputs.

A partial first level unit model expansion of the digraph of Fig. A-3b is shown in Fig. A-6. In general, a complete system digraph, such as Fig. A-6, will not be drawn by the analyst. The complete system digraph model is created by adding the data for each unit model to the data input list which describes the system digraph model of the previous expansion.

New components which are identified by the unit model expansion procedure can next become the center for continued unit model expansion. For example, power could be expanded to include appropriate motor control centers, buses, switches, relays, transformers, etc. As this expansion proceeds, components, locations operators and maintenance shared by systems may be discovered. As a result, the digraph can grow to a very large size. Singletons and Doubletons which arise through this expansion will not be apparent to the analyst or team

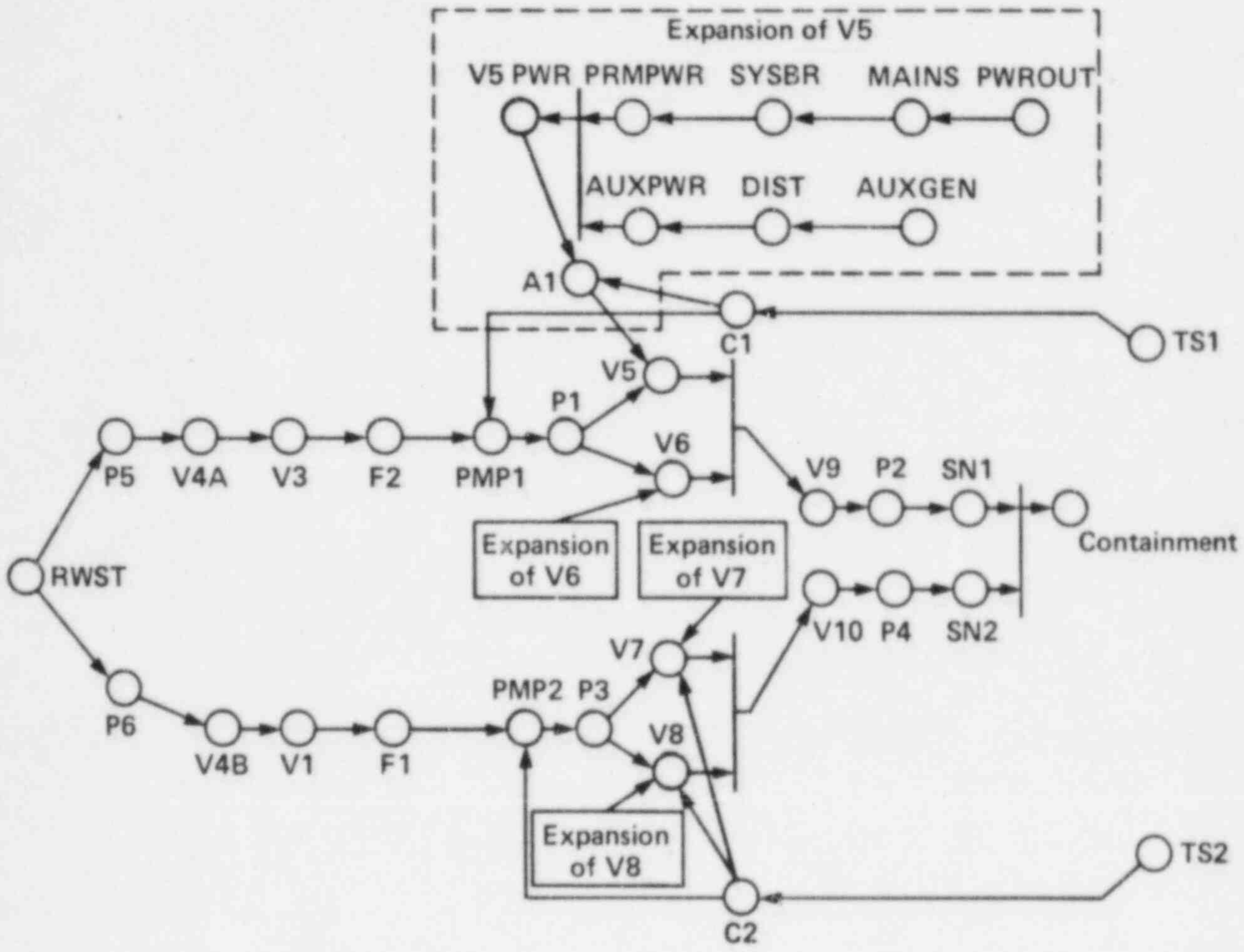


Figure A-6. Partial First Stage Unit Model Expansion of the Digraph of the System.

of analysts constructing the model and require a computer processing step for analysis.

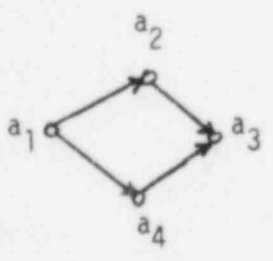
The DMA code, based on a reachability calculation, will be explained in the following section.

The large size of the complete digraph model also requires a procedure to divide this digraph into smaller equivalent units, each of which can be processed independently. This procedure is called "partitioning"; the results from the processing of each partition are then combined to yield the global digraph results. The partitioning procedure is described in a later section of this appendix.

A.2.4 DMA Computer Processing

The connectivity of a network can be represented as a graph (partially or completely directed), G, or equivalently as an adjacency matrix, A. Figure A-7 shows a typical graph and its adjacency matrix. The results that define an adjacency matrix are as follows:

$$a_{ij} = \begin{array}{l} 1 \text{ if node i and node j are directly connected} \\ 0 \text{ otherwise.} \end{array}$$



To

$$\begin{array}{c|cccc|} & a_1 & a_2 & a_3 & a_4 \\ \hline a_1 & 0 & 1 & 0 & 1 \\ a_2 & 0 & 0 & 1 & 0 \\ a_3 & 0 & 0 & 0 & 0 \\ a_4 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

Figure A-7. Graph and Corresponding Adjacency Matrix.

The adjacency matrix can be viewed as describing the possibility of flow from node i to node j. That is, an adjacency matrix is a matrix representation of a flow network directed graph. One-way flow in a network for a pair of nodes (i,j) for which flow propagates from i to j but not from j to i it is represented as:

$$a_{ij} = 1$$
$$a_{ji} = 0.$$

The determination of whether a given node is reachable from any other node can be made by Boolean manipulation of the adjacency matrix. The connectivity between All pairs of nodes in a network is contained in the Reachability matrix R. The reachability matrix can be derived from the following property (transitive property):

Connection from element k to element n = a_{kn}

Connection from element n to element l = a_{nl}.

Hence the connectivity between nodes k and l (i.e., the matrix element a_{kl}) is derived from two terms, a_{kn} and a_{nl} both of which must be nonzero for a nonzero a_{kl}; using the Boolean product operation (logical AND),

$$a_{kl} = a_{kn} * a_{nl} \text{ where } \begin{array}{l} 1*1 = 1 \\ 1*0 = 0 \\ 0*1 = 0 \\ 0*0 = 0 \end{array}$$

In matrix notation, for a network containing n nodes

$$R_2 = [A]*[A] = \begin{vmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1n} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nn} \end{vmatrix} \begin{vmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1n} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nn} \end{vmatrix}$$

Where R_2 represents the reachability matrix for all paths that require exactly two steps between all pairs of nodes.

For connectivity in exactly m steps, the reachability matrix becomes

$$R_m = [A]^m \quad .$$

Thus, the reachability matrix connections of all lengths between node pairs of any number of steps is given by

$$R = \sum_{m=1}^{\infty} [A]^m$$

where the summation represents the Boolean sum (logical OR) operation, i.e.,

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 1$$

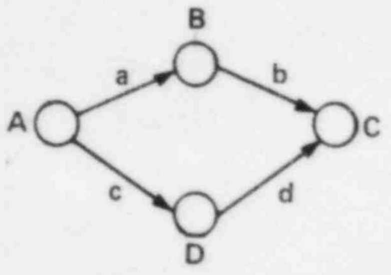
$$0 + 0 = 0$$

It can be shown that the R matrix converges to a steady-state value, that is,

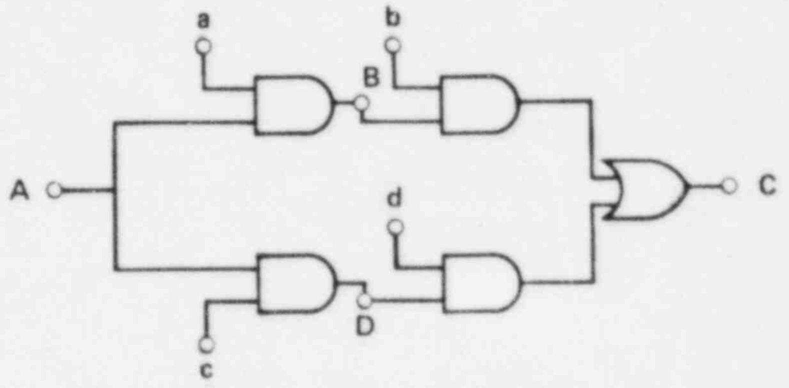
$$R_{ss} = \lim_{m \to \infty} \sum_{n=1}^{m} A^n$$

This procedure is computationally inefficient and many algorithms have been developed to more efficiently perform the reachability calculation, two of the more efficient being algorithms developed by Warren [A-3] and Warshall [A-2].

Sacks [A-4] has extended the concept of reachability to conditioned graphs*. A conditioned graphs is a representation of a logical network. This extension forms the basis of DMA. The weighted graph of Fig. A-8a is equivalent to the logic network of Fig. A-8b where the weights a and b are taken as binary variables.



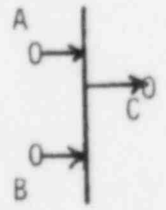
(a) Weighted Graph



(b) Logic Network

Figure A-8. Weighted Graph and Logic Network

The weighting in the graph represents a control on the connectivity of the graphs. The conditioned graph can be represented by an equivalent matrix representation. Figure A-9 shows the logical AND symbol, along with the Boolean equation that it represents and an equivalent conditioned adjacency matrix.



$C = A*B$

$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 0 & B \\ B & 0 & 0 & A \\ C & 0 & 0 & 0 \end{array}$$

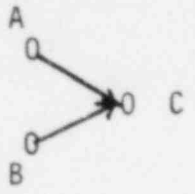
(a) AND Gate (b) Boolean Equation (c) Matrix Representation

Figure A-9. Representation of an AND Operation.

*This technique is somewhat similar to that proposed by Chamow [A-5].

The matrix of Fig. A-9 is read in the same from-to manner as before, that is, to get <u>from</u> A <u>to</u> C requires B. Note that C can be reached either from node A with B or from node B with A. These two adjacencies are equivalent.

Figure A-10 shows the logical OR symbol, along with the Boolean equation that it represents, and an equivalent adjacency matrix.



$C = A + B$

$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 0 & 1 \\ B & 0 & 0 & 1 \\ C & 0 & 0 & 0 \end{array}$$

(a) AND Gate (b) Boolean Equation (c) Matrix Representation

Figure A-10. Representation of an OR Operation.

The matrix is read in the same from-to manner as above; node C can be reached from either node A or node B.

