NASA Contractor Report 158992

MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS

Semi-Annual Status Report No. 3

(NASA-CR-158992) MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS Semiannual Status Report, 1 May - 31 Oct. 1977 (Michigan Univ.) 177 p HC A09/MF A01 CSCL 09B G3/60 N79-17563 Unclas 43726

John F. Meyer

THE UNIVERSITY OF MICHIGAN
Ann Arbor, Michigan 48109

NASA Grant NSG-1306
January 1978

FEB 1979
RECEIVED

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

SEMI-ANNUAL STATUS REPORT NO. 3
COVERING THE PERIOD FROM 1 MAY 1977 TO 31 OCTOBER 1977

MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS

PRINCIPAL INVESTIGATOR
JOHN F. MEYER

PREPARED FOR
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LANGLEY RESEARCH CENTER
HAMPTON, VIRGINIA 23365

NASA GRANT NSG 1306

JANUARY 1978

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
SEL REPORT NUMBER 116
THE UNIVERSITY OF MICHIGAN, ANN ARBOR

TABLE OF CONTENTS

1. INTRODUCTION 1

2. DELETED 4

3. TECHNICAL STATUS 5

3.1 System Models 6

3.1.1 The Model Hierarchy 6

3.2 Performability Evaluation 11

3.2.1 Performability 11

3.2.2 Capability Functions 15

3.3 Capability and Functional Dependence 21

3.3.1 Capability and the Model Hierarchy . . .21

3.3.2 Functional Dependence 27

3.3.2.1 Basic Definitions 28

3.3.2.2 R-Dependence 32

3.3.2.3 Conditional R-Dependence . . .43

3.4 Computation of Trajectory Set Probabilities . .46

3.4.1 Phased Models 46

3.4.2 Performability Evaluation of Phased Models 49

3.4.3 Simplification of Phased Models 52

3.5 Hierarchical Modeling of an Air Transport Mission 73

3.5.1 Notational Conventions 74

3.5.2 Mission Description and Accomplishment Set 81

3.5.3 Higher Level Models 82

3.5.3.1 Mission Level Model Development 82

3.5.3.2 Aircraft Functional Task Level Model Development. . 85

3.5.3.3 Computational Task Level Model Development 90

3.5.4 A Calculus of Trajectory Sets 93

3.5.4.1 An Algorithm for Determining γ^{-1} 94

3.5.4.2 Trajectory Sets, Array Products and Intersections . 96

3.5.5 Higher Level Partial Capability Function Preimage Sets 104

3.5.5.1 γ_0-Induced Trajectory Sets . . 104

3.5.5.2 γ_1-Induced Trajectory Sets . . 104

3.5.5.3 γ_2-Induced Trajectory Sets . . 111

3.5.6 Bottom Level Models 116

3.5.6.1 Dedicated Processor Model . . 118

3.5.6.2 Dedicated Group Processor Model 122

3.5.6.3 Gracefully Degrading Processor Model 127

3.5.7 Capability Function Preimage Sets . . . 133

3.5.7.1 γ^{-1} for the Dedicated Processor 137

3.5.7.2 γ^{-1} for the Dedicated Group Processor 141

3.5.7.3 γ^{-1} for the Gracefully Degrading Model 142

3.5.8 Performability Evaluation 146

3.5.8.1 METAPHOR 149

3.5.8.2 Performability Results. . 151

4. REFERENCES 171

1. INTRODUCTION

This report is the third Semi-Annual Status Report on the research project "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems" being conducted for the NASA Langley Research Center under NASA Grant 1306. The subject grant was initiated 1 May 1976 for a one year period and extended 1 May 1977 for a second one year period. This report concerns work accomplished during the first half of the second year, that is, the period from 1 May 1977 to 31 October 1977, hereafter referred to as the reporting period.

The purpose of this research project is to develop models, measures and techniques for evaluating the effectiveness of aircraft computing systems. By "effectiveness" in this context we mean the extent to which the user, i.e., a commercial air carrier, may expect to benefit from the computational tasks accomplished by a computing system in the environment of an advanced commercial aircraft. Thus the concept of effectiveness involves aspects of system performance, reliability and worth (value, benefit) which must be appropriately integrated in the process of evaluating system effectiveness. More specifically, the primary objectives of this project are:

1) The development of system models that can provide a basis for the formulation and evaluation of aircraft computer system effectiveness,

2) The formulation of quantitative measures of system effectiveness, and

3) The development of analytic and simulation techniques for evaluating the effectiveness of a proposed or existing aircraft computer.

Work accomplished during the first year [1], [2] was concerned primarily with objectives 1) and 2). Midway through the first year, a decision was made to decouple the performance and reliability aspects of effectiveness from the worth aspect, and to focus the effort on performance and reliability issues. As argued when this research was originally proposed, and as further substantiated by work accomplished to date, the issues of performance and reliability must be dealt with simultaneously in the process of evaluating system effectiveness. The term "performability" was introduced to refer to this unification of performance and reliability, and performability was identified with effectiveness in the above stated objectives.

During the current reporting period, work has been performed in connection with objective 3) as well as objectives 1) and 2). More specifically, this work has concerned:

1) Further formal development of the general modeling framework that serves as the basis for performability evaluation, including more precise definitions of "base model" and "system performance" which permit a general definition of "performability" relative to any discrete-valued performance variable,

2) Formal justification of the performability concept and identification of conditions under which performance and reliability can be treated independently,

3) Further development of the general concept of "capability" and verification of the fact that capability functions, in their ability to relate state behavior to system performance, are indeed more general than "structure functions" or equivalently, the representation of structure functions by "fault-trees",

4) Formulation of the capability function in terms of a model hierarchy and its associated "inter-level translations",

5) Further investigation of the "functional dependence" inherent in a capability function, including proofs of fundamental properties,

6) The use of time "phasing" and state "lumping" to simplify the evaluation of performability, in particular, the establishment of conditions under which different phases can employ different lumpings (refered to informally as "Michigan lumping"),

7) More detailed development of analytical methods for determining the trajectory set $\gamma^{-1}(a)$ of an accomplishment level a, including methods of representing trajectory sets at various levels of the model hierarchy and methods of computing the trajectory set $\gamma_{i+1}(a)$ at level i+1, given the trajectory set $\gamma_i(a)$ at level i, and

8) Application of the above theory and methodology to specific examples, including a comprehensive example illustrating the modeling and subsequent performability evaluation of a computer in the environment of a portal-to-portal air transport mission.

We believe that the following report attests to a substantial amount of progress in each of the above areas. Moreover, we feel that the progress to date represents the greater part of the total effort proposed for the second year of the project [3].

Section 2 of the report describes the manpower effort proposed for the current year, the personnel involved in conducting the investigation, and their levels of effort during the reporting period. Section 3, the body of the report, describes the technical status of the research performed during the reporting period.

DELETED

3. TECHNICAL STATUS

The following is a comprehensive description of the research performed during the reporting period. The report is divided into five subsections under the headings:

3.1 System Models,
3.2 Performability Evaluation,
3.3 Capability and Functional Dependence,
3.4 Computation of Trajectory Set Probabilities, and
3.5 Hierarchical Modeling of an Air Transport Mission.

Relative to the eight topics listed in the introduction, Subsection 3.1 reports on further work concerning the formalization of our general modeling framework (topic 1). Subsection 3.2 gives precise definitions of "performability" and "capability" and, in terms of these definitions, describes results which formally justify a unified performance-reliability approach to system evaluation (topics 2 and 3). Subsection 3.3 reports on our work concerning hierarchical formulation of the capability function (topic 4) and on the general concept of functional dependence (topic 5). Subsection 3.4 reports on research concerning model simplification for the purpose of computing trajectory set probabilities, in particular, the use of time "phasing" and state "lumping" (topic 6). Finally, Section 3.5 discusses the problem of trajectory set determination and develops a detailed example of the modeling and evaluation of an air transport mission (topics 7 and 8).

The numbering of definitions, theorems, and supporting results begins anew in each of these major subsections. Reference numbers in the margin carry the prefix of the major subsection, e.g., the first item referenced in subsection 3.1 is numbered 3.1.1.

3.1. System Models

During the reporting period, we have developed a probability-theoretic basis for the modeling framework discussed in the previous Semi-Annual Status Reports [1,2]. This formal representation permits us to rigorously restate various intuitive concepts and assumptions associated with models of the total system. It also provides us with a more precise foundation for the investigation of model simplification techniques such as time "phasing" and state "lumping."

3.1.1. The Model Hierarchy

As noted in the previous reports (see [2], Section 3.1.1, pp. 7-8), the total system may be viewed at several levels. At a lower level, there is a detailed view of how various components of the computer's hardware and software structure behave throughout the utilization period. At this level there is also detailed view of the behavior of the computer's "environment," where by this term we mean both man-made components (user input, peripheral subsystems, etc.) and natural components (radiation, weather, etc.) which can influence the computer's effectiveness. A second view of the total system is the user's view of how the system behaves during utilization, that is, what the system accomplishes for the user during the utilization period. A third, even higher level view, is the computing system's "worth" (as measured, say, in dollars) when operated in its use environment.

To formalize these views, we postulate the existence of a <u>probability space</u> (Ω, E, P) that underlies the total system,

where Ω is the (sample) <u>description space</u>, $\mathcal{E}$ is a set of (measurable) <u>events</u> and $P: \mathcal{E} \to [0,1]$ is the <u>probability measure</u> (see [4], for example). This probability space represents all that needs to be known about the total system in order to describe the probabilistic nature of its behavior at the various levels described above. It thus provides a hypothetical basis for defining higher level models. In general, however, it will neither be possible nor desirable to completely specify Ω, $\mathcal{E}$ and P.

In the discussion that follows, let S denote the total system, where S is comprised of a computing system C and its environment E. At the most detailed level, the behavior of S is formally viewed as a stochastic process

$$X_S = \{X_t | t \in T\}$$

where

T = a set of real numbers (observation times) called the <u>utilization period</u>

and, for all $t \in T$, X_t is a random variable

$$X_t: \Omega \to Q$$

defined on the underlying description space and taking values in the <u>state space</u> Q of the total system. Depending on the application, the utilization period T may be discrete (countable) or continuous and, in cases where one is interested in long-run behavior, it may be unbounded (e.g., $T = \mathbf{R}_+ = [0,\infty)$). The state space Q embodies the state sets of both the computer and its environment, i.e.,

$$Q = Q_C \times Q_E$$

where Q_C and Q_E can, in turn, be decomposed to represent the local state sets of computer and environmental subsystems. For our purposes, it suffices to assume that Q is countable (finite or countably infinite) and, hence, for all $t \in T$ and $q \in Q$, "$X_t = q$" has a probability (i.e., $\{\omega | X_t(\omega) = q\} \in E$). The random process X_S is referred to as the <u>base model</u> of S. An instance of the base model's behavior is a <u>state trajectory</u>

$$u_\omega : T \to Q \qquad (\omega \in \Omega) \qquad (3.1.1)$$

where

$$u_\omega(t) = X_t(\omega). \qquad (t \in T)$$

Thus, corresponding to an underlying outcome $\omega \in \Omega$, u_ω describes how the state of the total system changes as a function of time throughout the utilization period T. Accordingly, the "description space" for the base model is the set

$$U = \{u_\omega | \omega \in \Omega\}$$

which is referred to as the (state) <u>trajectory space</u> of S.

It is worth noting at this point that, for even moderately complex systems, the base model may be so large that practical methods of formulation or even simulation are precluded. In such cases, one must seek simplifications of the base model which nevertheless remain detailed enough to support the user's view of total system behavior. Accordingly, the question of base model simplification will be considered after the complete modeling framework has been described.