Combinations of AND and OR gates are easily represented in the conditional adjacency matrix format. Figure A-11 is a logic network composed of two OR gates and one AND gate.

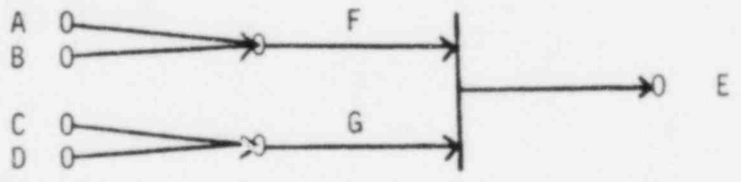


Figure A-11. Example of Multiple Gate Logic Network.

The individual component logic adjacency matrices for this network are given in Fig. A-12.

$$\begin{array}{c|ccc} & A & B & F \\ \hline A & 0 & 0 & 1 \\ B & 0 & 0 & 1 \\ C & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|ccc} & C & D & G \\ \hline C & 0 & 0 & 1 \\ D & 0 & 0 & 1 \\ G & 0 & 0 & 0 \end{array} \qquad \begin{array}{c|ccc} & F & G & E \\ \hline F & 0 & 0 & G \\ G & 0 & 0 & F \\ E & 0 & 0 & 0 \end{array}$$

Figure A-12. Component Matrices

These matrices can be combined into the single adjacency matrix shown in Fig. A-13, which represents the entire network of Fig. A-11.

$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ B & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & G & 0 & 0 \\ G & 0 & 0 & 0 & 0 & F & 0 & 0 \end{array}$$

Figure A-13. Adjacency Matrix for Logic Network of Figure A-11.

The reachability calculation procedure described previously can be applied to this adjacency matrix to yield all Singletons and Doubletons of the network of Fig. A-11. In so doing, the question we are attempting to answer is: Can node E be reached from any single node alone or from any combination of two nodes alone?

If the adjacency matrix of Fig. A-13 is Boolean AND-ed with itself and the result then OR-ed with the original adjacency matrix, the conditioned reachability matrix of Fig. A-14 results. This matrix is read in the same manner as the adjacency matrix; for example node E can be reached from node A with node G.

$$R = \sum_{m=1}^{2} [A]^m \quad \begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 1 & 0 & 0 & 0 & G & 1 & 0 \\ B & 0 & 1 & 0 & 0 & G & 1 & 0 \\ C & 0 & 0 & 1 & 0 & F & 0 & 1 \\ D & 0 & 0 & 0 & 1 & F & 0 & 1 \\ E & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & G & 1 & 0 \\ G & 0 & 0 & 0 & 0 & F & 0 & 1 \end{array}$$

Figure A-14. Reachability Matrix for the Network of Figure A-11.

Column E forms the basis of finding Singletons and Doubletons to node E. Substitution for G by all nodes which reach node G then yields network connectivity to E.

When this substitution process is done cut-sets to degree 2 are found. It is seen that the pairs

A, G

A, C

A, D

F, G

F, C

D, B, etc.

are Doubletons to node E.

The process described above was implemented with various capabilities in three mainframe computer codes, REACH, CLAMOR and SQUEAK, at Lawrence Livermore National Laboratory [A-6, A-7]. These codes are capable of finding reachability sets of any order which reach any node. CLAMOR and SQUEAK, require a large computer (CDC 7600 or equivalent) to process problems of about 200 nodes. Processing times are on the order of 30 minutes but these codes find all cut-sets of any size.

REACH has restricted the analysis to Singletons, Doubletons, and special case Tripletons, and is a faster code which runs on a CRAY-1 and is capable of processing large problems containing thousands of nodes in several minutes .

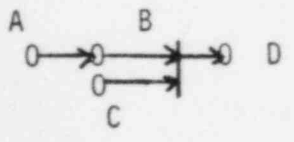
A minicomputer version has been developed for a PDP-11 and consists of several interfacing codes [A-9]. These codes will be described next.

A.2.5 Efficient DMA Codes

The present version of DMA utilizes a family of computer programs which work together to find Singleton, Doubleton, and specified Tripleton cut-sets of a digraph on a minicomputer. Figure A-16 shows the data processing flow used for the present implementation of DMA. A brief description of the operation of each program will now be given:

o <u>ADJ</u>

Program ADJ converts alphanumeric adjacency element data into numeric input and provides a count of the number of variables used in the alphanumeric input data file. Example input to the output from ADJ is shown in Fig. A-15.



a) Digraph

A,B,1	1-1	2,3,1
B,D,C	2-A	3,4,5
C,D,B	3-B	5,4,3
0,0,0	4-D	0,0,0
	5-D	
b) Adjacency List (input)	c) Variable List	d) Adjacency List (output)

Figure A-15. Input/Output for Code ADJ.

The 0,0,0 at the end of the alphanumeric data is used to indicate the end of the input data stream.

o <u>CONDENSE</u>

Program CONDENSE removes "redundant" node numbers from the numeric adjacency element list by a process of forward condensation. The rule for forward condensation is:

If a node is adjacent to only one other node, its number can be replaced by that of the adjacent node.

The digraph of Fig. A-15a condenses into the digraph of Fig. A-17a. The <u>condenser</u> program also renumbers the nodes, eliminating any repeated numbers in the numeric adjacency input. Typical output is shown in Figs. A-17b and A-17c.

In effect CONDENSE creates a generic or super node that represents a list of nodes that were originally ORed together.

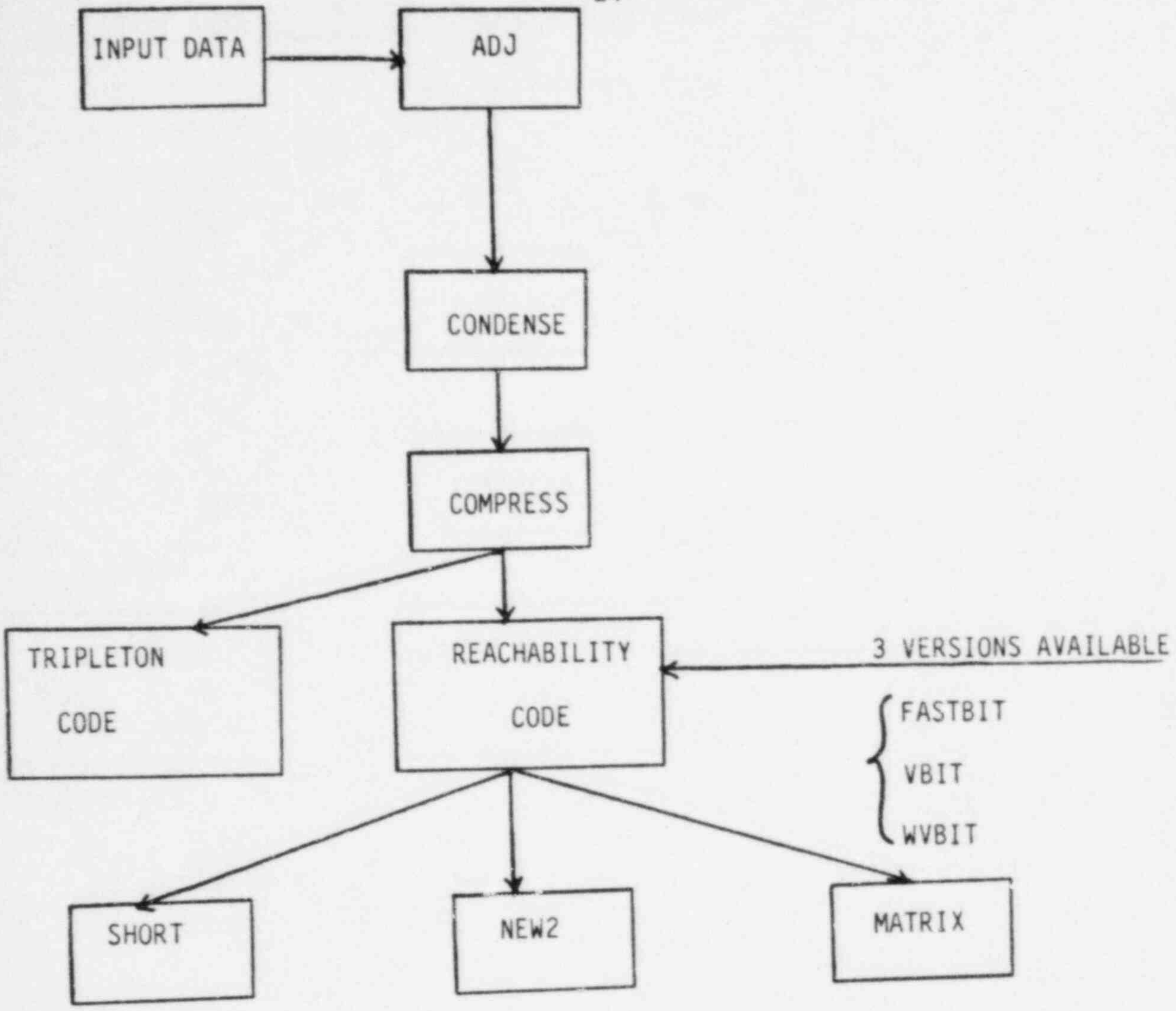
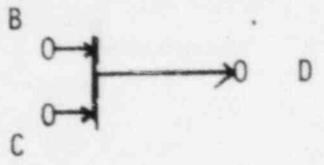


Figure A-16. DMA Data Processing Flow.



```
2,2,1
2,3,4
4,3,2

0,0,0
```

```
1-1
2-A
2-B
3-D
4-C
```

a) Condensed Digraph b) Adjacency List Output c) Variable List

Figure A-17. Condensation Program Operation

Condensation has a high payoff in the use of a matrix reachability code since processing times is approximately proportional to the third power to the order of the adjacency matrix and the computer memory requirement is proportional to the square of the order. Condensation typically reduces problem size by about 1/3. For example, an early version of a safety injection system model condensed from 1004 nodes to 625 nodes.

REACHABILITY

The reachability code finds all of the Singletons and Doubletons of the system digraph. This operation is performed using the logic shown in Fig. A-18.