In terms of the underlying probability space (Ω, E, P) the user's view of the system is formalized as follows. We assume that the user is interested in distinguishing a number of

different levels of accomplishment when judging how well the system has performed throughout the utilization period. (One such level may be total system failure.) The user's "description space" is thus identified with an accomplishment set A whose elements are referred to alternatively as accomplishment levels or (user-visible) performance levels. A may be finite, countably infinite, or uncountable (in the last case, A is assumed to be a subset of real numbers). Thus, for example, the accomplishment set associated with a nondegradable system is

$$A = \{a_0, a_1\}$$

where

$$a_0 = \text{"system success"}$$
$$a_1 = \text{"system failure."}$$

In their modeling of the PRIME system, Borgerson and Freitas viewed the accomplishment set as the set

$$A = \{a_0, a_1, a_2, \ldots\}$$

where a_k = "k crashes during the utilization period T." If the user is primarily concerned with system "throughput" a continuous accomplishment set might be appropriate, i.e., $A = \mathbb{R}_+$ where an element $a \in A$ is the "average throughput over the utilization period T."

In terms of the accomplishment set, system performance is formally viewed as a random variable

$$Y_S : \Omega \to A$$

where $Y_S(\omega)$ is the accomplishment level corresponding to

outcome ω in the underlying description space. Similarly, assuming that the economic gain (or loss) derived from using the system is represented by a real number r (interpreted, say, as r dollars), <u>system worth</u> is a random variable defined as

$$W_S : \Omega \to \mathbf{R} \quad \text{(the set of all real numbers)}$$

where $W_S(\omega)$ is the worth associated with outcome ω. The terminology and notation defined above is summarized below.

Model	Description Space
Base model X_S	Trajectory space U
System performance Y_S	Accomplishment set A
System worth W_S	The real numbers $\mathbf{R}$

3.2 Performability Evaluation

As discussed in the first Semi-Annual Status Report ([1]; Section 3.3.4.2) and, subsequently, in the proposal for the second year ([3]; p. 14), our research effort has focused on the problem of formulating and evaluating the probabilities of accomplishing various types and qualities of missions. This problem was referred to informally as "performability evaluation" to distinguish it from the more general problem of "effectiveness evaluation." During the reporting period, we have established a more precise meaning for the concept of performability so as to further justify our claims that i) performability is a component of effectiveness, and ii) performability evaluation cannot in general be accomplished via independent evaluations of performance and reliability.

3.2.1. Performability

In terms of the general modeling framework discussed in Section 3.1, a natural measure that quantifies both system performance and reliability (ability to perform) is the probability function of the performance variable Y_S. Accordingly, we have identified the concept of performability with this measure, that is:

Definition 1: If S is a total system and A is the accomplishment set associated with system performance Y_S, then the performability of S is the probability function $p_S : A \to [0,1]$ where $p_S(a)$ = the probability that S performs at level a, that is, $p_S(a) = P(\{\omega | Y_S(\omega)=a\})$.

The above definition presumes that the performance variable Y_S is discrete (i.e., there are countably many accomplishment levels) although a similar approach, using probability density functions, can be applied to continuous performance variables. We will assume that Y_S is discrete throughout the following discussion.

Given the performability of S and assuming the existence of a worth measure (see [1], pp. 36-37), system effectiveness can be expressed as the sum

$$\mathrm{Eff}(S) = \sum_{a \varepsilon A} w(a)\, p_S(a) \qquad (3.2.1)$$

where w is a worth measure

$$w : A \to \mathbb{R}$$

such that, for all $\omega \varepsilon \Omega$,

$$W_S(\omega) = w(Y_S(\omega)).$$

(If $a \varepsilon A$, $w(a)$ is interpreted as the "worth of performance level a.") Equation 3.2.1 generalizes a relationship noted in our original proposal and shows that performability is an important component of effectiveness.

To further justify this concept, we note that traditional evaluations of computer performance and computer reliability are concerned with special types of performability. Performance evaluation is concerned with evaluating p_S under the assumption that the computer part of S is fixed (i.e., its structure does not change as the consequence of internal faults). Reliability evaluation is concerned with evaluating $p_S(B) = \sum_{a \varepsilon B} p_S(a)$ where B is a designated subset of accomplishment levels associated

with system "success." In these terms, a performability evaluation can alternatively be regarded as $|A|$ reliability evaluations, one for each singleton success set $B = \{a\}$ and, if A is finite, the evaluation may actually be carried out in this manner. As this process is generally more complex than a typical reliability evaluation procedure (in particular, it involves distinguishing all the performance levels as well as determining their probabilities), we reserve the term "reliability evaluation" to mean the evaluation of "probability of success" for some specified success criterion B. Thus performability reduces to reliability only when S is nondegradable, i.e., $|A| = 2$. Due to the special nature of both performance and reliability evaluations, we find that a direct combination of the two is generally unable to support an evaluation of system effectiveness. A case where independent evaluations do suffice (and hence the more general concept of performability is not really needed) is the following.

Let S be a system with accomplishment set A and suppose that successful performance of S can be associated with the performability of some fault-free reference system $\overline{S}$ (i.e., the computer part of $\overline{S}$ is fault-free). More precisely, this says that for some designated subset B of the accomplishment set A the <u>conditional performability of S given B</u>, i.e., the function $p_{S,B}: A \to [0,1]$ where

$$p_{S,B}(a) = \frac{P(Y_S = a \text{ and } Y_S \varepsilon B)}{P(Y_S \varepsilon B)}^{*} \qquad (3.2.2)$$

---

* As is standard practice, our notation here omits explicit reference to the underlying space, e.g., "$Y_S = a$ and $Y_S \varepsilon B$" means the set $\{\omega | Y_S(w) = a \text{ and } Y_S(\omega) \varepsilon B\}$.

is equal to the performability of $\overline{S}$, i.e.,

$$p_{\overline{S}}(a) = p_{S,B}(a), \text{ for all } a\varepsilon A. \qquad (3.2.3)$$

(Note that $p_{S,B}(a) = 0$ if $a \notin B$ and thus the accomplishment set of $\overline{S}$ need only include B.) Assuming further that accomplishment levels outside of B are of no worth to the user, i.e.,

$$w(a) = 0 \text{ if } a \notin B$$

then, by equation 3.2.1 ,

$$\mathrm{Eff}(S) = \sum_{a\varepsilon A} w(a)p_S(a) = \sum_{a\varepsilon B} w(a)p_S(a).$$

But $a\varepsilon B$ implies $p_S(a) = P(Y_S = a \text{ and } Y_S \varepsilon B)$. Hence, by the definition of conditional performability (equation 3.2.2),

$$\mathrm{Eff}(S) = \sum_{a\varepsilon B} w(a)p_{S,B}(a)p_S(B)$$

where $p_S(B) = P(Y_S \varepsilon B)$. By assumption 3.2.3 we conclude that

$$\mathrm{Eff}(S) = \sum_{a\varepsilon B} w(a)p_{\overline{S}}(a)p_S(B) = \left(\sum_{a\varepsilon B} w(a)p_{\overline{S}}(a) \right) p_S(B)$$

or equivalently,

$$\mathrm{Eff}(S) = \mathrm{Eff}(\overline{S})p_S(B). \qquad (3.2.4)$$

Accordingly, a performance-worth (effectiveness) evaluation of the fault-free reference system $\overline{S}$, along with a reliability evaluation of S, suffice to determine the effectiveness of S. Alternatively, equation 3.2.4 can be regarded as expressing the effectiveness of S relative to two levels of accomplishment, B (success) and A-B (failure), and a worth function w where $w(B) = \mathrm{Eff}(\overline{S})$ and $w(A-B) = 0$. Then, by equation 3.2.1,

$$\begin{aligned} \mathrm{Eff}(S) &= w(B)p_S(B) + w(A-B)p_S(A-B) \\ &= w(B)p_S(B) \\ &= \mathrm{Eff}(\overline{S})p_S(B). \end{aligned}$$

The secret here, of course, is to find a system $\overline{S}$ that satisfies assumption 3.2.3 . $\overline{S}$ is easily identified only when S = (C,E)* is such that faults of C which are tolerated (i.e., $Y_S \varepsilon B$) cause no change in the performance of S. In this case, $\overline{S} = (\overline{C},E)$ can serve as the reference system where $\overline{C}$ is the fault-free version of C. In particular, this is the case for fault-tolerant computer architectures which employ standby sparing [5], N modular redundancy, or combinations thereof. On the other hand, if tolerated faults can alter the performance of S, the discovery of $\overline{S}$ requires an evaluation of conditional performability (see 3.2.3) which is tantamount to evaluating the performability of S.

3.2.2 Capability Functions

A critical first step in the evaluation of performability is to establish a relationship between the base model X_S and the user-oriented performance model Y_S (see Section 3.1.1). To accomplish this, we assume that the base model is refined enough to distinguish the levels of accomplishment perceived by the user, that is, for all $\omega,\omega' \varepsilon \Omega$,

$$Y_S(\omega) \neq Y_S(\omega') \text{ implies } u_\omega \neq u_{\omega'}, \qquad (3.2.5)$$

where u_ω and $u_{\omega'}$ are the state trajectories associated with outcomes ω and ω' (see equation 3.1.1). This implies that each trajectory $u \varepsilon U$ is related to a unique accomplishment level $a \varepsilon A$. Accordingly, the concept of capability (introduced during the previous reporting period; see [2], [3]) can be more precisely defined as follows:

---

* C is the computer part of S; E is the environment.

Definition 2: If S is a system with trajectory space U and accomplishment set A then the capability function of S is the function $\gamma_S : U \to A$ where $\gamma_S(u)$ is the level of accomplishment resulting from state trajectory u, that is,

$$\gamma_S(u) = a \text{ if, for some } \omega \in \Omega,\ u_\omega = u \text{ and } Y_S(\omega) = a.$$

By condition 3.2.5 it follows that γ_S is well-defined and, when there is no chance for ambiguity, γ_S will be written simply as γ.

Given the capability function of S, the performability p_S can be expressed in terms of the base model X_S. Let $\mathcal{V}$ denote the collection of measurable trajectory sets, i.e., $V \in \mathcal{V}$ if and only if there is an event $E \in \mathcal{E}$ such that $V = \{u_\omega | \omega \in E\}$. Let $Pr : \mathcal{V} \to [0,1]$ denote the probability measure of the base model where $Pr(V) = P(E)$ if V corresponds to the underlying event E. In practice, of course, the measure Pr is derived from known properties of the base model (e.g., X_S is Markovian) rather than from the underlying probability space. If a is an accomplishment level then, by the definition of performability (Def. 1),

$$\begin{aligned} p_S(a) &= P(\{\omega | Y_S(\omega) = a\}) \\ &= Pr(\{u_\omega | Y_S(\omega) = a\}) \end{aligned}$$

and hence, by the definition of capability (Def. 2)

$$\begin{aligned} p_S(a) &= Pr(\{u | \gamma_S(u) = a\}) \\ &= Pr(\gamma_S^{-1}(a)). \end{aligned} \qquad (3.2.6)$$

The preimage $\gamma_S^{-1}(a)$ is referred to as the trajectory set of a and its determination requires an analysis of how an accomplishment level $a \in A$ relates back down via γ_S^{-1} to trajectories of the

base model. $p_S(a)$ is then determined by a probability analysis of $\gamma_S^{-1}(a)$. Methods of implementing this process are discussed in Section 4.

The role of a capability function in performability evaluation is similar to that of a "structure function" in reliability evaluation. However, even when performability is restricted to reliability, the concept of a capability function is more general. The special class which corresponds to structure functions may be characterized as follows. Let S be a system where Q is the state space of the base model and $A = \{0,1\}$ is the accomplishment set (where 1 denotes "success" and 0 denotes "failure"). Then the capability function γ is <u>structure-based</u> if there exists a <u>structure function</u>* $\varphi: Q \to \{0,1\}$ such that, for all $u \in U$,

$$\gamma(u) = 1 \text{ iff } \varphi(u(t))=1, \text{ for all } t \in T.$$

Thus, when capability is structure-based, a local (in time) success criterion can be applied to "snapshots" of u throughout T to determine whether u results in system success. Alternatively, this criterion can be specified as membership in a prescribed set of "success states" R where

$$R = \varphi^{-1}(1) = \{q \mid \varphi(q) = 1\}.$$

When system success is viewed in structural terms, as in the case in most reliability studies, a structure-based capability function will often suffice. On the other hand, when success relates to system behavior (e.g., when reliability

* The usual definition (see [6], for example) requires that $Q = \{0,1\}^n$ where the ith coordinate of $q \in Q$ is interpreted as the operational state of the ith component of the system.

is "computation-based" [7]), we find that success is generally not definable in terms of a local success criterion such as φ or R. The following example serves to demonstrate this fact.

Example 1

Let S = (C,E) where C represents a distributed computer comprised of n subsystems, and E represents the computer's workload. Suppose further that system "throughput" (i.e., the user-visible work rate of C given E) varies as a function of the number of fault-free subsystems. Assuming a constant workload E, the operational states of S can be represented by the state space

$$Q = \{0,1,\ldots,n\}$$

where state i corresponds to "i fault-free subsystems." The variation in throughput is described by a function

$$\tau : Q \to \mathbf{R}_+$$

where $\tau(i)$ = throughput of S in state i.

Assuming S is used continuously throughout a utilization period T = [0,T], the base model of S is a stochastic process

$$X_S = \{X_t \mid t \varepsilon [0,T]\}$$

where each X_t is a random variable taking values in Q. (The probabilistic nature of X_S is not an issue here.) As for performance, suppose that the user is interested in the average throughput of the system, where the average is taken over the utlization period T. Then, depending on the nature of the accomplishment set, the performability of S can be expressed in several different ways. For a continuum of

accomplishment levels, A can be identified with $\mathbb{R}_+ = [0,\infty)$ and the capability of S is the function $\gamma_1 : U \to \mathbb{R}_+$ where

$$\gamma_1(u) = \int_0^T \tau(u(t))dt/T.$$

From a more practical point of view, the user may be interested in only a finite number of accomplishment levels

$$A = \{0, r_1, r_2, \ldots, r_\ell\} \subset \mathbb{R}_+$$

where $0 < r_1 < \ldots < r_\ell$ and r_i represents a range of average throughputs between r_i and r_{i+1}. More precisely, the capability function in this case is the function $\gamma_2 : U \to A$ where

$$\gamma_2(u) = \begin{cases} 0 & \text{if } \gamma_1(u) \in [0, r_1) \\ r_i & \text{if } 0 < i < \ell \text{ and } \gamma_1(u) \in [r_i, r_{i+1}) \\ r_\ell & \text{if } \gamma_1(u) \in [r_\ell, \infty). \end{cases}$$

Finally, the user may be interested only in success or failure where success is identified with a minimum average throughput $\bar{\tau}$. In this case γ_S is the function $\gamma_3 : U \to \{0,1\}$ where

$$\gamma_3(u) = \begin{cases} 1 \text{ if } \int_0^T \tau(u(t))dt/T \geq \bar{\tau} \\ 0 \text{ otherwise.} \end{cases}$$

For each of the capability functions $\gamma_S = \gamma_i$ of the above example, it is obvious that the value $\gamma_S(u)$ depends on a complete knowledge of the state trajectory u, due to the inherent memory of the integration operation. In particular, when throughput is degradable (i.e., the interesting case where different states can exhibit different positive throughputs), it follows that $\gamma_S = \gamma_3$ will generally not

admit to a structure-based formulation. (A simple, two-state example will verify this.) This remains true when the concept of structure-based capability is extended to permit different structure functions to be associated with different "phases" of the utilization period (see [8], for example), i.e., there exists a decomposition of T into k disjoint time periods (phases) $T_1, T_2, \ldots, T_k$ and there exist structure functions $\varphi_1, \varphi_2, \ldots, \varphi_k$ such that

$$\gamma(u) = 1 \text{ iff } \varphi_i(u(t)) = 1, \text{ for all } i \in \{1,2,\ldots,k\} \text{ and for all } t \in T_i.$$

In general, we have found that a capability function cannot be structure-based wherever there exists an intermediate time period T' and a "success trajectory" v (i.e., $\gamma(v) = 1$) such that the knowledge that a trajectory u agrees with v during T' alters the success criterion for u during T - T'. Thus, even in the case of two accomplishment levels, the concept of a capability function (Definition 2) represents a proper extension of relations between state behavior and system performance that are typically assumed in the theory of reliability.

3.3. Capability and Functional Dependence

3.3.1. Capability and The Model Hierarchy

As discussed in Section 3.2., the performability of the total system S with accomplishment set A may be expressed as the function $p_S:A \to [0,1]$ where

$$p_S(a) = Pr(\gamma_S^{-1}(a))$$

and γ_S is the capability function. From this formulation it may be seen that one method of evaluating a particular $p_S(a)$ is to i) determine a characterization of the set $V = \gamma^{-1}(a)$ which suffices to ii) calculate the probability Pr(V). In this section we consider the problem of expressing γ^{-1} for the purpose of characterizing the sets $V = \gamma^{-1}(a)$. Section 3.3.2 presents one tool, functional dependence, for use in calculating Pr(V).

Recall that for a system S with trajectory space U and accomplishment set A the capability function is the function $\gamma_S:U \to A$ where $\gamma_S(u)$ is the level of accomplishment resulting from state trajectory u. Thus, γ_S expresses the relationship between the base model X_S and the performance model Y_S. A major problem in expressing this relationship for a given X_S and Y_S is due to the potential dissimilarities between the two models. To ease the problems of transition, a model hierarchy is introduced. The hierarchy provides for a step-by-step, top-down elaboration of each accomplishment level a, terminating in the desired base model description $\gamma^{-1}(a)$. The performance model Y_S itself sits above the level-0 (top) model of the hierarchy. Each intermediate model of the hierarchy is defined

in a manner similar to that of the base model. More precisely, if there are m+1 levels in the hierarchy, the level-i model (i=0,1,...,m) is a stochastic process X^i defined in terms of two independent processes X_c^i and X_b^i referred to as the composite and basic parts of the level-i model. The composite part inherits its behavior from the next lower (level-(i+1)) model; the basic part represents new information, external to the level-(i+1) model, that is introduced at level-i.

For instance, at a certain level in modeling the actual hardware of a system, the effects of weather may be of no consideration in developing an accurate model. Higher up in the hierarchy, however, the effects of weather on the system may have to be considered to adequately reflect the user's needs. Thus, weather would be regarded as a part of the total system whose effects may be introduced as a basic part at a higher level. After introduction, as a basic part, it supports the higher (lower numbered) level composite parts.

We express this more precisely by

$$X_c^i = \{X_{c,t}^i \mid t \varepsilon T_c^i\}$$

where $X_{c,t}^i : \Omega \to Q_c^i$ is a random variable taking values in the composite state space Q_c^i (at level-i). Q_c^i may be further coordinatized. The projection of $X_{c,t}^i$ on a particular coordinate is called a composite variable (at level-i). A composite trajectory is a function $u_{c,\omega} : T_c^i \to Q_c^i$ where $u_{c,\omega}(t) = X_{c,t}^i(\omega)$; the composite trajectory space is the set $U_c^i = \{u_{c,\omega} \mid \omega \varepsilon \Omega\}$. Similar definitions, terminology and notation apply to the basic process X_b^i. To permit extension

of either X_c^i or X_b^i to larger time-bases, a fictitious state $\not{c}$ is adjoined to each of Q_c^i and Q_b^i. Then, relative to a time-base $T_0 \supset T_c^i$, X_c^i (similar remarks apply to X_b^i) is taken to be the process

$$X_c^i = \{X_{c,t}^i | t \in T_0\}$$

where, if $t \in T_0 - T_c^i$, $X_{c,t}^i$ is defined to be the constant-valued random variable

$$X_{c,t}^i(\omega) = \not{c}, \text{ for all } \omega \in \Omega .$$

Extending both X_c^i and X_b^i to $T^i = T_c \cup T_b^i$, the <u>level-i model</u> is the stochastic process

$$X^i = \{X_t^i | t \in T^i\}$$

where $X_t^i = (X_{c,t}^i, X_{b,t}^i)$. The <u>state space of the level-i model</u> is

$$Q^i = Q_c^i \times Q_b^i$$

and its <u>trajectory space</u> U^i is represented by the set

$$U_c^i \otimes U_b^i = \{(u_{c,\omega}, u_{b,\omega}) | \omega \in \Omega\} .$$

(With a slight abuse of terminology and notation, $U_c^i \otimes U_b^i$ will be denoted as U^i and referred to as the trajectory space of X^i.) In case there are no composite (basic) variables at level-i, Q_c^i (Q_b^i) is simply deleted, that is $Q^i = Q_b^i$ ($Q^i = Q_c^i$). In these cases the corresponding trajectory space is $U^i = U_b^i$ ($U^i = U_c^i$). Combining such models, we have:

<u>Definition 3</u>: If S is a total system with base model X_S and capability function γ_S, the collection $\{X^0, X^1, \ldots, X^m\}$ of level-0 to level-m models is a <u>model hierarchy for S</u> if the

following conditions are satisfied:

i) $X^m = X_b^m$, that is, all variables of the "bottom model are basic.

ii) If each model X^i is extended to the utilization interval T, the base model X_S is the stochastic process

$$X_S = \{X_t | t \in T\}$$

where $X_t = (X_{b,t}^m, X_{b,t}^{m-1}, \ldots, X_{b,t}^0)$.

(Accordingly, the state space of X_S is $Q = Q_b^m \times Q_b^{m-1} \times \ldots \times Q_b^0$ and the trajectory space U is represented by the set $U_b^m \otimes U_b^{m-1} \otimes \ldots \otimes U_b^0$.)

iii) For each level i, there exists an <u>interlevel translation</u> κ_i where

$$\kappa^0 : U_c^0 \otimes U_b^0 \to A$$

$$\kappa_i : U_c^i \otimes U_b^i \to U_c^{i-1} \quad (1 < i < m)$$

$$\kappa_m : U_b^m \to U_c^{m-1}$$

such that the capability function γ_S can be decomposed as follows. If $u \in U$ where $u = (u_m, u_{m-1}, \ldots, u_1)$ with $u_i \in U_b^i$, then

$$\gamma_S(u) = \kappa_0(\ldots \kappa_{m-1}(\kappa_m(u_m), u_{m-1}), \ldots, u_0).$$

The terminology and notation of Definition 3 is summarized in Figure 1. It should be clear that this formal definition of a model hierarchy follows from the less rigorous approaches in [1] and [2]. Notice that while the performance model Y_S sits above the level-0 model, the base model X_S may be completely represented by the m+1 intermediate level base models.