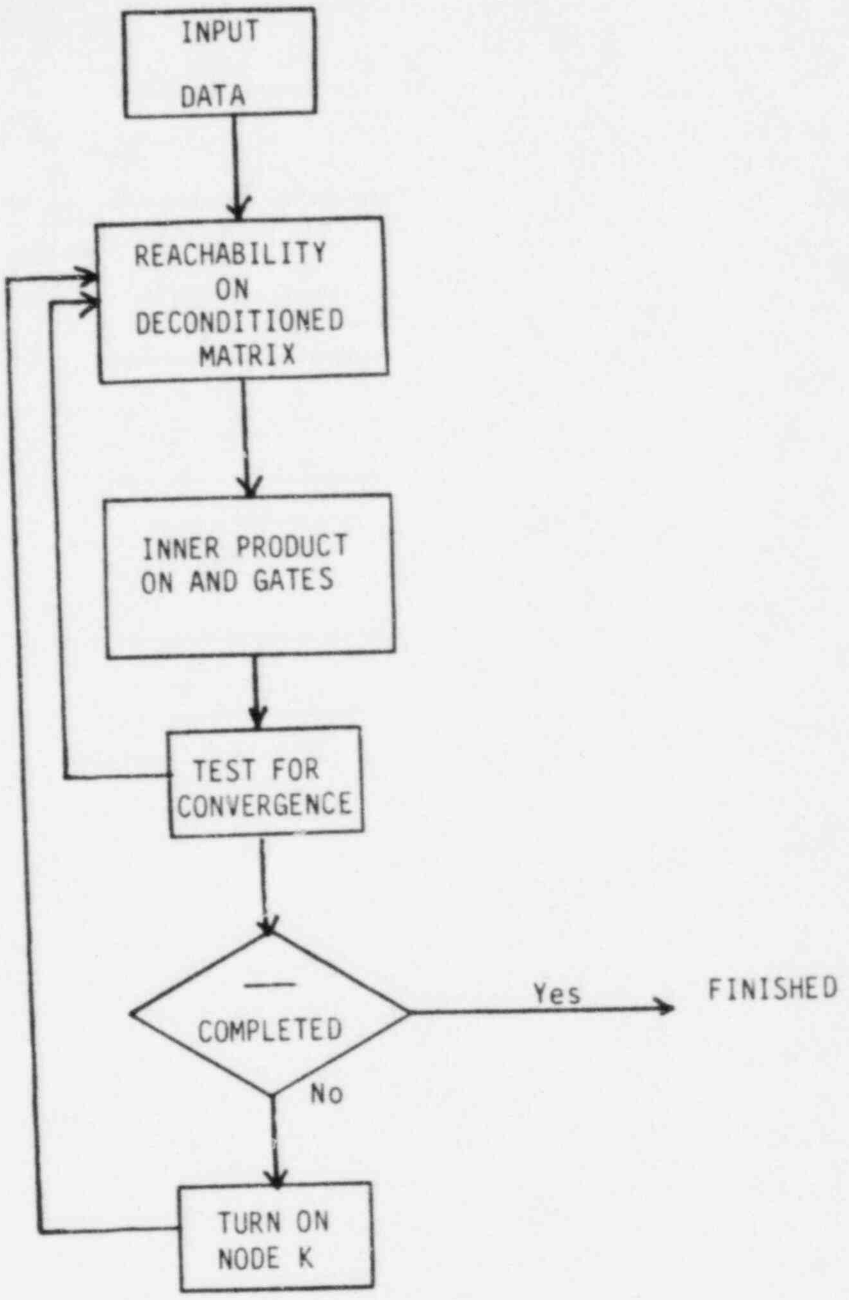


Figure A-18. Reachability Code Flow

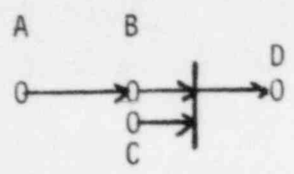
All unconditional adjacency data (e.g., A,B,1 nodes that do not connect through an AND gate) is processed by a fast binary reachability algorithm. To conserve storage and enhance speed, each element in the reachability (and also the adjacency) matrix is represented by one bit, and computer hardware Boolean logic operations are used on words containing N bits. Thus, a reachability matrix of 16 elements would take 1 x 16 words of storage in a computer with a 16 bit word size. The Warren Algorithm [A-3] is used for the reachability calculation. This algorithm appears to be more efficient for sparse matrices than the Warshall Algorithm [A-2] and is given below.

<u>Warren Algorithm</u> (Taken from [A-3])

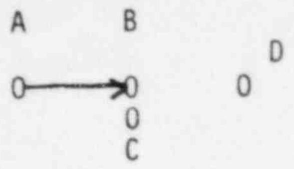
```
S1   (initialization) A is the adjacency matrix of G.
S2   Do S3 for i = 2, ..., n.
S3   Do S4 for j = 1, 2, ..., n.
S4   If A(i,j) = 1, then Do S5 for k = 1, 2, ..., n.
S5   A(i,k) = A(i,k) + A(j,k).
S6   Do S7 for i = 1, 2, ..., n-1.
S7   Do S8 for j = i+1, i+2, ..., n.
S8   If A(i,j) = 1, then Do S9 for k = 1, 2, ..., n.
S9   A(i,k) = A(i,k) + A(j,k).
S10  HALT.
```

Note: + represents the Boolean OR operation.

The result of this binary reachability calculation is the set of Singletons for the digraph with all AND gates removed. Figure A-19 shows this case.



a) Digraph



b) Deconditioned Digraph

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 \end{array}$$

c) Reachability Matrix

Figure A-19. The Deconditioned Graph

The reachability code then performs an inner product operation on each conditional adjacency entry. This inner product AND's the columns given by the two input entries (B and C in the example) and places the result in the column indicated by the output entry (D). Each conditioned entry is processed through this inner product.

The Reachability Matrix which results from this inner product process is then processed through the Warren Algorithm to "connect up" all of the new partial paths found by the inner product operation. This inner product/reachability loop is repeated until the Reachability Matrix converges. The resulting matrix is the Single Dependency Reachability Matrix which contains all Singletons of the system digraph.

Double dependency is found in a similar manner with one difference. One of the nodes in the problem is "turned on" (the failure of that component is set to TRUE) and the single dependency calculation is repeated. The resulting "single dependency matrix" is conditional on the "turned on" node. As a result, the process identifies double dependencies (Doubletons). Each of the nodes is "turned on" is succession and the corresponding single dependency matrix generated. The output ultimately contains <u>all</u> Singletons and <u>all</u> Doubletons of the model.

OUTPUT CODES

There are three types of output codes presently used. These codes generate:

1. A list of the Singletons and Doubletons (SHORT);
2. A set of Reachability Matrices (MATRIX); and
3. A Singleton/Doubleton Matrix (NEW2).

Program SHORT allows the user to search the output file generated by the reachability code for specific "reaches" either from or to a node. Using this code, it is possible to find all Singletons and Doubletons to a specific terminal node and to determine why they have occurred. A typical output from SHORT is shown in Fig. A-20.

Program MATRIX presents the reachability output file in conventional adjacency matrix format. The output from this program is composed of N+2 matrices. The first two of these are the deconditioned adjacency and single dependency reachability matrices, respectively. The other N matrices are the conditional "single dependency" reachability matrices for each of the N components taken one at a time. Typical output from MATRIX is shown in Fig. 21.

Program NEW2 generates the Singletons and Doubletons in the most useful and compact format (Fig. A-22). The Singletons are listed below the Doubleton Matrix. The Doubleton Matrix is read as follows:

> Each element i,j with an asterisk presents a Doubleton composed of component i and component j.

```
WHAT REACH PAIRS DO YOU WANT? I?J?...1,1 FOR ALL
1,J FOR ALL NODES THAT REACH J
1,4
J=    4I=    3TEMP=    5COND=    0
J=    4I=    5TEMP=    3COND=    0
J=    4I=    4TEMP=    1COND=    1
J=    4I=    2TEMP=    1COND=    5
J=    4I=    3TEMP=    1COND=    5
J=    4I=    5TEMP=    1COND=    3
J=    4I=    5TEMP=    1COND=    2
WHAT REACH PAIRS DO YOU WANT? I?J?...1,1 FOR ALL
1,J FOR ALL NODES THAT REACH J
1,3
J=    3I=    2TEMP=    1COND=    0
J=    3I=    2TEMP=    1COND=    1
J=    3I=    3TEMP=    1COND=    1
WHAT REACH PAIRS DO YOU WANT? I?J?...1,1 FOR ALL
1,J FOR ALL NODES THAT REACH J
0,0
```

Figure A-20. Output from Code SHORT

```
          ADJACENCY MATRIX
                 1 00000
                 2 00100
                 3 00000
                 4 00000
                 5 00000
     SINGLE DEPENDENCY REACHABILITY MATRIX
                 1 00000
                 2 00100
                 3 00000
                 4 00000
                 5 00000
  TWO VARIABLE REACHABILITY MATRIX            2 AND
                 1 00000
                 2 00000
                 3 00000
                 4 00000
                 5 00010
  TWO VARIABLE REACHABILITY MATRIX            3 AND
                 1 00000
                 2 00000
                 3 00000
                 4 00000
                 5 00010
  TWO VARIABLE REACHABILITY MATRIX            4 AND
                 1 00000
                 2 00000
                 3 00000
                 4 00000
                 5 00000
  TWO VARIABLE REACHABILITY MATRIX            5 AND
                 1 00000
                 2 00010
                 3 00010
                 4 00000
                 5 00000
```

Figure A-21. Output from Code MATRIX.

```
         DOUBLETON MATRIX

           2    3    5
        ______________
      2 |  -    -    *

      3 |  -    -    *

      5 |  *    *    -


                       *** SINGLETONS ***

                            4         D
```

Figure A-22. Output from Code NEW2.

A.3 Application of DMA to a Plant Safety Assessment

In Section A-2, we presented general concepts of DMA which would be applicable in any system risk analysis. In this section, we present an approach to extend the guidance for DMA in order to be able to conduct a risk assessment of a nuclear power plant.

Guidance for the implementation of DMA systems interaction assessment was published in Ref. A-10. This section summarizes the major steps and procedures in carrying out such an analysis.

The four major steps in a digraph matrix analysis are listed in Table A-1

Table A-1. Overview of Digraph-Matrix Analysis for Nuclear Power Plants.

Step 1:	Select combinations of systems for detailed evaluation. (This is equivalent to the Probabilistic Risk Assessment (PRA) event tree analysis designed to find accident sequences).
Step 2:	Construct a global digraph model for each system combination (e.g. accident sequence).
Step 3:	Partition digraph models into independent subdigraphs and find Singleton and Doubleton minimum cut-sets of accident sequences.
Step 4:	Evaluate Singletons and Doubletons on the basis of probability and display results.

Each of these steps, broken into substeps, will be discussed in turn.

Step 1: Selection of Combinations of Systems for Detailed Evaluation

This step focuses on the four safety-systems functions at a nuclear power plant and uses event tree methodology resulting in the selection of combinations of front-line systems among which system interactions might exist. This includes consideration of other plant operating modes.

This step is accomplished in a manner similar to a PRA event tree analysis. Event tree analysis is an inductive logic technique that sequentially models events, with success and failure, following a selected initiating event and proceeding to a series of logical outcomes. An event tree begins with an initiating event, and then displays a sequence of events on the system level that forms a set of branches, each of which represents a specific accident sequence whose effects relate directly from the events in the sequence. Complete event tree analysis requires the identification of all possible initiating events and the development of an event tree for each.

The first step in finding accident sequences (or combinations of systems for detailed evaluation) can be accomplished by the following six substeps.

Substep 1A: Study of Plant Design and Operating History

Analysts first gather all pertinent existing information about the plant. A large amount of information is collected, synthesized, and documented to form the basis for subsequent analytical activities. A list of plant systems is developed and reviewed for its potential impact on risk. Appropriate sources of information include design documents, safety evaluation reports, plant system descriptions and operating procedures and previous safety studies.

Substep 1b: Development of a List of Accident Initiators

The Reactor Safety Study (WASH-1400) [A-9] generic list of accident initiators is reviewed to see which apply to the plant being studied. This list should reflect applicable operating experience. The accident initiators are then grouped in terms of common mitigation requirements.

Accident-initiating events are identified and grouped according to the similarity of plant response. Generic lists, operating histories, and plant-specific data can be factored into a generalized engineering process in which exhaustive lists of initiating events, including their occurrence frequency, are compiled and grouped. Efforts must be made to ensure that the set of initiating events considered is as complete and comprehensive as practical.

Their are two major classes of accident initiators: loss of coolant accidents (LOCA) and power transients.

Substep 1c: Development of Functional Event Trees

To avoid unacceptable reactor core damage and a release of unacceptable levels of radioactivity to the site environs, four basic safety functions have been specified by the NRC:

1. To maintain the primary coolant inventory.
2. To transfer the heat from the reactor to the final heat sink.
3. To render and keep the entire core subcritical.
4. To maintain the integrity of the containment and control radioactive releases.

Systems that fail to meet either one or more of the basic safety functions are of concern.

For each group of accident initiators identified in Substep 1b, the four basic safety functions are subdivided into subfunctions; the corresponding functional event trees are then generated.

To summarize, once the group of initiating events has been selected, the attendant response of the plant must be determined. This may be accomplished through a functional analysis in which the safety functions required for each response are defined and ordered in a function event tree. Success criteria for each function are stated in terms of the required collection of systems that perform each individual function. Success criteria are then developed for individual systems and form the basis for characterizing the logic description of the top event of the system event tree. A primary value of this approach is the stepwise ordered approach of identifying broad functional considerations with specific systems. It provides a framework for the complex task of sorting system responses.

Substep 1d: Assignment of Front-Line Systems to Functional Event Trees

The safety functions utilized in preparing the functional event trees (Substep 1c) are performed by engineered systems designed specifically for this purpose. In other words, further decomposition of the safety functions into simpler functions yields the specific engineered systems. The operability of these systems defines whether the safety functions are performed or not and thus completely defines the course of an accident. These systems are called front-line systems (FLS). The success criteria for the FLS are defined in this step.

The event tree headings are defined as logic statements describing composite events representing the minimal operability states of front-line systems and their required supporting systems. This approach leads to event trees with a minimum number of event tree branch headings, and thus facilitates an understanding of the overall accident progression path. It does, however, require that support systems be included in the system models.

Each event tree heading must have a definite logic statement of the minimum acceptable complement of equipment and system performance required to successfully accomplish the function described by the event tree. These success criteria should be stated in discrete hardware terms, such as number of pumps or required flow.

Substep le: Results of Event Tree Analysis

The event tree analysis produces a set of system combinations (e.g. accident sequences). Each system combination is one sequential combination of those front-line systems whose success or failure (as specified) results in serious consequences to the plant involving the degradation of one or more safety-system functions.

In a simplifying assumption, it is possible to conservatively assume (as is done in some cases for PRA) that systems required to work in an accident sequence, work with probability equal to one. We are then left with system combination that are composed of front-line systems that must fail if there is to be a serious consequence to the plant.

Substep lf: Assignment of Support Systems to Front-Line Systems

To successfully perform their functions, front-line systems depend on the operability of other support systems. Support systems affect the accident response of a plant only through their effect on the FLS.

To identify the support systems for each front-line system, the following six procedures can be followed:

1. The operation of the front-line system must be examined in detail, identifying both the necessary inputs and all of the outputs. If, for example, the FLS is a fluid system, all potential sources of the

fluid should be identified. Next, all of the systems with which the FLSs interface directly (e.g., as discharge points) or indirectly (e.g., as secondary sides of heat exchangers) should be identified.

2. The power sources necessary for the operation of the active components (e.g., electric power and steam) should be identified.

3. The modes of actuating and/or controlling the system must be identified, in particular, whether the system is actuated and controlled automatically or by the operator action. In both cases the signals necessary to initiate the control system or operator actions must be identified. The possibility of manually overriding automatic control should also be established. In the case of automatic control, the type of controlling system should be identified (e.g., electrical pneumatic) along with the systems associated with each type (e.g., power supply, instrument air).

4. The cooling systems of the various components of the FLS should be identified.

5. The lubrication systems (if any) of the various components of the FLS should be identified.

6. The general location of the FLS should be established. More detailed location identification will support the unit model expansion of relevant components.

The support systems that contribute to the initiating event are then identified for a full compilation to be used in Step 2.

Step 2: Constructing a Global Digraph Model for Each System Combination

The accident sequences found in Step 1 are combinations of front line and related support systems that must fail in order to produce a severe consequence to the plant involving the loss of one or more vital safety functions. We will refer to these accident sequences as system combinations. In Step 2, a single global digraph model is constructed for each system combination

The digraph modeling techniques described in Sec. A.2 are first applied to each of the front line and support systems included in the system combination. Two major additional items are needed for the integration of the individual systems into a global model for the system combination. They are:

1. identification of system level failure criteria, and
2. identification of boundary nodes.

These two items are related in that system failure criteria are used to characterize how failures in a system can propagate across boundaries into other systems.

The initial digraph model of a given system will result in the identification of boundary nodes, or nodes at which connections to other systems are required. In addition, the unit model expansion process invariably results in the identification of boundary nodes from support systems, operator interfaces and location.

The coordination of boundary node identifiers is a main concern in the development of the global model. Once boundary nodes common to systems included in the accident sequence have been identified, they are incorporated into each system model with a unique node identifier. Not only shared components, but also shared location and operator actions are included in this process.

The individual system digraphs can next be combined into successive combinations of systems. As each system combination model is produced, it is checked for consistency and completeness by processing using the DMA codes discussed in Sec. A.2.5. Corrections to the system combination models are made as necessary. Ultimately, this process results in the integration of all the systems in a given system combination into a single global model.

Step 3: Partitioning of the Global Digraph Model into Independent Subdigraphs and DMA Processing

The global digraph produced from Step 2 is the digraph of the expanded operational model for the specified accident sequence. The digraph and its corresponding adjacency matrix grow larger with each expansion step. At some size, the matrix will exceed the memory space limits of the reachability code or will take excessive amounts of computer processing time. There are two ways of overcoming these computer limitations. First, we can separate the global digraph into independent subdigraphs. Subdigraphs that can not cause the failure of system(s) can be partitioned out of the model.

Partitioning into independent subdigraphs can be accomplished according to the following definition:

> A connected graph G is separate (capable of being partitioned) if G contains a subgraph g such that the complement of g (g') and g have only one vertex in common.

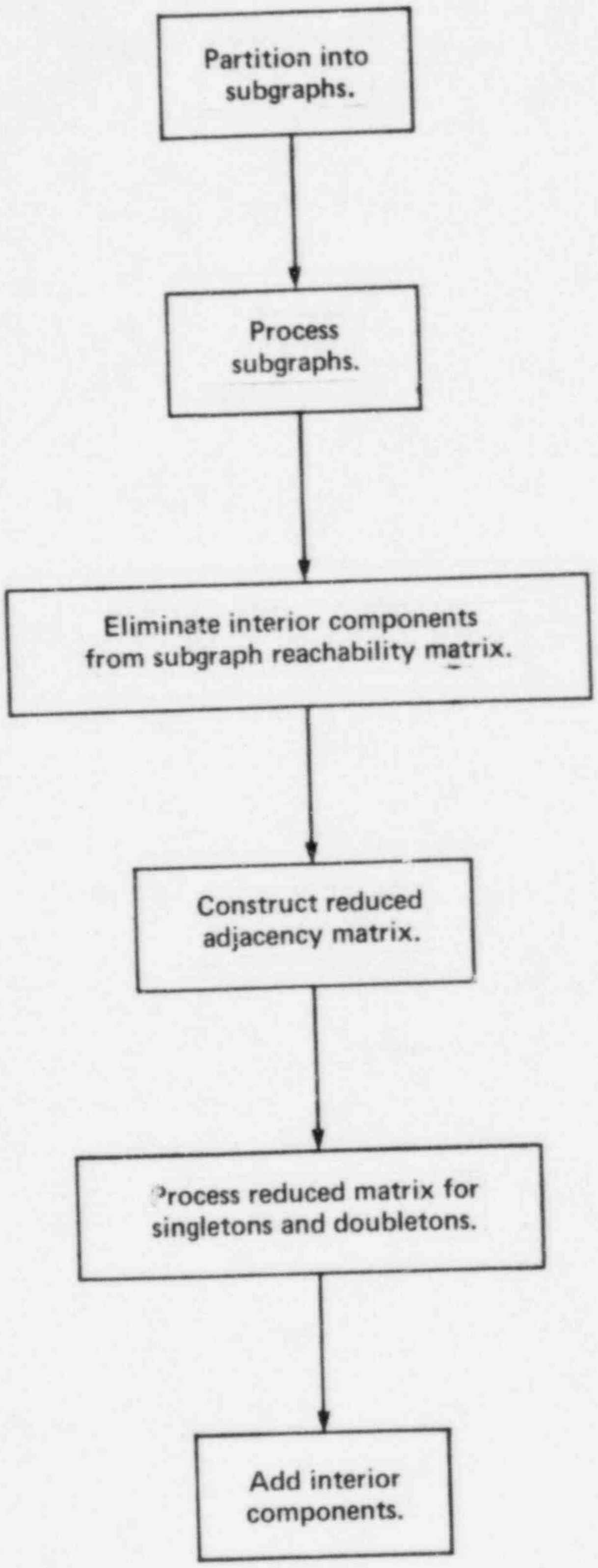
Wherever possible, analysts can partition the global digraph into independent subdigraphs.

A <u>second</u> way to overcome computer limitations involves: (1) substitution of equivalent (reduced) networks, and (2) generating generic or super nodes (these nodes can represent many other nodes that are all simply ORed together).

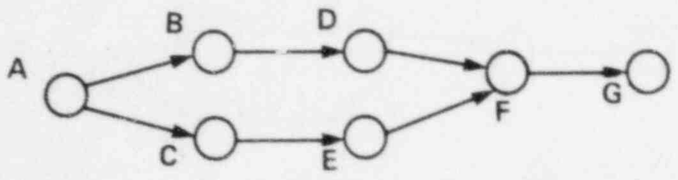
The adjacency matrix can be partitioned into submatrices that can later be recombined into the global digraph. The submatrices can be replaced by equivalent but smaller submatrices. That is, nodes that are on only the boundary of the graph represented by the submatrix are eliminated through Boolean absorption before the recombination.

The partitioning/recombination procedure must not eliminate any Singletons or Doubletons from the global digraph. The steps in this partitioning procedure, shown in Fig. A-23, will be briefly discussed in the following subsections. Figure A-24 schematically illustrates this partitioning, reduction, recombination, and expansion process. For a relatively simple digraph model.

Figure A-23. Partitioning of the Global Digraph.



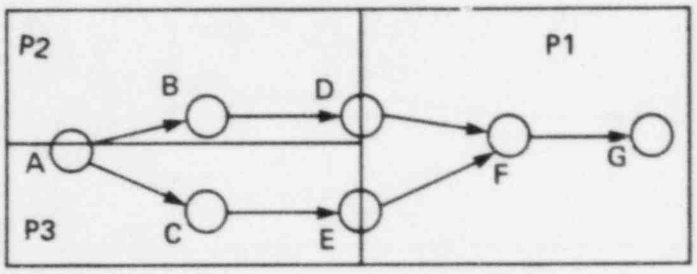
Partition into subgraphs.
Process subgraphs.
Eliminate interior components from subgraph reachability matrix.
Construct reduced adjacency matrix.
Process reduced matrix for singletons and doubletons.
Add interior components.