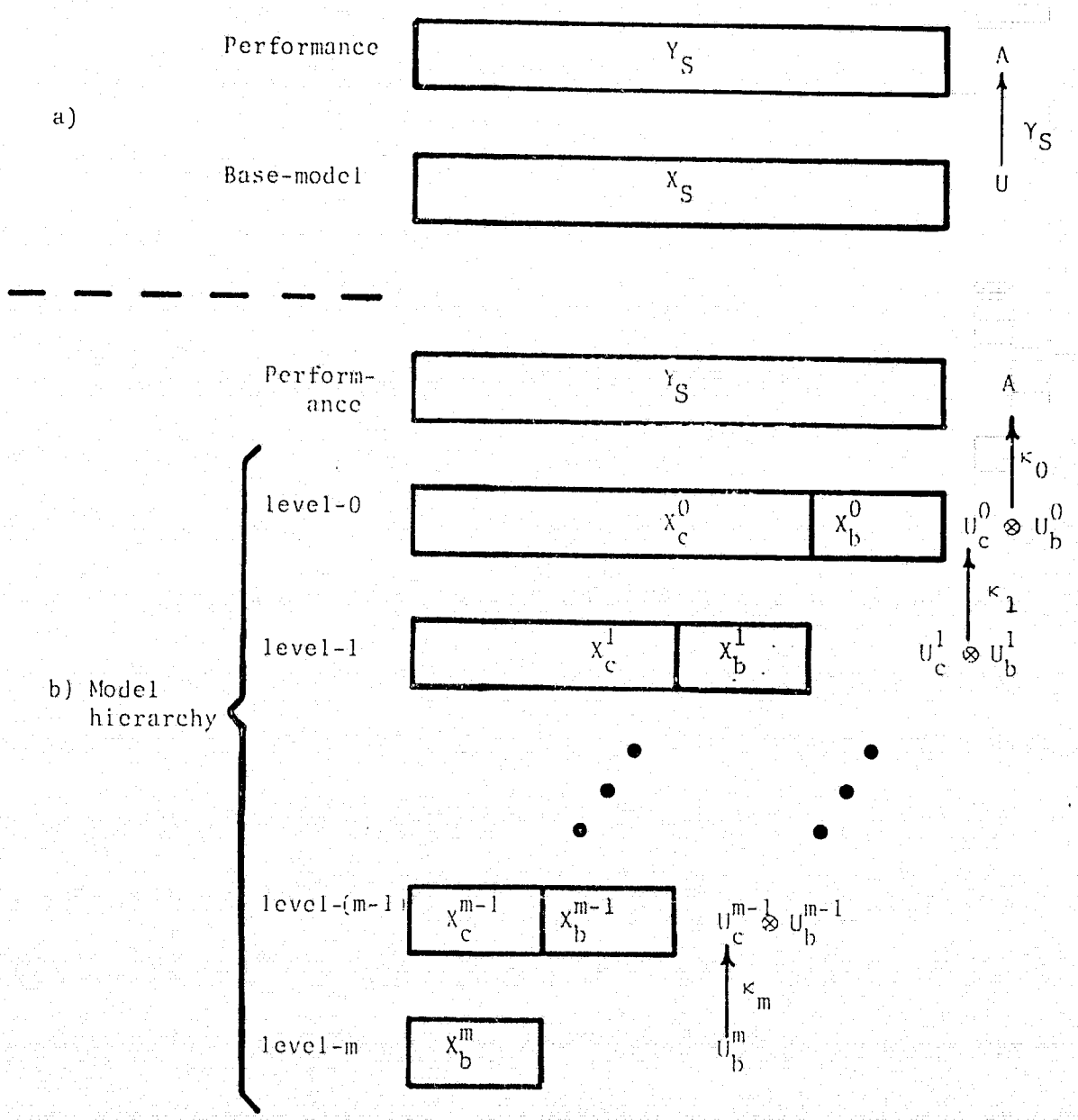


Figure 1. A total system S with base-model X_S(1.a) and a model hierarchy for S(1.b).

A model hierarchy thus provides a step-by-step formulation of the capability function in terms of interlevel translations of state trajectories, beginning with a translation of the bottom model. It also permits the expression of capability relative to higher level (less detailed) views of total system behavior. More precisely, beginning at the highest level, the <u>i-level based capability function</u> (denoted γ_i) can be defined inductively as follows. Recalling that $U^i = U^i_c \otimes U^i_b$ and letting $U_b(i) = U^i_b \otimes U^{i-1}_b \otimes \ldots \otimes U^0_b$, if $i = 0$ then

$$\gamma_0 : U_0 \to A \text{ where } \gamma_0(u) = \kappa_0(u).$$

If $i > 0$, then $\gamma_i : U^i \otimes U_b(i-1) \to A$ where, if $u \in U^i$, $u' \in U_b(i-1)$ then

$$\gamma_i(u,u') = \gamma_{i-1}(\kappa_i(u),u').$$

It is easily shown that γ_i has its intended interpretation, i.e., if u and u' correspond to a base model trajectory v then $\gamma_i(u,u') = \gamma_S(v)$. In particular, if $i = m$ then $\gamma_m = \gamma_S$.

The capability functions γ_i, in turn, provide the basis for a systematic method of determining $\gamma^{-1}(a)$ for a given accomplishment level a. Beginning with level-0-based capability, by (4.1) we have

$$\gamma_0^{-1}(a) = \kappa_0^{-1}(a).$$

Assuming that $\gamma_{i-1}^{-1}(a)$ has been determined, by (4.2) it follows that

$$\gamma_i^{-1}(a) = \bigcup_{(u,u') \in \gamma_{i-1}^{-1}(a)} (\kappa_i^{-1}(u),u').$$

where $(\kappa_i^{-1}(u),u') = \{(v,u')|\kappa_i(v) = u\}$.
This process is iterated until i=m, yielding $\gamma_m^{-1}(a) = \gamma_S^{-1}(a)$.

We have described one way of evaluating the sets $\gamma_S^{-1}(a)$ by introducing a model hierarchy which permits us to write γ_S as the composition of several smaller functions, namely the interlevel translations. A simplified but relatively complete example of this process of evaluating the sets $\gamma_S^{-1}(a)$ is presented in Section 3.5.

3.3.2 Functional Dependence

Elaboration of the capability function γ_S yields a characterization of the trajectory sets corresponding to a particular level of accomplishment a. These trajectory sets may possess properties which either aid or hinder probability calculations. One such property is the apparent functional dependency of the various system components on one another. For instance, the knowledge that certain dependencies exist between the operational states of a system over time may permit the simplification of considering certain states of the system only at specific times. The concept of R-dependence (see [1], [2]) is a characterization of functional dependency as reflected in trajectory sets.

The remainder of Section 3.3 introduces a further generalization of R-dependence, together with the notion of "conditional" R-dependence, and some basic properties of these concepts. The idea of conditional R-dependence is embodied in the question "If C is known, does the knowledge of B increase the knowledge of A?" For the purposes of the following discussion, we restrict consideration to systems whose trajectories may be

sampled at discrete intervals with no loss of relevant information.

3.3.2.1 Basic Definitions

Suppose we have a "phased" model (Section 3.4.2) of system S. Let S be the system of interest with subsystems $S_1,\ldots,S_n$. A subsystem may be any part of the total system (that is, the computer and its environment) whose behavior influences the overall performance of S. An operational state of S will be defined in terms of the operational states of the subsystems of S. Thus, the sets of states of S considered here are "structured" or "coordinatized" sets in the following sense.

Definition 1: Let D be a totally ordered (index) set. A structured set V is some subset of the Cartesian (cross) product of an indexed family of sets $\{V_d | d \varepsilon D\}$, that is, $V \subseteq \underset{d\varepsilon D}{X} V_d$ (see [3]).

Note that the ordering on the index set D may be arbitrary. However, once chosen, the ordering is fixed. Any set $D' \subseteq D$ will inherit that ordering, and cross products will be taken according to the order of the indices $d\varepsilon D'$.

Two examples should help to clarify this definition. In [1], with each subsystem $S_i (1 \le i \le n)$ of S was associated a corresponding state set Q_i. The state set Q of S was defined to be $Q = Q_1 x \ldots x Q_n$. This set Q is a structured set where the index set is $D = \{1,2,\ldots,n\}$ with the natural ordering. The collection of sets $\{Q_i | 1 \le i \le n\}$ corresponds to $\{V_d | d\varepsilon D\}$ of Definition 1. In [2], the set of state trajectories of a system S is described. Each subsystem S_i is sampled at k different times. The set Q_i^t $(1 \le i \le n,\ 1 \le t \le k)$ denotes the set of possible operational

states of S_i at the tth time sample. Then a state trajectory for S is an n x k array $u = [q_{it}]$ where $q_{it} \varepsilon Q_i^t$. Let $U = \{[q_{it}] | q_{it} \varepsilon Q_i^t, 1 \le i \le n, 1 \le t \le k\}$. The set U is a structured set with index set $D = \{(i,t) | 1 \le i \le n, 1 \le t \le k\}$ totally ordered, the indexed family of sets is $\{Q_i^t | (i,t) \varepsilon D\}$, and $U = \underset{(i,t)\varepsilon D}{X} Q_i^t$. Throughout this report, the ordering imposed on a structured index set as in this example will be row-major order. This ordering is defined by

$$(a,b) < (c,d) \quad \text{if } a < c \text{ or } (a=c \text{ and } b<d)$$
$$(a,b) = (c,d) \quad \text{if } a=c \text{ and } b=d$$
$$(a,b) > (c,d) \quad \text{otherwise.}$$

Due to this linear ordering we can represent a state trajectory as either an n x k array or as an $(n \cdot k)$-tuple. We shall use whichever representation is most suggestive in what follows. Often an m-tuple (arbitrary m) is used. The methods and results described, however, apply to arrays of any dimension and size.

For any structured set V with index set D one can define a family of (single) coordinate projections which, when applied to an element $v \varepsilon V$, will yield the value of a particular coordinate of v.

<u>Definition 2</u>. Let V be a structured set, $V \subseteq \underset{d\varepsilon D}{X} V_d$. For each $d \varepsilon D$, the <u>projection on d</u>, denoted ξ_d, is the function

$$\xi_d : V \to V_d \quad \text{where}$$
$$\xi_d((v_1,\ldots,v_d,\ldots,v_m)) = v_d.$$

While the family of projections $\{\xi_d | d \varepsilon D\}$ provides a method for examining the value of a single coordinate, one would like to be able to examine the values of several coordinates simultaneously.

In order to make the requisite extension, the notion of a cross product function [3] is introduced.

<u>Definition 3</u>. Let V be a structured set with index set D, let $D' \subseteq D$, $D' \neq \emptyset$ and let $\{f_d : V \to V_d \mid d \in V\}$ be an indexed family of functions. A <u>cross product function</u> on V is a function

$$(\mathop{\times}_{d \in D'} f_d) : V \to \mathop{\times}_{d \in D'} V_d$$

defined by

$(\mathop{\times}_{d \in D'} f_d)(v) = (f_{d_1}(v), \ldots, f_{d_j}(v))$ where $D' = \{d_1, \ldots, d_j\}$ and $d_1 < d_2 < \ldots < d_j$.

For $D' = \emptyset$, define

$$(\mathop{\times}_{\emptyset} f_{\emptyset}) : V \to \{1_{\emptyset}\} \text{ where } 1_{\emptyset} \text{ is an arbitrary constant.}$$

Using Definition 3 one can define projections on sets of coordinates. Thus for V, D, D' and $\{\xi_d \mid d \in D\}$ as above, define

$$\xi_{D'} = (\mathop{\times}_{d \in D'} \xi_d).$$

For example, if $V = \mathbf{R} \times \mathbb{R} \times \mathbf{R}$, $D = \{1,2,3\}$ and $D' = \{2,3\}$, then $\xi_{D'}((8,9,10)) = (9,10)$. For an array, an example is

$$\xi_{\{(1,1),(2,3)\}} \left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \right) = (1,6) \text{ and}$$

$$\xi_{\emptyset} \left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \right) = 1_{\emptyset}$$

When $|D'| = 1$ (D' is a singleton set) the set brackets will often be dropped, i.e.,

$$\xi_{\{1\}}((8,9,10)) = \xi_1((8,9,10)) = (8).$$

For $V' \subseteq V$, $D' \subseteq D$ define

$$\xi_{D'}(V') = \{\xi_{D'}(v') \mid v' \in V'\}$$

In this report, a set will usually be partitioned relative to projection equality. This is defined by

$$v_1 \equiv_A v_2 \text{ if } \xi_A(v_1) = \xi_A(v_2),\ A \subseteq D,\ v_1, v_2 \in V.$$

Thus if we partition V by equality of projection on A, denoted π_A^V, then two elements $v_1, v_2 \in V$ are in the same block of π_A if $v_1 \equiv_A v_2$. In cases where ambiguity is precluded, we may denote π_A^V by π_A. For example, let $D = \{1,2,3\}$ and

$$V = \left\{\begin{matrix}(0,0,0),(1,0,0)\\(0,0,1),(1,0,1)\\(0,1,0),(1,1,0)\\(0,1,1),(1,1,1)\end{matrix}\right\}.$$

Then

$$\pi_{\{1\}} = \{\{(0,0,0),(0,0,1),(0,1,0),(0,1,1)\},$$
$$\{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}\},$$
$$\pi_{\{2,3\}} = \{\{0,0,0),(1,0,0)\},\{(0,0,1),(1,0,1)\},$$
$$\{(0,1,0),(1,1,0)\},\{(0,1,1),(1,1,1)\}\}, \text{ and}$$
$$\pi_{\emptyset} = V .$$

If π_A^V and π_B^V are two partitions of V and each block in π_A is a subset of a block in π_B, then π_A "refines" π_B, denoted $\pi_A \leq \pi_B$ or $\pi_B \geq \pi_A$.