(a) Global digraph

$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ G & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

(b) Global adjacency matrix



(c) Digraph partitions

$$\begin{array}{c|ccc} & A & B & D \\ \hline A & 0 & 1 & 0 \\ B & 0 & 0 & 1 \\ D & 0 & 0 & 0 \end{array}$$

P2

$$\begin{array}{c|cccc} & D & E & F & G \\ \hline D & 0 & 0 & 1 & 0 \\ E & 0 & 0 & 1 & 0 \\ F & 0 & 0 & 0 & 1 \\ G & 0 & 0 & 0 & 0 \end{array}$$

P1

$$\begin{array}{c|ccc} & A & C & E \\ \hline A & 0 & 1 & 0 \\ C & 0 & 0 & 1 \\ E & 0 & 0 & 0 \end{array}$$

P3

(d) Partition adjacency matrices

$$\begin{array}{c|ccc} & A & B & D \\ \hline A & 0 & 1 & 1 \\ B & 0 & 0 & 1 \\ D & 0 & 0 & 0 \end{array}$$

P2

$$\begin{array}{c|cccc} & D & E & F & G \\ \hline D & 0 & 0 & 1 & 1 \\ E & 0 & 0 & 1 & 1 \\ F & 0 & 0 & 0 & 1 \\ G & 0 & 0 & 0 & 0 \end{array}$$

P1

$$\begin{array}{c|ccc} & A & C & E \\ \hline A & 0 & 1 & 1 \\ C & 0 & 0 & 1 \\ E & 0 & 0 & 0 \end{array}$$

P3

(e) Reachability matrices for partitions

A or B = D   D or E or F = G   A or C = E

(f) Singletons for partitions

$$\begin{array}{c|cccc} & A & D & E & G \\ \hline A & 0 & 1 & 1 & 0 \\ D & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 0 & 1 \\ G & 0 & 0 & 0 & 0 \end{array}$$

(g) Reduced adjacency matrix

$$\begin{array}{c|cccc} & A & D & E & G \\ \hline A & 0 & 1 & 1 & 1 \\ D & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 0 & 1 \\ G & 0 & 0 & 0 & 0 \end{array}$$

(h) Reduced reachability matrix

D is a singleton for G	but	B is a singleton for D
E is a singleton for G		C is a singleton for E
A is a singleton for G		F is a singleton for G

therefore the full singleton set is A, B, C, D, E, F

(i) Full set of singletons

Figure 24. Partitioning process.

Substep 3a: Partitioning of the Global Digraph Model

Analysts partition the global digraph into independent subdigraphs according to the above. Each subdigraph is labeled (i = 1, ..., n). If no independent subdigraphs are found, a "partitioner" subroutine (described below) can be used.

The partitions that are created by the "partitioner" subroutine are based on the basic structure of the digraph and the unit model expansions around this graph. The unit model expansions arise from "natural" partitions caused by operational and spatial considerations and equivalent circuits are constructed. Most of the components connected to a given valve are different from components attached to a different valve. As an illustration of this natural partitioning, consider the global digraph shown in Fig. A-24.

Each of the two "unit models" added to the structure shares only one component with the other unit model and one component with the basic structure. In addition, each of the three partitions (P1,P2,P3) shown in Fig. A-24(c) contains components that do not link outside of the partitions. Components that link outside of the partition are defined as lying on the boundary of a partition. By removing the components that are fully contained within a partition (i.e., do not lie on a boundary), the size of the adjacency matrix for each partition can be reduced in order. The "partitioner" subroutine identifies these "interior" components and records their identity. The actual selection of subgraph partitions is performed by an algorithm that traces through the digraph backwards from the components of the basic structure digraph. The tracing for each component continues until all subgraphs meet two conditions:

1. The number of components is less than or equal to the maximum size the reachability code can process.
2. A set of "interior nodes" that can be eliminated exists in each subgraph.

It is not necessary to make the subgraphs disjoint when conducting this equivalent circuit substitution. In other words, subgraphs may share common components. But shared components will not be interior nodes and therefore will not be eliminated in the subsequent processing. The next step is to process each subgraph through the reachability code.

Substep 3b: Reachability Processing of Subdigraphs

The adjacency matrices of the subgraph partitions are now individually processed through the reachability code. The reachability matrices for each of the subgraphs are shown in Fig. A-24(e).

Substep 3c: Elimination of Interior Components

Components that are totally interior to the partitions are now identified and redefined (on the basis of Boolean absorption) from the subgraph reachability matrices. The unreduced subgraph reachability matrices are retained for use later in determining if these "eliminated" components are Singletons or Doubletons of the global digraph.

Substep 3d: Construction of the Reduced Adjacency Matrix

The reduced reachability matrices of the subgraphs are now combined into a "reduced global digraph adjacency matrix," shown in Fig. A-24(g). This matrix is the same as the original adjacency matrix, but interior components have been eliminated. This matrix contains all of the connectivity information between boundary components that is contained in the original adjacency matrix.

Substep 3e: Reachability Processing on Reduced Adjacency Matrix

The reduced matrix is now processed by the reachability code. This step links up the partitions. The result of this processing is shown in Fig. A-24(h). In this processing sequence, the size of the global adjacency matrix is reduced, thus overcoming the size constraint of the code. The reduced reachability matrix contains all Singletons and Doubletons for the boundary components. In addition, processing can be carried out for selected tripletons. The "interior" components can now be considered.

Substep 3f: Addition of Eliminated Components

The reachability matrix for each of the partitions contains all essential information about the interior components. Each of these interior components that is connected to a boundary component that is a Singleton or part of a Doubleton in the reduced reachability matrix must be considered as a potential Singleton or Doubleton. The type of connection is important. If there is a direct (unconditional) element between the boundary component in the reduced matrix and the interior component, then the interior component has the same impact as the boundary component in the reduced matrix. If the interior component has a conditional relationship, the effect of this relationship must be considered.

Summary

Each accident sequence (system combination) results in a violation of at least one of the four basic safety functions. In this step, the system combination was organized in terms of the frontline and support systems. A global digraph for each system combination was constructed. Its Singleton and Doubleton element cut-sets were found through reachability processing of the digraph's adjacency and matrix.

Once all Singleton, Doubleton and selected Tripleton minimum cut-sets of the global digraph model of the system combination are found, Step 4 is used to evaluate and quantify them.

Step 4: Evaluation on the Basis of Probability and Display of Results

Once the Singleton, Doubleton and selected Tripleton minimal cut-sets for all system combinations are found, it is necessary to evaluate the systems interactions using ranking criteria. These can be evaluated, for example, by normal PRA techniques such as risk analysis, which combines the accident sequence consequence with probability of failure. In addition, quantification of complete system combination system interaction probabilities can be carried out.

The quantification of system combination probabilities is conducted by assigning a failure probability to each component which was a member of the failure sets. The data used for component failure probabilities can be taken from data sources such as: WASH 1400 [A-9], IEEE Standard 500 [A-10], the Zion Seismic Safety Study [A-11], the Indian Point-3 Probabilistic Safety Study [A-12] and others [A-13, A-14, A-15, A-16]. All component failures which appear in Singletons or Doubletons can be treated as independent. This independence assumption is accurate to the level of modeling detail in a DMA.

Any physical dependency that can be identified would be included in the quantification of the DMA model except for the following:

Common Location

Common Maintenance

Common Manufacturer

Common Environmental Conditions

Shared support systems which would make apparently independent components dependent are explicitly modeled. For example, the failure of two pumps because of the loss of a common component in the component cooling system would appear in a failure set as a Singleton in component cooling or as part of a Doubleton. The pumps which fail due to the cooling failure would not appear in the failure set. Figure A-25 illustrates this situation for a simplified case.

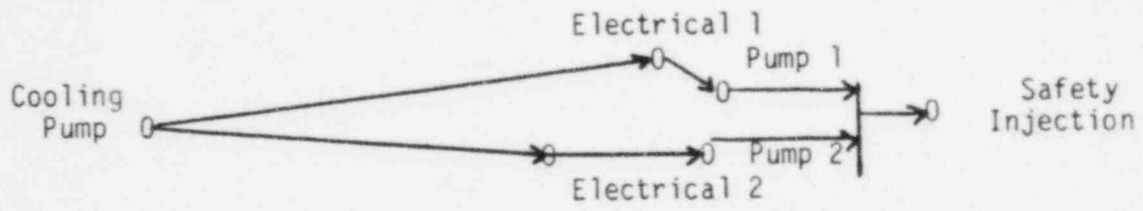


Figure A-25. Simplified Example of Common Mode Failures

The failure sets for this case are:

Cooling Pump

Electrical 1 * Electrical 2

Pump 1 * Pump 2

Electrical 1 * Pump 2

Electrical 2 * Pump 1

Notice that the pumps do not appear in a cut set with the cooling pump. The appearance of a pump in the cut set means that the pump itself fails due to internal reasons and not due to the failure of an external component.

The data bases cited previously indicate that most components have several potential failure causes. Each of these failure causes is taken as independent and combined into a single failure probability for the component by the following equation:

$$P = 1 - \prod_{i=1}^{n} (1 - P_i)$$

where n is the number of failure causes of the component and P_i is the probability of failure due to the i^{th} cause.

The total probability of failure due to all Singletons (and Doubletons) is computed using the SIGPI code developed at LLNL [A-11]. In this code the cut sets are not assumed to be independent. The SIGPI program uses two fast complementary methods of computing the probabilistic performance of complex systems: the PI method and the SIGMA method. The former exploits the fact that, when system variables are carefully defined, these variables are often statistically independent conditional to the environment in which they are embedded, a very convenient fact from a computational point of view. The later is used to compute the probability of combinations of events produced by the PI method by disjointing such events, thereby allowing the exact computation of performance. The computational complexity of the overall process is a polynomial function of the number of components. For very large problems, where costs of precise answers may be prohibitive, a relaxed accuracy can be specified, and the SIGPI algorithms will halt when that accuracy has been reached.

A.4 Additional Digraph Modeling Discussions

A.4.1 Break Modeling

The digraph modeling procedure previously described is valid primarily for the downstream propagation of effects such as system blockage. A pipe break can affect upstream as well as downstream flow. The upstream propagation results from the fact that a break acts as a sink into which liquid, for example, can drain. This sink could cause upstream components to fail to function as designed. Figure A-26 illustrates this phenomenon. A break in the pipe downstream of valve 1 could effect the flow of water from the refueling water storage tank to Pump 2.

A procedure which extends a digraph model to include the effects of the propagation of breaks upstream will now be described. This extension propagates the effects of a break both downstream and upstream. The block digraph of the system in Figure A26 is shown in Figure A27a. The digraph which propagates the effect of breaks both upstream and downstream is shown in Fig. A-27b. In this figure, the primed components (e.g., P1') represent the failure of components in the break mode, and the arrows on both ends of the connectives between the primed nodes indicate bidirectional flow. The effect of a break failure anywhere in the system will propagate to all other components.