It will be useful to be able to refer to a particular block of π_A^V $(A \subseteq D)$. Thus if $s \in \xi_A(R)$ define

$$\mathbf{B}_A^V(s) = \{q \in V \mid \xi_A(q) = s\}.$$

The set $\mathbf{B}_A^V(s)$ is the set of all elements of V whose projection on A is equal to s. Clearly, if $\xi_A(V) = \{s_1,\ldots,s_m\}$ then

$$\pi_A^V = \{\mathbf{B}_A^V(s_1),\ldots,\mathbf{B}_A^V(s_m)\}.$$

As with partitions, the superscript V will be dropped when the meaning is unambiguous. (The notation $\mathbb{B}^R_{\{i\}}(q_j)$ corresponds to the $R_\varphi(j,q_j)$ of [2]). Further to the above example,

$$\mathbb{B}^V_{\{1\}}((1)) = \{(1,0,0),(1,1,0),(1,0,1),(1,1,1)\}$$

and

$$\mathbb{B}^V_{\{2,3\}}((0,0)) = \{(1,0,0),(0,0,0)\}.$$

3.3.2.2 R-dependence

In [1], the CARSRA notion of functional dependence (see [5]) was formalized as φ-dependence. Its application was restricted to the context of structure-based reliability analysis. The concept of φ-dependence was subsequently extended in [2] to R-dependence. While this extension provided several useful generalizations, it still restricted "dependence" to a single coordinate depending on another single coordinate. The extension to subsets of coordinates is given below together with certain basic results. It should be noted that previous characterizations are special cases of the extended definition.

The context of our investigation is that one knows something about the behavior of the system; that is, one has been given some set of states or state trajectories which give rise to a certain desired performance of the system of interest. One such set might be the set of all "success" states relative to a structure function. More generally, these sets may be induced by the values which the capability function γ [see sec 3.2] assumes. Thus, in the following discussion, let $Q = Q_1 \times \ldots \times Q_m$ be a structured set and let $R \subseteq Q$. R is the set relative to which

R-dependency is defined. Let $D = \{1,\ldots,m\}$ be the index set for Q and R.

<u>Definition 4</u>. If $A, B \subseteq D$ then <u>A R-depends on B</u> (denoted $A \Delta_R B$ or $(A,B) \in \Delta_R$) if $\exists r \in \xi_A(R)$ and $\exists s \in \xi_B(R)$ such that $\forall q \in R[\xi_B(q)=s = \xi_A(q) \neq r]$.

Accordingly, A is R-independent of B ($A \not\Delta_R B$) if and only if $\forall r \in \xi_A(R)$, $\forall s \in \xi_B(R), \exists q \in R[\xi_A(q) = r$ and $\xi_B(q) = s]$.

Several items should be noted about this definition. First, there are two general modes of dependence (see also [2]). The "stronger" mode is dependence as found in "linear dependence" of vector spaces. There "A depends on B" means that knowing B tells us everything of interest about A. The "weaker" mode is demonstrated by statistical dependence. In this case, knowing that "A depends on B" and knowing B tells something about A. R-dependence is a type of weak dependence. Second, R-dependence is defined relative to the set R under consideration. The fact that A R-depends on B does <u>not</u> mean that A R-depends on B relative to a set $R' \supseteq R$. Third, the notion of R-dependence is extended to sets of coordinates. However, in the attempt to achieve maximum generality, references to subsystems have been dropped. Since the specific context of our research indicates that each coordinate represents an operational state of a subsystem at some particular time, the relationships between subsystems may easily be inferred.

The above definition of R-dependence is related to the projection approach ([1],[2]) and the partition approach ([2]) as stated in the following theorem.

<u>Theorem 1</u>: Let $R \subseteq Q$ and $A, B \subseteq D$. The following statements are equivalent.

i) A R-depends on B.

ii) $\exists s \in \xi_B(R)$ such that $\xi_A(\mathbb{B}_B^R(s))$ is a proper subset of $\xi_A(R)$.

iii) $\exists r \in \xi_A(R)$ and $\exists s \in \xi_B(R)$ such that $\mathbb{B}_A(r) \cap \mathbb{B}_B(s) = \emptyset$.

<u>Proof</u>: (i) $\Rightarrow$ (ii). Suppose $A \Delta_R B$ and let r,s be as guaranteed in Definition 4. Let $q \in \mathbb{B}_B(s)$. Then $\xi_B(q) = s$ which implies that $\xi_A(q) \neq r$, i.e., $r \notin \xi_A(\mathbb{B}_B(s))$. But $r \in \xi_A(R)$ so $\xi_A(\mathbb{B}_B(s)) \subset \xi_A(R)$.

(ii) $\Rightarrow$ (iii). Suppose (ii) and let $\mathbb{B}_B(s)$ be the block with the property guaranteed by (ii), $s \in \xi_B(R)$. Then $\exists r \in \xi_A(R)$ such that $r \notin \xi_A(\mathbb{B}_B(s))$. If we now consider $\mathbb{B}_A(r)$ it must be the case that $\mathbb{B}_A(r) \cap \mathbb{B}_B(s) = \emptyset$. (Suppose not. Then $\exists q \in R$ such that $q \in \mathbb{B}_A(r)$ and $q \in \mathbb{B}_B(s)$. But $q \in \mathbb{B}_B(s) \Rightarrow \xi_A(q) \in \xi_A(\mathbb{B}_B(s))$ and $q \in \mathbb{B}_A(r) \Rightarrow \xi_A(q) = r$. Therefore, $r \in \xi_A(\mathbb{B}_B(s))$. Contradiction.)

(iii) $\Rightarrow$ (i). Suppose $\exists r \in \xi_A(R)$, $s \in \xi_B(R)$ such that $\mathbb{B}_A(r) \cap \mathbb{B}_B(s) = \emptyset$. Then $\forall q \in R$, if $q \in \mathbb{B}_B(s)$, $q \notin \mathbb{B}_A(r)$. But $q \in \mathbb{B}_B(s) \Leftrightarrow \xi_B(q) = s$ and similarly $q \in \mathbb{B}_A(r) \Leftrightarrow \xi_A(q) = r$. Thus $\exists r \in \xi_A(R)$, $\exists s \in \xi_B(R)$ such that $\forall q \in R[\xi_B(q) = s \Rightarrow \xi_A(q) \neq r]$.

Due to the equivalence of the above three formulations, we are now free to use whichever is most applicable when deriving new results. It should be noted that (ii) in Theorem 1 corresponds to the "projection" formulation of [2] and (iii) corresponds to the "partition" formulation of [2], each extended to deal with subsets of coordinates.

Consider the following. Let

$$R = \left\{\begin{matrix}(0,0,0,0),(0,1,1,0)\\(0,0,1,0),(1,0,1,0)\\(0,1,0,0),\\(1,0,0,0),\end{matrix}\right\}, \quad D = \{1,2,3,4\}.$$

Then $\forall q \in R[[\xi_1(q) = 1 \Rightarrow \xi_2(q) \neq 1]$ so $\{1\} \Delta_R \{2\}$. However, $\{1\} \not\Delta_R \{3\}$. Looking at coordinate 4, $\pi_4 = \{R\}$ because the value of coordinate 4 is constant. In this case no coordinate or set of coordinates may R-depend on $\{4\}$. We call such a set of coordinates "universally independent" [6]. Now consider $R' = R \cup \{(1,1,0,0)\}$. Again $\{4\}$ is universally independent, but $\{1\} \not\Delta_{R'} \{2\}$. However, $\forall q \in R'$ $[\xi_{\{1,2\}}(q) = (1,1) \Rightarrow \xi_3(q) \neq 1]$ so $\{1,2\} \Delta_{R'} \{3\}$.

Several observations should be made regarding the nature of R-dependence. First, R-dependence is symmetric, that is, $\forall A,B \subseteq D$ if $A \Delta_R B$, then $B \Delta_R A$. This fact is easily seen from (iii) of Theorem 1. Second, if $|\xi_A(R)| = 1$, $A \subseteq D$, then A will always be R-independent of any other set $B \subseteq D$, including A itself (i.e., A is universally independent). However, $\forall A \subseteq D$ such that $|\xi_A(R)| > 1$, A R-depends on A. This is easily seen from the fact that if $|\xi_A(R)| > 1$ then $\exists r,s \in \xi_A(R)$ such that $r \neq s$. Then $\forall q \in R[\xi_A(q) = r \Rightarrow \xi_A(q) = r \neq s]$. R-dependence is not transitive. Consider the set

$$R = \left\{\begin{matrix}(0,0,0)\\(1,0,0)\\(0,0,1)\\(1,1,1)\end{matrix}\right\}.$$

Then $\{1\} \Delta_R \{2\}$ and $\{2\} \Delta_R \{3\}$ but $\{1\} \not\Delta_R \{3\}$. Hence, in general, R-dependence is neither reflexive nor transitive.

The following result is useful in establishing other properties of R-dependence.

<u>Lemma 1</u>. Let $A,B \subseteq D$. If $A \Delta_R B$ then $\forall A' \supseteq A$, $\forall B' \supseteq B$, $A' \Delta_R B'$.

Proof: Suppose $A \Delta_R B$. Let $A' \supseteq A$, $B' \supseteq B$. We know that $\exists r \in \xi_A(R)$, $\exists s \in \xi_B(R)$ such that

$$\mathbf{B}_A(r) \cap \mathbf{B}_B(s) = \emptyset.$$

Since $A \subseteq A'$, $\pi_{A'} \leq \pi_A$, that is each block in $\pi_{A'}$ is a subset of some block in π_A and each block in π_A is a superset of some block in $\pi_{A'}$. Similarly for B' and B. Hence $\exists r' \in \xi_{A'}(R)$, $\exists s' \in \xi_{B'}(R)$ such that

$$\mathbf{B}_{A'}(r') \cap \mathbf{B}_B(s) = \emptyset$$

and so

$$\mathbf{B}_{A'}(r') \cap \mathbf{B}_{B'}(s') = \emptyset.$$

Therefore $A' \Delta_R B'$. Intuitively, this lemma says that if A R-depends on B, then A R-depends on any superset of B.

The notions of "strong" and "weak" dependence were introduced above. R-dependence itself is a weak form of dependence, of which a strong form is a special case. This special case is distinguished as follows. Consider (iii) of Theorem 1. Suppose that for A, $B \subseteq D$ where $|\xi_A(R)| > 1$ and $|\xi_B(R)| > 1$, $\pi_A{}^R \leq \pi_B{}^R$. Then $\forall r \in \xi_A(R)\ \exists s \in \xi_B(R)$ such that $\mathbf{B}_A(r) \subseteq \mathbf{B}_B(s)$. Clearly A R-depends on B. But also, $\forall q \in R.[\xi_A(q) = r \Rightarrow \xi_B(q) = s]$ for r,s as designated above. Thus if one knows the values of the coordinates in A, one knows the values of the coordinates in B. This is precisely a characterization of a type of strong dependence. Thus if

$|\pi_A^R| > 1$, $|\pi_B^R| > 1$, and $\pi_A \leq \pi_B$, then A R-depends (strongly) on B. Note that while the weak form of R-dependence is symmetric, the stronger form is not necessarily so.

So far the notion of a set of coordinates R-depending on another set of coordinates has been introduced. Eventually, one would like to have a quick test to discover, given a structured set with its index st, whether any dependencies exist. In order to characterize a set which contains dependencies, the notion of an "R-dependent" set of coordinates has been introduced.

Definition 5. Let $C \subseteq D$. C is R-dependent if $\exists A,B \subseteq C$ where $A \cap B = \emptyset$ and A R-depends on B. C is R-independent if C is not R-dependent.

Essentially, this says that C is R-dependent if some part of C R-depends on some other part of C. The requirement that A and B be disjoint insures that a set is not characterized as dependent simply because some subset of coordinates R-depends upon itself. (If this qualification was not made, then only universally independent sets of coordinates would be R-dependent.)

Theorem 2: A coordinate set C is R-dependent if and only if $\exists i \in C$ such that $\{i\} \Delta_R C-\{i\}$.

Proof: ($\Leftarrow$) Suppose that $\exists i \in C$ such that $\{i\} \Delta_R C-\{i\}$. By choosing $A = \{i\}$ and $B = C-\{i\}$, C is R-dependent by Definition 6.