The nodes which represent component failure as a break are connected to the nodes which represent component block failure, but the reverse is not true.

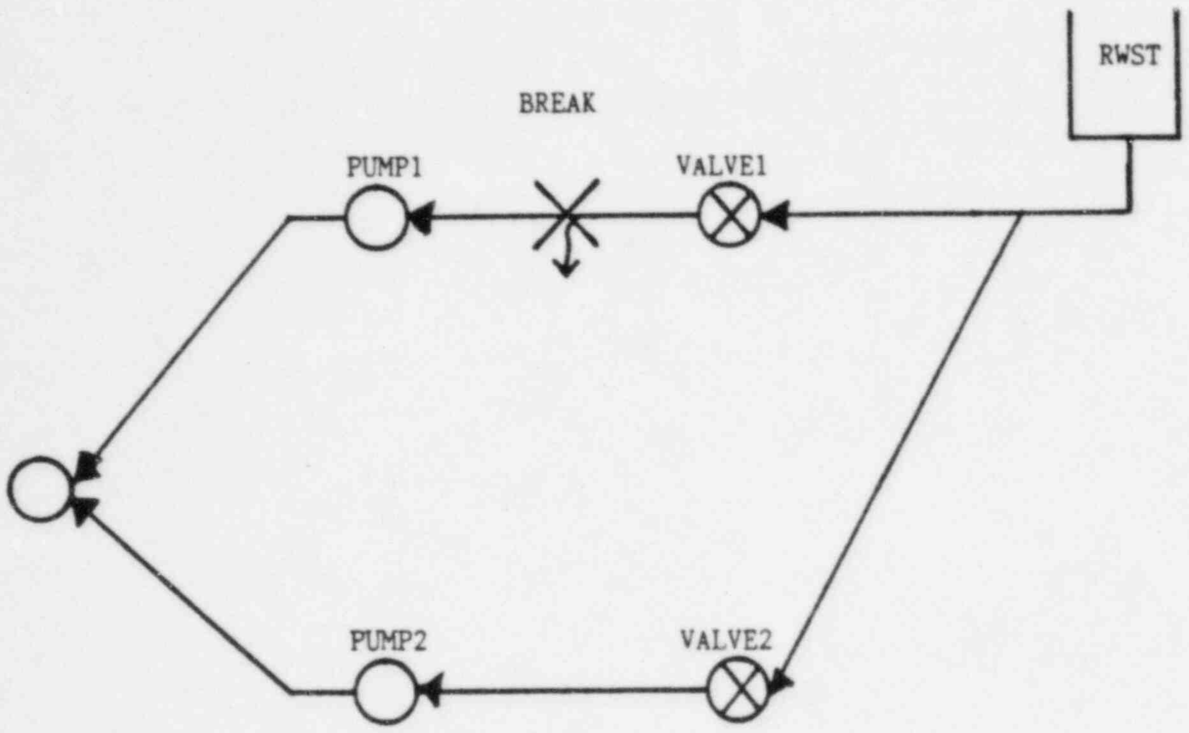
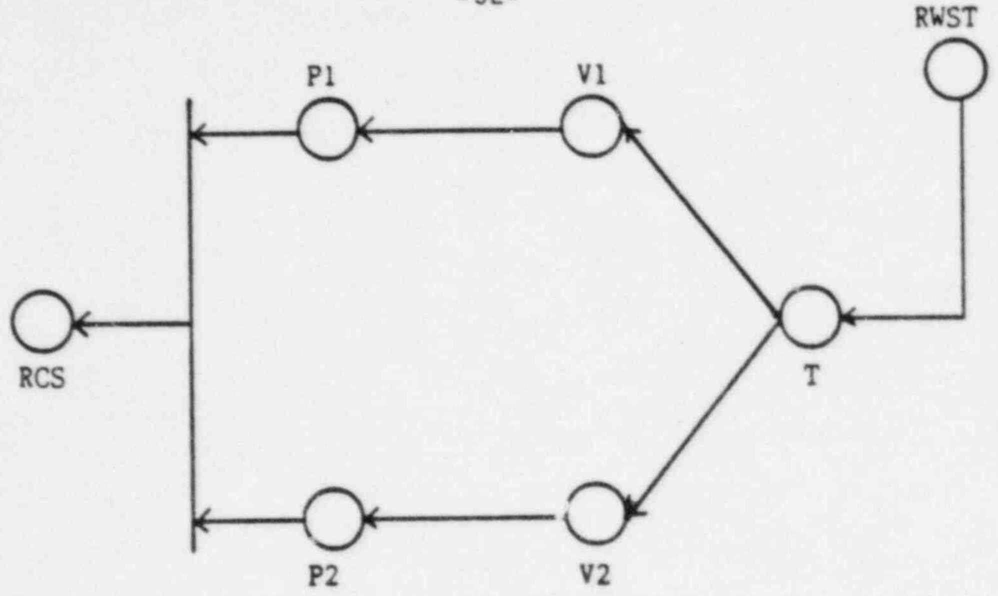


Figure A-26. Effect of Pipe Break



(a) Block Digraph

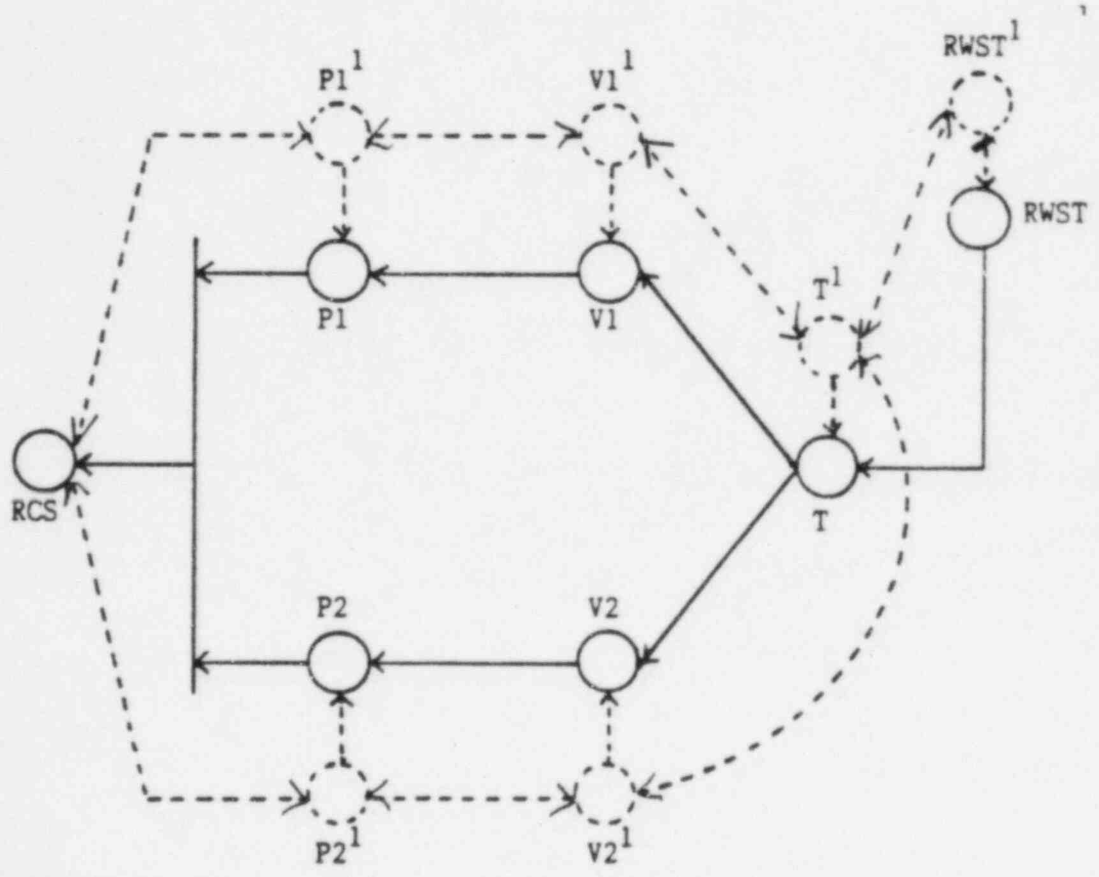


Figure A-27 Break and Block Digraphs

The addition of a break node for each component will approximately double the size of the system digraph. Fortunately, a group of bidirectionally connected nodes can be combined into a single equivalent node (or strong component) reducing the network size (see Sect. A.4.2). The effect of this reduction is shown in Fig. A-28.

Good system designs will normally have components which are used as break mitigators. In fluid flow networks, automatically or manually operated valves and check valves are used for this purpose. In electrical networks, the break (short-circuit) mitigation function is performed by circuit breakers or fuses. In DMA, these break mitigators are modeled by an AND gate on the bidirectional connective between adjacent nodes which represent the component break modes. The modeling of a typical break mitigating component is shown in Fig. A-29. In this figure, the valve A1, can be used to limit the effect of a pipe break P1', from affecting upstream components. The use of the valve as a break mitigator is indicated by the double prime in the symbol for the valve, V1". It should be noted that a break in both P1' and V1' will still propagate downstream. The nodes which represent components used for break mitigation are now candidates for unit model expansion following the procedure described in the preceding sections.

A.4.2 Modeling Complex Networks with Bidirectional Flows

The number of possible flow paths through combinations of systems is strongly dependent upon the number of switching or flow junctions in the network and can quickly become quite large. In the case of piping networks, these junctions are the pipe headers where the flow direction is controlled by