($\Rightarrow$) Suppose C is R-dependent. Then $\exists A,B \subseteq C$ such that $A \cap B = \emptyset$ and $A \Delta_R B$. This means that $\exists r \in \xi_A(R)$, $\exists s \in \xi_B(R)$ such that

$$\mathbf{B}_A(r) \cap \mathbf{B}_B(s) = \emptyset .$$

Let $A = \{i_1,\ldots,i_k\}$, $B = \{j_1,\ldots,j_\ell\}$ and $s = (s_1,\ldots,s_\ell)$. Notice $\mathbb{B}_B(s) = \mathbb{B}_{j_1}(s_1) \cap \mathbb{B}_{j_2}(s_2) \cap\ldots\cap\mathbb{B}_{j_\ell}(s_\ell)$. (There is nothing special about using B. The argument is the same whether we choose to start with A or B.) Then $\mathbb{B}_A(r) \cap \mathbb{B}_{j_1}(s_1) \cap\ldots\cap\mathbb{B}_{j_\ell}(s_\ell) = \emptyset$. Consider $\mathbb{B}_A(r) \cap B_{j_1}(s_1)$. This intersection is either empty or non-empty. If $\mathbb{B}_A(r) \cap \mathbb{B}_{j_1}(s_1) = \emptyset$ then $\{j_1\} \Delta_R A$. From Lemma 1, $A \subseteq C-\{j_1\}$ so $\{j_1\} \Delta_R C-\{j_1\}$ and we are done.

Suppose $\mathbb{B}_A(r) \cap \mathbb{B}_{j_1}(s_1) \neq \emptyset$. Then $\exists q \in R$ such that $\xi_A(q) = r$ and $\xi_{j_1}(q) = s_1$. Let $A' = A \cup \{j_1\}$. There exists a $t \in \xi_{A'}(R)$ such that $\xi_A(\mathbb{B}_{A'}(t')) = r$ and $\xi_{j_1}(\mathbb{B}_{A'}(t)) = s_1$, i.e., $\mathbb{B}_{A'}(t) = \mathbb{B}(r) \cap \mathbb{B}_j(s_1) \neq \emptyset$.

Consider now $\mathbb{B}_{A'}(t) \cap \mathbb{B}_{j_2}(s_2)$. Either this intersection is empty or non-empty. Employing the above arguments, either $\{j_2\} \Delta_R C-\{j_2\}$ or $\exists t'' \in \xi_{A' \cup \{j_2\}}(R)$ such that $\mathbb{B}_{A' \cup \{j_2\}}(t'') = \mathbb{B}_{A'}(t') \cap \mathbb{B}_{j_2}(s_2)$. Now look at $\mathbb{B}_{A' \cup \{j_2\}}(t'') \cap \mathbb{B}_{j_3}(s_3)$ and repeat. The process must terminate at some j_m, i.e., $\{j_m\}$ R-depends on $C-\{j_m\}$, i.e.,

$$\mathbb{B}_{A \cup \{j_1,\ldots,j_{m-1}\}}(t^*) \cap \mathbb{B}_{j_m}(s_m) = \emptyset$$

because, at worst,

$$(\mathbb{B}_A(r) \cap \mathbb{B}_{j_1}(s_1) \cap\ldots\cap \mathbb{B}_{j_{\ell-1}}(s_{\ell-1})) \cap \mathbb{B}_{j_\ell}(s_\ell) = \emptyset .$$

Therefore, C is R-dependent if and only if $\exists i \in C$ such that $\{i\} \Delta_R C-\{i\}$.

<u>Corollary</u>: D is R-dependent if and only if $\exists i \in C$ such that $\{i\}$ R-depends on $D-\{i\}$. Conversely, D is R-independent if and only if $\forall i \in D$, $\{i\}$ is R-independent of $D-\{i\}$.

To further the understanding of R-dependence and to continue the search for simple characterizations of R-dependent (R-independent) coordinate sets, we now turn to consideration of R-independence. From results derived above one can immediately make the following characterization.

<u>Theorem 3</u>: Let $A,B \subseteq D$ be disjoint sets and let Ψ be a coordinate mapping such that

$$\forall q \in R[\Psi(\xi_{A \cup B}(q)) = (\xi_A(q),\xi_B(q))]$$

(Such a map Ψ always exists.) Then A is R-independent of B if and only if $\Psi(\xi_{A \cup B}(R)) = \xi_A(R) \times \xi_B(R)$.

<u>Proof</u>: Suppose that $A \not\Delta_R B$. It suffices to show that Ψ is onto. Let $r \in \xi_A(R)$, $s \in \xi_B(R)$. By negation of Definition 5, $\exists q \in R$ such that $\xi_A(q) = r$ and $\xi_B(q) = S$. Accordingly, $\Psi(\xi_{A \cup B}(q)) = (\xi_A(a),\xi_B(q)) = (r,s)$.

Conversely, suppose Ψ is onto. Then $\forall r \in \xi_A(R)$ and $\forall s \in \xi_B(R)$, $\exists q \in R[\Psi(\xi_{A \cup B}(q)) = (r,s)]$. But $r \in \xi_A(R)$ and $s \in \xi_B(R)$ so $\exists q \in R[\xi_A(q) = r$ and $\xi_B(q) = s]$. Hence $A \not\Delta_R B$.

This theorem says that if two disjoint sets A,B are such $(A,B) \notin \Delta_R$, then the projection relative to A and B can be written as a cross product. More precisely, the coordinate mapping Ψ re-orders the set $A \cup B$ such that all elements of A appear before the elements of B. (The union operator preserves the original ordering of D in $A \cup B$ and so may interweave elements of A and B.) The re-ordering allows one to write the (re-ordered) set as a Cartesian product.

For example, let $D = \{1,2,3\}$ and let R be as given in Table I where, for each $q \in R$ we associate a unique label to simplify notation. Let $A = \{1,3\}$, $B = \{2\}$. Then $A \cup B = D$. Then

$$\pi_A^R = \{\{ac\},\{bd\},\{eg\},\{fh\}\} \text{ and}$$

$$\pi_B^R = \{\{abef\},\{cdgh\}\}.$$

Each block of π_A has a non-trivial intersection with both blocks of π_B so $A \not\sim_R B$. If $\Psi:\{0,1\}^3 \to \{0,1\}^3$ such that $\Psi((q_1,q_2,q_3)) = (q_1,q_3,q_2)$ it is clear that $\Psi(\xi_{A \cup B}(R)) = \xi_A(R) \times \xi_B(R) = \{0,1\}^2 \times \{0,1\}$.

R	label
(0,0,0)	a
(0,0,1)	b
(0,1,0)	c
(0,1,1)	d
(1,0,0)	e
(1,0,1)	f
(1,1,0)	g
(1,1,1)	h

Table I

Further examination of R shows that R is in fact a Cartesian set. How do such sets fit into the notion of R-independence?

The relation is demonstrated in Theorem 4 below. However, in order to ease the discussion we first prove the following useful result.

<u>Lemma 2</u>: Let $A,B \subset D$. If $A \not\perp_R B$ then $\forall A' \subset A$, $\forall B' \subset B$, $A' \not\perp_R B'$.

<u>Proof</u>: Let A,B be as above, $A' \subseteq A$, and $B' \subseteq B$. Because A is R-independent of B, $\forall r \in \xi_A(R))$, $\forall s \in \xi_B(R))$ $\exists q \in R[\xi_A(q) = r$ and $\xi_B(q) = s]$, that is $\forall r \in \xi_A(r)$ and $\forall s \in \xi_B(R)$, $\mathbb{B}_A(r) \cap \mathbb{B}_B(s) \neq \emptyset$. Reflection shows that if $r' = \xi_{A'}(\mathbb{B}_A(r)) \in \xi_{A'}(R)$ then $\mathbb{B}_A(r) \subseteq \mathbb{B}_{A'}(r')$. Thus for r', s' so described ($s' = \xi_{B'}(\mathbb{B}_B(s))$, $\mathbb{B}_{A'}(r') \cap \mathbb{B}_{B'}(s') \neq \emptyset$.

Since each $r \in \xi_A(R)$ has an associated $r' \in \xi_{A'}(R)$ (similarly for s,s') we see that

$$\forall r \in \xi_A(R), \forall s \in \xi_B(R)[\mathbb{B}_A(r) \cap \mathbb{B}_B(s) \neq \emptyset] \Rightarrow \forall r' \in \xi_{A'}(R), \forall s' \in \xi_{B'}(R)$$
$$[\mathbb{B}_{A'}(r') \cap \mathbb{B}_{B'}(s') \neq \emptyset].$$

Therefore A' is R-independent of B'.

This says that if A is R-independent of B then any subset of A is R-independent of any subset of B. This knowledge is used to obtain the following characterization of an R-dependent set of coordinates.

<u>Theorem 4</u>: A coordinate set $A \subseteq D$ is R-independent if and only if $\xi_A(R) = \bigvee_{a \in A} \xi_a(R)$.

<u>Corollary</u>: R is Cartesian if and only if $\forall d \in D$, $\{d\}$ is R-independent of $D-\{d\}$.

<u>Proof</u>: Suppose A is R-independent, that is, $\forall a \in A$, $\{a\}$ is R-independent of $A-\{a\}$. Relabel the elements $a_1,\ldots,a_m$ of A as $1,\ldots,m$. For $a = 1$, by Theorem 3, we have $\Psi(\xi_A(R)) = \xi_1(R) \times$

$\xi_{A-\{1\}}(R)$. Since a = 1 is the first element of A, Ψ is just the identity function, i.e., $\Psi(\xi_A(R)) = \xi_A(R) = \xi_1(R) \times \xi_{A-\{1\}}(R)$. Consider the coordinate set A' = A-{1}. Then {2} is R-independent of A'-{2} since, by assumption, {2} is R-independent of A-{2} and so by Lemma 2 {2} is R-independent of A'-{2} (A'-{2} $\subseteq$ A-{2}). Repeating the above argument with a = 2 we derive $\xi_{A'}(R) = \xi_2(R) \times \xi_{A'-\{2\}}(R)$ and therefore $\xi_A(R) = \xi_1(R) \times \xi_2(R) \times \xi_{A'-\{2\}}(R)$. Continuing in this fashion we obtain

$$\xi_A(R) = \bigtimes_{a \in A} \xi_a(R).$$

Conversely, suppose $\xi_A(R) = \bigtimes_{a \in A} \xi_a(R)$. Let a ε A and let Ψ_a be the coordinate transformation of A which replaces the first coordinate of A by a and increases by one the rank (in the ordering) of every other coordinate in A. (For example, $\Psi_3((q_1,q_2,q_3,q_4)) = (q_3,q_1,q_2,q_4))$. Then

$$\Psi_a(\xi_A(R)) = \xi_a(R) \times \bigtimes_{a' \in A-\{a\}} \xi_{a'}(R)$$
$$= \xi_a(R) \times \xi_{A-\{a\}}(R).$$

By Theorem 3, {a} is R-independent of A-{a}.

From the corollary to Theorem 4 we see that R-independence of the coordinate set D characterizes a Cartesian structure for R. This yields a straightforward computational test for discovering whether D has an absence of R-dependent coordinate subsets, namely, test R to see if it is Cartesian. The fact that R-independent coordinate subsets correspond to Cartesian projections (Theorem 3) likewise provides a simple test for the R-independence of a pair of coordinate sets.

In any large system, the number of disjoint coordinate

sets which may R-depend upon each other is also very large, and not all the dependencies reflected may be relevant to the analysis. One area for further investigation is in determining which coordinate sets to examine. Along with this problem is the problem of characterizing the strength of dependency between sets. Given $A \Delta_R B$, one possible measure is the minimal number of elements of Q which must be included in R so that A is R-independent of B. This is important because, in general, the stronger the dependence between a set of subsystems, the likelier it is to be able to use that set as a subunit in decomposing the overall system.

3.3.2.3 Conditional R-Dependence

The idea of conditional dependence was characterized in Section 3.3.2 by the question "If C is known, does the knowledge of B increase the knowledge of A?" If so, we say that "A depends on B given C." More formally, in the context of R-dependence, we introduce the following.

Definition 6: For $A,B,C \subseteq D$, A R-depends on B given C (denoted $(A \Delta_R B)|C$) if $\exists t \in \xi_C(R)$, $\exists s \in \xi_B(\mathbb{B}_C(t))$, $\exists r \in \xi_A(\mathbb{B}_C(t))$ such that $\forall q \in \mathbb{B}_C(t)[\xi_A(q) = s \Rightarrow \xi_A(q) \neq r]$.

This concept of conditional R-dependence can be likened to the concept of conditional probabilities. The closest parallel lies in the observation that in both cases, by reducing the universe of discourse to the particular subset (subpopulation) under consideration, all theorems about "absolute" R-dependencies

(probabilities) once again hold. In effect, the presence of conditional R-dependence may be alternately regarded as showing the existence of R'-dependence for particular choices of R', that is, $(A \Delta_R B)|C$ if $\exists t \in \xi_C(R)$ such that A $\mathbb{B}_C(t)$-depends on B.

For example, let $D = \{1,2,3,4\}$, $A = \{1,2\}$, $B = \{4\}$, $C = \{3\}$, and

$$R = \left\{\begin{matrix} (0,1,0,0) \\ (1,0,0,0) \\ (1,0,0,1) \\ (0,1,1,0) \\ (0,1,1,1) \\ (1,0,1,1) \end{matrix}\right\} .$$

Then for $\xi_C(q) = 0$ (i.e., $t = (0)$ in Definition 7), $\forall q \in \mathbb{B}_3(0)$, $\xi_A(q) = (0,1) \Rightarrow \xi_B(q) \neq 1$. Hence $(\{1,2\} \Delta_R \{4\})|\{3\})$. Note that $\mathbb{B}_3(0) = \{(0,1,0,0),(1,0,0,0),(1,0,0,1)\}$ so, alternatively, $\{1,2\}$ $\mathbb{B}_3(0)$-depends on $\{4\}$.

One property that conditional dependence should have follows: if $B \subset C$, then A should be R-independent of B given C. This is intuitively justified by the argument that because knowledge carried by B is contained in the knowledge carried by C, no further information is being added. That this is a property of conditional R-dependence as in Definition 6 is shown in Theorem 5.

<u>Theorem 5</u>: If $A,B,C \subseteq D$ and $B \subseteq C$ then A does not R-depend on B given C (A is R-independent of B given C).

<u>Proof</u>: Let $t \in \xi_C(R)$. If $B \subseteq C$ then $\xi_B(\mathbb{B}_C(t)) = \{s_0\}$ for some $s_0 \in \xi_B(R)$, i.e., ξ_B takes on but one value. Let $r \in \xi_A(\mathbb{B}_C(t))$. We must show that $\exists q \in \mathbb{B}_C(t)$ such that $\xi_A(q) = r$ (since $\forall q \in \mathbb{B}_C(t)$, $\xi_B(q) = s_0$). But this is true by definition of

$\xi_A(\mathbb{B}_C(t))$. Hence A is R-independent of B given C when $B \subseteq C$. One may verify that in the above example that $(\{1,2\} \not\perp_R \{4\})|\{3,4\}$.

The study of the properties of conditional R-dependence has just begun. It is hoped that it will provide a powerful tool for use in system decomposition. One area for further investigation is the delineation of the properties of conditional R-dependence. Another area is suggested by the similarities between probabilistic notions and R-dependence concepts. What is the relationship between existing R-dependencies and the underlying stochastic processes? How can such relationships be discovered and used? These questions guide our further research.

3.4 Computation of Trajectory Set Probabilities

As discussed in Section 3.2, if S is a total system, the performability of S for accomplishment level $a \in A$ may be expressed as

$$p_S(a) = Pr(\gamma^{-1}(a)).$$

The research reported in this section concerns the evaluation of probability function Pr for a given trajectory set $\gamma^{-1}(a)$. During the previous reporting period, this problem was studied for the special case when the capability function γ is phasewise structure based (see [2], Section 3.2.2) in the sense of Esary and Ziehms [8]. In this case, we proposed and illustrated an iterative method of computing the trajectory set probabilities associated with the "success" level of a two-level accomplishment set. (See [2], pp. 43-47.) During the current reporting period, we have investigated extensions of this computational method to i) any "Cartesian" trajectory set, and ii) phased base models that are not necessarily stationary Markov processes.

3.4.1 Phased Models

Borrowing from the terminology of earlier work concerning "phased missions" (see [ ], for example), a model of a total system is phased if the observation times in the utilization period is finite, i.e.,

$$T = \{t_0, t_1, \ldots, t_k\}$$

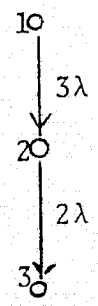
where $t_0 < t_1 < \ldots < t_k$. The interval $[t_{m-1}, t_m]$, $1 \leq m \leq k$, is referred to as the mth phase. Although phased models appear at the outset to be quite restricted, this is not the case, for

given a non-phased model, there often exists a phased model where performability is the same as that of the non-phased model. In general, system models having the same performability will be referred to as <u>equivalent models</u>. In other words, a total system model with capability function γ and base model probability function Pr is equivalent to a second model with capability function $\underline{\gamma}$ and probability function $\underline{Pr}$ if and only if for all accomplishment levels $a \in A$

$$Pr(\gamma^{-1}(a)) = \underline{Pr}(\underline{\gamma}^{-1}(a)).$$

Much of the traditional reliability analysis is facilitated by the fact that equivalent phased models (often single-phased) can be used to evaluate system reliability.

To illustrate this point, consider a typical continuous time Markov model of a TMR system (with a perfect voter) where the simplex system has failure rate λ, i.e., the Markov process $\underline{X}_S = \{\underline{X}_t | t \in T\}$ is represented by the graph



If the utilization period is $T = [t_0, t_1]$ and the accomplishment set is $A = \{a_0, a_1\}$ (where a_0 = success and a_1 = failure), then the capability function is given by:

$$\underline{\gamma}_S(u) = \begin{cases} a_0 & \text{if } u(t) \in \{1,2\},\ \forall\ t \in T \\ a_1 & \text{otherwise} \end{cases} .$$

Accordingly,

$$p_S(a_0) = \underline{Pr}(\underline{\gamma}_S^{-1}(a_0)) = \underline{Pr}(\{u|u(t) \in \{1,2\}, \forall\ t \in T\}).$$

However, since $\underline{\gamma}_S$ is structure based and the probability of entering a success state (1 or 2) from the failure state (3) is zero, there exists a one-phased model having the same performability. More precisely, consider the base model

$$X_S = \{\underline{X}_{t_0}, \underline{X}_{t_1}\}$$

i.e., the new base model is a pair of random variables describing the state of the original model at the beginning and the end of the utilization period. Furthermore, if we let γ_S be the function

$$\gamma_S(u) = \begin{cases} a_0 & \text{if } u(t_1) \in \{1,2\} \\ a_1 & \text{otherwise,} \end{cases}$$

then

$$\begin{aligned} p_S(a_0) &= Pr(\gamma_S^{-1}(a_0)) \\ &= Pr(X_{t_1} \in \{1,2\}) \\ &= \underline{Pr}(X_{t_1} \in \{1,2\}) \\ &= \underline{Pr}(\{u|u(t_1) \in \{1,2\}\}) \\ &= \underline{Pr}(\{u|\underline{\gamma}_S(u) = a_0\}) + \\ &\quad \underline{Pr}(\{u|\underline{\gamma}_S(u) \neq a_0 \text{ and } u(t_1) \in \{1,2\}\}). \end{aligned}$$

Since $\underline{\gamma}_S(u) \neq a_0$ and $u(t_1) \in \{1,2\}$ imply that $\underline{Pr}(\{u\}) = 0$,

$$\begin{aligned} p_S(a_0) &= Pr(\gamma_S^{-1}(a_0)) \\ &= \underline{Pr}(\{u|\underline{\gamma}_S(u) = a_0\}) \\ &= \underline{Pr}(\underline{\gamma}_S^{-1}(a_0)) \quad . \end{aligned}$$

Thus, the single phase model is equivalent to the original model

permitting performability (which, in this case, is reliability) to be computed in terms of the state of the system at the end of its utilization period.

Such single phase equivalents (or multiphase equivalents in the case of phased missions) exist whenever traditional reliability modeling assumptions are made with regard to the intra-phase processes. Accordingly, we have continued our investigation of phased model evaluation methods, where the results obtained during the current reporting period are discussed in the subsections that follow.

3.4.2 Performability Evaluation of Phased Models

Let S be a phased total system model (see Section 3.4.1) with base model X_S, state space $Q = \{q_1, q_2, \ldots, q_n\}$ and utilization period $T = \{t_0, t_1, \ldots, t_k\}$. Since the utilization period is finite, the trajectory space of S can be represented by $U = Q^k = \underbrace{Q \times \ldots \times Q}_{k \text{ times}}$ and the capability function is a function

$$\gamma : Q^k \to A$$

where k is the number of observation times. With regard to evaluating the probability of a trajectory set $\gamma^{-1}(a) \subseteq Q^k$, we have found that Cartesian trajectory sets are amenable to interactive methods of evaluation. Accordingly, by decomposing $\gamma^{-1}(a)$ into a finite number of disjoint Cartesian subsets, $Pr(\gamma^{-1}(a))$ can be evaluated in a straight-forward manner.

As defined in Section 3.3, a trajectory set $V \subseteq Q^k$ is Cartesian if $V = \mathop{X}_{i=1}^{k} \xi_i(V)$ where $\xi_i(V)$ is the projection of V onto the ith

coordinate. Note that the projection $\xi_i(V)$ provides one with a method for examining the state of the system at the ith observation time t_i with respect to the trajectory set V. Thus, the coordinatization system used here is temporal rather than the more general case (both temporal and spatial) discussed in Section 3.3.

As demonstrated in Section 3.3.2.2, Cartesian sets are characterized by the notion of R-independent coordinate sets. Thus, a test for determining whether a set is Cartesian is to determine R-dependencies between its coordinates.

For each $\xi_i(V)$, let $\mathbb{B}_i = \{u \in Q^k | \xi_i(u) \in \xi_i(V)\}$ be the set of all state trajectories in U that assume values in $\xi_i(V)$ at the ith observation time. Using the notation developed in the previous section, $\mathbb{B}_i$ can be expressed as

$$\mathbb{B}_i = \bigcup_{s \in \xi_i(V)} \mathbb{B}_i^V(s) \quad .$$

Moreover, the probability of $\mathbb{B}_i$ can be expressed as a one-dimensional distribution of the base model X_S, i.e.,

$$Pr(\mathbb{B}_i) = P(\{\omega | X_{t_i}(\omega) \in \xi_i(V)\})$$

(see Section 3.1).

When V is a Cartesian set, it is clear that V can be represented as the intersection of those rather elementary sets $\mathbb{B}_i$, i.e.,

$$\begin{aligned} V &= \mathop{\times}_{i=1}^{k} q_i(V) \\ &= \bigcap_{i=1}^{k} \mathbb{B}_i \quad . \end{aligned}$$

By iteratively applying the definition of conditional probability, it is also clear that

$$\Pr(V) = \Pr(\mathbf{B}_k \mid \bigcap_{i=1}^{k-1} \mathbf{B}_i) \quad \Pr(\mathbf{B}_{k-1} \mid \bigcap_{i=1}^{k-2} \mathbf{B}_i)$$

$$\ldots \quad \Pr(\mathbf{B}_2 | \mathbf{B}_1) \Pr(\mathbf{B}_1) \quad .$$

Since each term in the product involves only elementary sets $\mathbf{B}_i$, we show in the following discussion (see equation 3.4.1) that Pr(V) can be determined iteratively using matrix multiplications.

Without loss of generality, we suppose that the initial time $t_0 = 0$ and we let I(0) denote the initial state distribution for the base model, that is,

$$I(0) = [p_1(0), \ldots, p_n(0)]$$

where $P_i(0) = \Pr[X_{t_0} = q_i]$, $1 \leqslant i \leqslant n$. Let P(m) be the state transition matrix of the ith phase of the base model X_S, i.e.,

$$P(m) = [P_{ij}(m)]$$

where $P_{ij}(m) = \Pr(X_{t_m} = q_j | X_{t_{m-1}} = q_i)$.