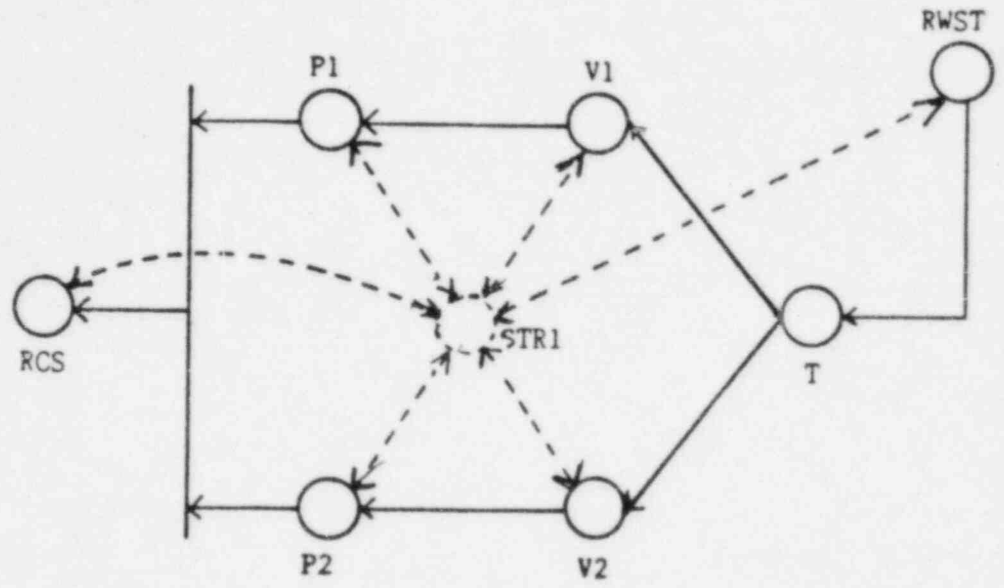


Figure A-28 Use of a Strong Component

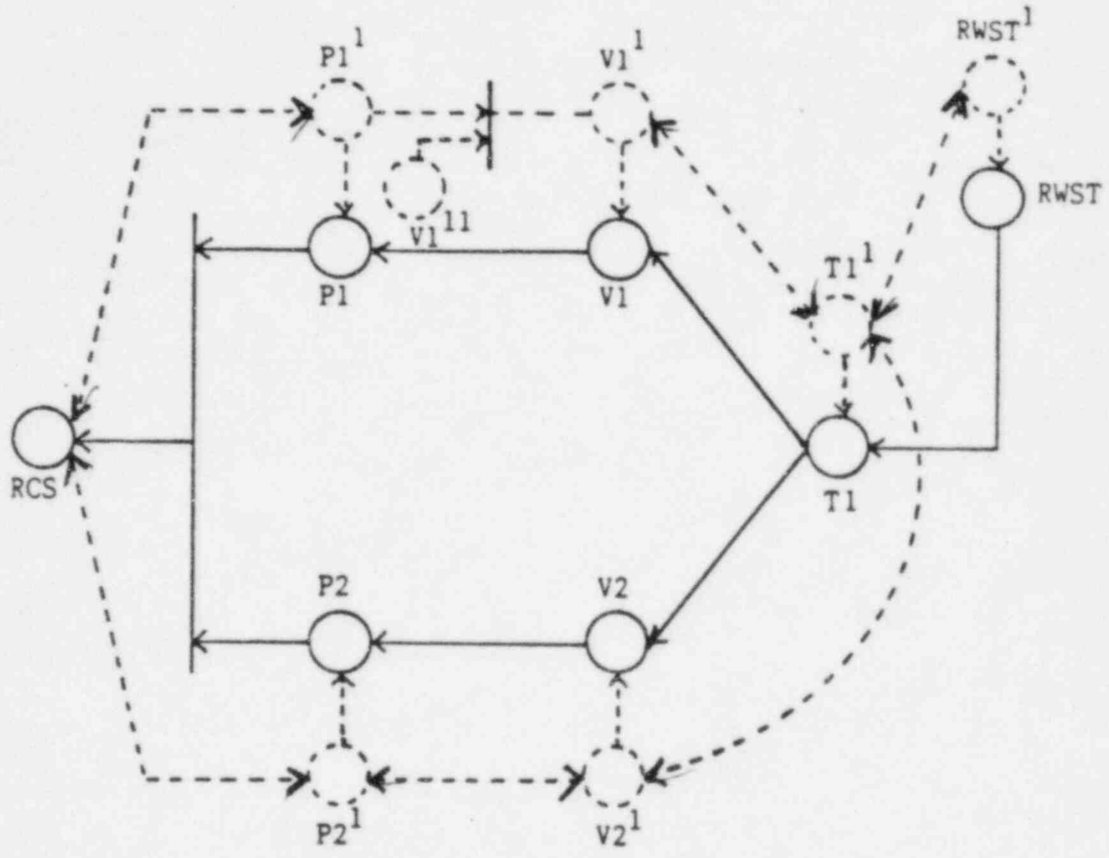


Figure A-29. Modeling of Break Mitigation.

upstream and downstream pressure conditions. Two approaches can be used to model the potential use of all of the possible paths. The first is a global method whereby an exhaustive list of all possible paths through the network from source(s) to sink(s) would be generated. Such an approach would require substantial effort due to the large number of paths. The second approach is a local method whereby the network is modeled length-by-length and header-by-header using a simple digraph algorithm to characterize the switching behavior of the headers.

Consider the network shown in Fig. A-30a. Flow must pass from the RWST to pumps P1, P2, and P3. Crosstie valve V3 is normally closed so, under normal conditions, with valve V1 failed closed, P1 and P2 would not have an open path from RWST. These injection paths would fail. However, allowing the operator to open V3 changes the outcome since flow through V2 could supply all three pumps (provided that the piping has been sized to allow for this contingency). The digraph for this network is shown in Fig. A-30b. The network is considered to consist of pipe headers and the connections between them. The digraph is constructed a header at a time without the need to consider global path searches. The algorithm used is as follows:

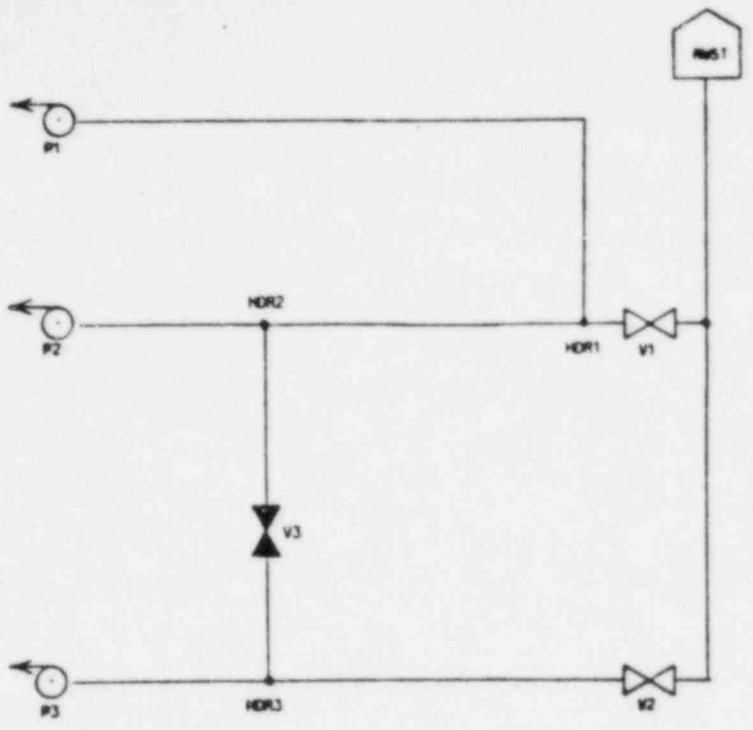


Figure A-30a. Crosstie Network

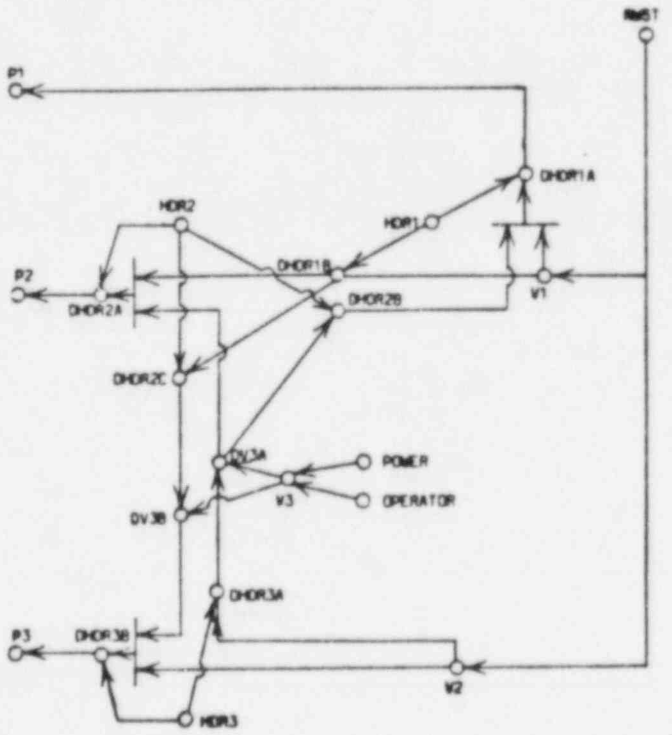


Figure A-30b. Digraph of Crosstie Network.

At each header, flow can exit through each of the pipes which form the junction (unless a check valve or pump constrains fluid from flowing away from the header in a given pipe). Considering each exit independently, the possible sources of flow to it are AND-ed together and input to a dummy node. The sources are nodes adjacent to the header. The node representing the header is OR-ed into this dummy node and represents the necessity of an open path through the header to enable flow through the exit path being considered. This process is repeated for each output from the header, and the entire scheme is repeated at each header through out the network.

An example of this modeling is shown by the model for flow through header HDR2. This header is the junction of three pipes. Fluid can flow away from the header through any of the three paths. These paths are considered one at a time, in any order, and the status (success or failure) of each path is embodied by a dummy node. Dummy nodes in this example all begin with D. Node DHDR2A is the status node for flow away from HDR2 and toward P2. Flow to it can come from either of the two other entrances to the header, so these two flow paths are AND-ed together and input to the status node. One of these flow paths originates at DHDR1B and the other at DV3A. The first is the flow away from HDR1 in the pipe connecting it to HDR2. Inputs to DHDR1B will not be developed until modeling has progressed to HDR1. Node DV3A is flow away valve V3 in the direction toward HDR2. As before, Node HDR2 inputs to DHDR2A since integrity of a path through the header is needed with either of the two flow paths for the flow to reach out of the header toward P2. Once this simple analysis has been applied to the other branches out of HDR2 and to the other components through which flow can pass in more than one direction, the digraph is complete.

A.4.3 Tripleton Code

The DMA model provides a rich basis for investigation of the effect of various component or system failures. For example, the effect of the loss of offsite power could be investigated by "turning on" the node which represents loss of offsite power. A special version of the reachability code was designed and is used to allow these investigations. This code functions in a manner similar to the reachability code described earlier with one difference. After the single dependency reachability calculation is performed, a double dependency reachability calculation is performed using the node to be "turned on" (a given component failure is set to TRUE) as an initial condition to the calculation. The reachability result of this calculation is then used as a result of the single dependency calculation for all subsequent (Doubleton) calculations. The resulting Doubletons are Tripletons with the turned on node.

This technique allows a simulation study of the impact of specific failures on the system. For example, the loss of onsite power could be investigated by creating a master node for onsite power and making it adjacent to all onsite power sources. This master node would then be turned on for a Tripleton run.

A.4.4 Conditioned Cycle

A conditioned cycle occurs when a model contains an AND gate whose output cycles back to become one of its inputs. Figure A-31 shows a graphic example of a simplified digraph with a conditioned cycle.

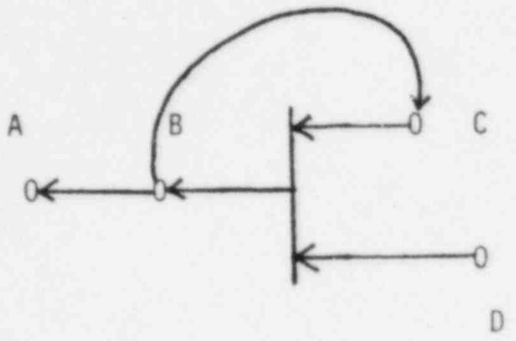


Figure A-31. Conditioned Cycle Structure

The networks can contain any kind of adjacency, including other conditioned cycles, and can have inputs and outputs to and from the rest of the model. In general, though, conditioned cycles contain an AND-gate whose output is upstream of either or both of its inputs. They are generated primarily by the implementation of the junction model for bidirectional crossties and by limitations in modeling timing sequences. Should an ultimate source (such as the RWST) provide an input to a conditioned cycle AND gate, its failure will not propagate to the output of the gate since the output feeds back to itself through the gate. This problem can be corrected by identifying where the RWST feeds into the conditioned cycle and shorting across the gate.

The conditioned cycle problem consists of two parts:

1) Identification of conditioned cycles that need to be broken, and
2) Adding the appropriate short(s).

o <u>Identification of Conditioned Cycles that Need to be Broken</u>

All conditioned cycles can be quickly found for any size problem by first changing all AND-gates into OR-gates and then processing the resulting model for reachability. For each deconditioned AND-gate, the reachability results are scanned to see if the AND-gate output reaches either of its inputs. Such a reach indicates the presence of a conditioned cycle. For each such pair of AND-gate output/input reach nodes there is a path connecting them. This path can be found using a digraph path finding code. The resultant path is the conditioned cycle.

For all of the conditioned cycles found, only those which block the propagation of source failures need to be broken. Therefore, the only conditioned cycles which need shorts added are those in which the ultimate source reaches "into but not out of" the conditioned cycle. Components downstream of the source are also implicitly taken into account by this procedure. A single dependence reachability calculation on the fully conditioned model can be used to ascertain which of the conditioned cycles found need shorting.

A.4.5 NOT Gates and Non-Coherent Models

DMA allows complement events in the diagraph. The Disjunctive Normal Form solution that the DMA codes find can be converted to the set of Prime Implements that capture Concensus Law contributions using the following theorems:

THEOREM 1: Let Φ be a biform Boolean expression in disjuction normal form, such that $\Phi = \psi \vee \phi$ without loss of generality, where ϕ is monoform, and $\ell(\phi) \wedge \ell(\psi)$ = Null Set (no literal or its complement in common). Then $\Phi'' = \psi'' \vee \phi$ gives the complete set of prime implicants.

Proof (by contradiction):

Assume $\Phi'' = \psi'' \vee \phi$ is not the complete set of prime implicants

then $\Phi'' = \psi'' \vee \phi \vee A$ where A is a prime implicant.

Then

$$\Phi'' = (\psi \vee \phi)'' = (\psi' \wedge \phi')' \qquad (2.1)$$

$$\Phi'' = (\psi'' \vee \phi) \vee A \qquad (2.2)$$

which implies either,

$$\phi'' = \phi \vee A \text{ by Concensus Law} \qquad (2.3)$$

or

$$(\psi' \wedge \phi')' = \psi'' \vee A \vee \phi \qquad (2.4)$$

but equation (2.3) is contrary to the requirement that ϕ is monoform and equation (2.4) is contrary to the requirement than $\ell(\phi) \wedge \ell(\psi)$ = Null Set. Therefore by contradiction, $\Phi'' = \psi'' \wedge \phi$.

The next theorem generalizes theorem 1.

THEOREM 2: Let Φ be a biform Boolean expression in disjunctive normal form, such that $\Phi = \psi \vee \phi$ (without loss of generality) where ϕ is monoform and $\phi = \{ \chi \mid \chi$ is any literal such that its complement does not appear in $\Phi \}$.

Then $\Phi'' = \psi'' \vee \phi$ gives the complete set of prime implicants after applying only Boolean absorption laws.

Proof (by contradiction):

Assume $\Phi'' = \psi'' \vee \phi$ will not yield the complete set of prime implicants after applying only Boolean absorption laws, then $\underline{\exists}$ a prime implicant A such that

$$(\Phi)'' = (\psi \vee \phi)'' = (\psi' \wedge \phi')' = \psi'' \vee \phi \vee A \qquad (3.1)$$

This implies either,

$$\phi'' = \phi \vee A \qquad (3.2)$$

after Boolean absorption; or

$$(\psi' \vee \phi')' = \psi'' \vee \phi \vee A \qquad (3.3)$$

after Boolean absorption; but equation (3.2) is contrary to the requirement that ϕ is monoform, and equation (3.3) is contrary to the requirement that ψ does not have the complement of any literal in ϕ.

A.4.6 Methods of Reducing Problem Size

To facilitate solution of the model, two independent techniques were used to reduce its size without loss dependency information. Their application is in addition to reduction by Boolean condensation as described in Section A.2.4. The automated techniques are called pruning and partitioning. Pruning is the process by which parts of the model which cannot flow to the terminal node are discarded (eliminates independent subgraphs that are not relevant). Partitioning is used to replace selected pieces of the model with simpler, smaller "equivalent circuits" which behave like the original. Together, the two techniques can reduce global models by up to 50% beyond that due to application of Boolean condensation alone.

Pruning

Consider the simple model in Figure A.32. The model contains 15 nodes.

The subgraph relevant to the performance of any given node, however, contains fewer than 15 nodes. For instance, performance of node F is not dependent on node P because no path is possible from P to F. Thus, P could be discarded from the model without affecting global dependency of F. In general, all nodes not upstream of F could be discarded since they cannot possible affect F. This would leave the model in Figure A.33 which contains only 11 nodes.

Every node retained in the pruned model in turn retains all information about what it is dependent on. That is because, if a node, i.e., D, is upstream of F then every node upstream of it is also upstream of F and is therefore kept. The original adjacency is pruned by standing on the designated terminal node and then "walking" upstream. After the whole model has been swept in this fashion, the list of nodes encountered on the walk is then used to modify the original model. Any line of adjacency input which contains a node which was not encountered on the walk is discarded since it cannot be conduit to the terminal node. The resultant adjacency input is pruned and can be processed like a normal adjacency file.

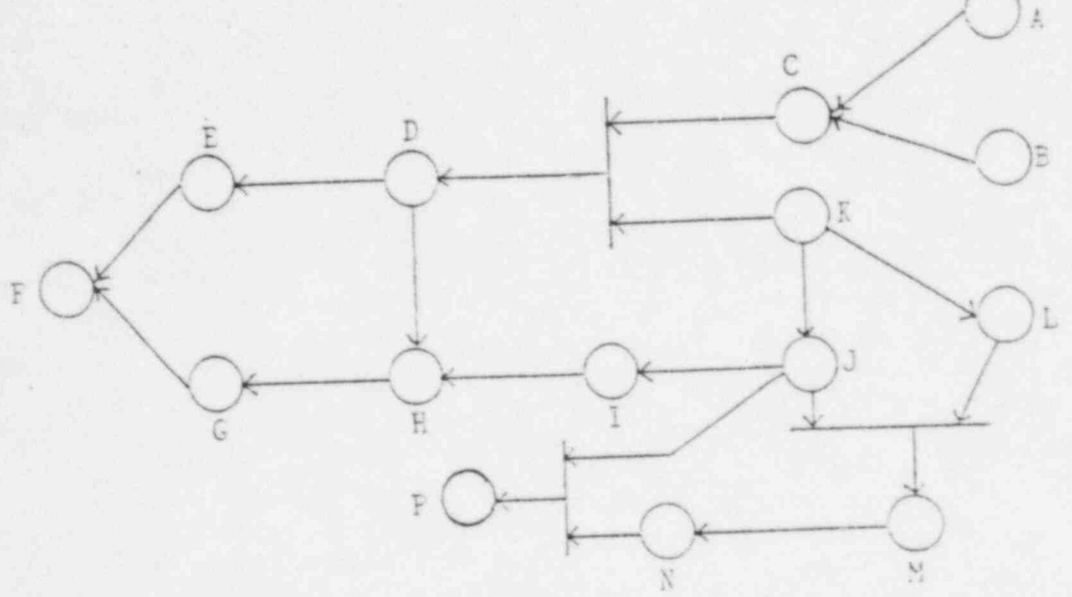


Figure A-32

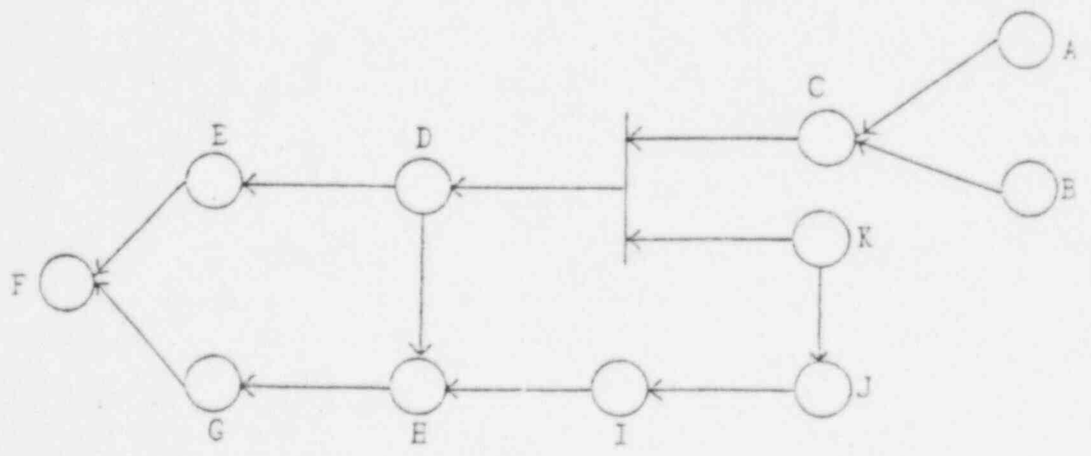


Figure A-33

REFERENCES FOR APPENDIX A

A-1. J. L. Peterson, "Petri Nets," Computing Surveys, Volume 9, Number 3, September 1977.

A-2. S. Warshall, "A Theorem on Boolean Matrices," Journal of the Association for Computing Machinery, Volume 9, 11-12, 1962.

A-3. H. S. Warren, "A Modification of Warshall's Algorithm for the Transitive Closure of Binary Relations," Comm. ACM, Volume 18, 218-220, 1975.

A-4. I. J. Sacks, "Techniques for the Determination of Potential Adversary Sources with Tampering (Level 4.1)," Lawrence Livermore National Laboratory, October 1978.

A-5. M. F. Chamow, "Directed Graph Technique for the Analysis of Fault Trees," IEEE Transactions on Reliability, R-27, No. 1, April 1978.

A-6. P. A. Renard, "Clamor", Lawrence Livermore National Laboratory, MC-79-96, January 1979.

A-7. C. J. Patenaude, D. W. Freeman, "Tampering Analysis in the Structured Assessment Approach - A Description of the Level 4 Capability," Lawrence Livermore National Laboratory, November 1980.

A-8. M. N. S. Swamy, K. Thulasiraman, "Graphs, Networks, and Algorithms," John Wiley & Sons, 1981.

A-9. I. J. Sacks, "Digraph Matrix Analysis," IEEE Transactions on Reliability (to appear).

A-10. H. P. Alesso, I. J. Sacks and C. F. Smith, Initial Guidance on Digraph-Matrix Analysis for Systems Interaction Studies," NUREG ICR-2915 1983.

A-11. G. C. Corynen, "Evoluating the Response of Complex Systems to Environmental Threats: The $\Sigma\pi$ Method," UCRL 53399, Lawrence Livermore National Laboratory, May 1985.