For each phase m ($1 \leqslant m < k$), let G(m) denote the <u>characteristic matrix</u> of the mth phase, i.e., $G(m) = [g_{ij}(m)]$ where

$$g_{ij}(m) = \begin{cases} 1 \text{ if } i = j \text{ and } q_i \in \xi_m(V) \\ 0 \text{ otherwise} \end{cases} .$$

For the final phase, we define a <u>characteristic vector</u>

$$F(k) = \begin{bmatrix} f_1(k) \\ \vdots \\ f_n(k) \end{bmatrix}$$

where

$$f_i(k) = \begin{cases} 1 \text{ if } q_i \in \xi_k(V) \\ 0 \text{ otherwise} \end{cases} .$$

Then as a special case of the more general formula proved in Theorem 3, the probability of the Cartesian set V can be formulated as:

$$Pr(V) = I(0) \left[\prod_{i=1}^{k-1} P(i)G(i) \right] P(k)F(k). \qquad (3.4.1)$$

Given the above result concerning Cartesian sets, an important step in evaluating $Pr(\gamma^{-1}(a))$ is to express $\gamma^{-1}(a)$ in terms of Cartesian components. Thus, if $\gamma^{-1}(a) = \bigcup_{i=1}^{m} V_i$ where $\{V_i | i = 1,2,\ldots,m\}$ are Cartesian sets and $V_i \cap V_j = \emptyset$ if $i \neq j$, then

$$Pr(\gamma^{-1}(a)) = \sum_{i=1}^{m} Pr(V_i)$$

and hence performability can be calculated by summing the probabilities of Cartesian sets. The existence of the set $\{V_i | i = 1,2,\ldots,m\}$ can be shown as follows. Since each singleton set $\{u \in U\}$ is a Cartesian set, by definition, $\gamma^{-1}(a) = \bigcup_{u \in \gamma^{-1}(a)} \{u\}$ satisfies the above conditions. However, in practical situations where $\gamma^{-1}(a)$ is very large each singleton set will have negligible probability and the cumulative error resulting from the sum of a large number of single probabilities will generally be intolerable. To avoid this enumeration approach, we have developed a method (see Section 3.5.4.2) for determining $\{V_i | i = 1,\ldots,m\}$ in a systematic manner, using a hierarchical formulation of the capability function.

3.4.3 Simplification of Phased Models

Let S be a phased total system model with base model

$$X_S = \{X_{t_i} | i = 0,1,\ldots,k\}$$

where $t_0 < t_1 < \ldots < t_k$ and where each random variable X_{t_i} takes values in the state space

$$Q = \{q_1, q_2, \ldots, q_n\} .$$

Since the phased base model X_S may have been derived from a larger equivalent model $\underline{X}_S$ (e.g., a continuous-time Markov model), the state space Q may be much larger than needed to distinguish accomplishment levels via the capability function of the phased model. Accordingly, we have continued to pursue our investigation of state "lumping" methods which can further simplify the evaluation of state trajectory set probabilities. (See [12], for example, where a Markov model with 146 states is reduced to a model with 11 states.) In particular, we have conducted a more detailed study of "Michigan lumping" wherein different lumping relations can be associated with different phases of the phased model.

In general, we define the <u>lumping relation of phase m</u> $(1 \leqslant m \leqslant k)$ to be an equivalence relation $\equiv_m$ on the state space Q. The partition of $\equiv_m$ is denoted

$$Q_m = \{m1, m2, \ldots, mb_m\}$$

where b_m is the number of equivalence classes (lumps) of the lumping relation $\equiv_m$. Each equivalence class mi is a subset of the state space Q where, if $r \in mi$, then

$$mi = \{q \in Q | q \equiv_m r\} .$$

To illustrate, suppose S is a triplicated system with state space $Q = \{0,1\}^3$, where $q = (0,0,0)$ means all three subsystems are fault-free and, at the other extreme, $q = (1,1,1)$ says that all three subsystems are faulty. Supposing further that there is only one phase with the lumping relation

$q \equiv_1 r$ if q and r represent the same number of faulty subsystems

then the corresponding partition (lumping) is given by:

$$Q_1 = \{11,12,13,14\}$$

where

$$11 = \{(0,0,0)\}$$

$$12 = \{(1,0,0),(0,1,0),(0,0,1)\}$$

$$13 = \{(1,1,0),(1,0,1),(0,1,1)\}$$

$$14 = \{(1,1,1)\} .$$

In general, given m lumping relations, one for each phase, we can associate a <u>lumped</u> base model $\overline{X}_S$ with the original phased base model X_S by defining $\overline{X}_S$ as follows:

$$\overline{X}_S = \{\overline{X}_m | m = 0,1,\ldots,k\}$$

where if $m = 0$ then

$$\overline{X}_0 = 1i \text{ if } X_{t_0} \in 1i \ (1 \leqslant i \leqslant b_1)$$

and if $1 < m \leqslant k$ then

$$\overline{X}_m = mi \text{ if } X_{t_m} \in mi \ (1 \leqslant i \leqslant b_m) .$$

(The variables X_{t_m}, $0 \leqslant m \leqslant k$, are the random variables of the phased base model X_S.) For a lumped model to be useful, it must be compatible with the capability function γ in the sense that system performance can be determined knowing the state trajectory of $\overline{X}_S$ at times $t_1,t_2,\ldots,t_k$. More precisely, if U is the trajectory space of the phased model then two trajectories $u, u' \in U$ are <u>equivalent</u> (denoted $u \equiv u'$) if $u(t_m) \equiv_m u'(t_m)$, for $m = 1,2,\ldots,k$. Then we require that the lumping relations $\equiv_m$ be such that

$$u \equiv u' \text{ implies } \gamma(u) = \gamma(u')$$

for all u, u' ε U. Under this condition, the trajectory space $\overline{U}$ of $\overline{X}$ can be effectively regarded as the space

$$\overline{U} = Q_1 \times Q_2 \times \ldots \times Q_k$$

and the induced capability function $\overline{\gamma}:\overline{U} \to A$ is given by

$$\overline{\gamma}(1i_1, 2i_2, \ldots, ki_k) = \gamma(u)$$

where u is any trajectory in U such that $u(t_m) \varepsilon mi_m$, m = 1,2,...,k. Finally, if $V \subseteq \overline{U}$, the induced probability function $\overline{Pr}$ of the lumped model is given by

$$\overline{Pr}(V) = Pr(W)$$

where

$$W = \left\{ u \varepsilon U \;\middle|\; \begin{array}{l} \text{for some } (1i_1, 2i_2, \ldots, ki_k) \varepsilon V, \\ u(t_m) \varepsilon mi_m,\ m = 1,2,\ldots,k \end{array} \right\}.$$

In particular, it follows that

$$\overline{Pr}(\overline{X}_m = mi) = Pr(X_{t_m} \varepsilon mi) \qquad (3.4.2)$$

and, more generally, that

$$\overline{Pr}(\overline{X}_1 = 1i_1, \ldots, \overline{X}_k = ki_k) = Pr(X_{t_1} \varepsilon 1i_1, \ldots, X_{t_k} \varepsilon ki_k) \qquad (3.4.3$$

Given the above definitions of $\overline{\gamma}$ and $\overline{Pr}$, it is easily verified that the lumped model is equivalent to the original phased model. Although the probability function $\overline{Pr}$ is well defined for arbitrary lumping relations, the lumpings may be such that $\overline{Pr}$ is very difficult to evaluate, due to the fact that lumping does not, in general, preserve special stochastic properties. For example, if the phased base model X_S is a stationary Markov process, a lumped model $\overline{X}_S$ is generally neither stationary nor Markovian. This problem is addressed in the subsections that follow, beginning with the case

where the lumpings are unrestricted.

We suppose first, as in the previous subsection, that the structure of the trajectory set in question is Cartesian, that is, $V \subseteq \bar{U}$ where

$$V = \mathop{X}_{i=1}^{k} \xi_i(V) ,$$

or, alternatively, letting $R_i = \xi_i(V)$,

$$V = R_1 \times R_2 \times \ldots \times R_k .$$

Then the objectsof study are a) the probability

$$\overline{Pr}(V) = \overline{Pr}(\bar{X}_1 \varepsilon R_1, \bar{X}_2 \varepsilon R_2, \ldots, \bar{X}_k \varepsilon R_k) .$$

and b) its formulation in terms of the probability function Pr of the (unlumped) phased model.

We begin by considering the more restricted problem, that of evaluating the one-dimensional probabilities

$$\overline{Pr}(\bar{X}_\ell \varepsilon R_\ell) , \quad \ell = 1,2,\ldots,k .$$

Relative to the mth phase of the phased model, define

$$p(m) = [p_{ij}(m)]$$

where

$$p_{ij}(m) = Pr(X_{t_m} \varepsilon mj | X_{t_{m-1}} \varepsilon mi) ,$$

i.e., the probability of being in lumped state mj at the mth observation time given that the phased model state is in lump mi at the beginning of the mth phase. Thus p(m) is the <u>initial to final state transition matrix</u> of the mth phase. For all but the final phase define

$$H(m) = [h_{ij}(m)]$$

where

$$h_{ij}(m) = Pr(X_{t_m} \varepsilon (m+1)j | X_{t_m} \varepsilon mi) ,$$

i.e., the probability of being in state $(m + 1)j$ at the beginning of the $m + 1^{th}$ phase given that the lumped model is in state mi at the mth observation time. Thus $H(m)$ is the <u>interphase transition matrix</u> between phase m and phase m + 1. Note that the above matrices are definable beginning with an arbitrary process X_S.

Let $I(0)$ be the initial state probability distribution of the lumped model $\overline{X}_S$, i.e.,

$$I(0) = [p_1, p_2, \ldots, p_{b_1}]$$

where

$$p_i = Pr(X_{t_0} \varepsilon 1i) = \overline{P}r(\overline{X}_0 = 1i)$$

Let $J(\ell)$ be the state probability distribution of $\overline{X}_S$ at the end of phase ℓ, i.e.,

$$J(\ell) = [r_1, \ldots, r_{b_\ell}]$$

where

$$r_i = \overline{P}r(\overline{X}_\ell = \ell i)$$

is the probability of being in state ℓi at the ℓth observation time. Then

<u>Theorem 1</u>: $$J(\ell) = I(0)\left[\prod_{m=1}^{\ell-1} P(m)H(m)\right] P(\ell) \qquad (3.4.4).$$

<u>Proof</u>: We prove this by induction. For $\ell = 1$,

$$J(1) = I(0)P(1) = [a_1, \ldots, a_j, \ldots, a_{b_1}]$$

where

$$a_j = \sum_{i=1}^{b_1} Pr(X_{t_0} \varepsilon 1i) \cdot Pr(X_{t_1} \varepsilon 1j | X_{t_0} \varepsilon 1i)$$

$$= \sum_{i=1}^{b_1} \Pr(X_{t_1} \in 1j,\ X_{t_0} \in 1i)$$

$$= \Pr(X_{t_1} \in 1j) = \overline{\Pr}(\overline{X}_1 = 1j) \quad .$$

Suppose that the formula holds for ℓ, $1 \leqslant \ell < k$, we have to show that

$$J(\ell + 1) = I(0)\left[\prod_{m=1}^{\ell} P(m)H(m)\right]P(\ell+1) \ .$$

When multiplied by $H(\ell)P(\ell+1)$ on both sides, the equation for $J(\ell)$ becomes

$$J(\ell)H(\ell)P(\ell) = I(0)\left[\prod_{m=1}^{\ell} p(m)H(m)\right] P(\ell+1) \quad .$$

When we iteratively compute the matrix product on the left hand side, beginning from the left, then the first two terms become

$$J(\ell)H(\ell) = [c_1,\ldots,c_j,\ldots,c_{b_{\ell+1}}]$$

where

$$c_j = \sum_{i=1}^{b_\ell} \overline{\Pr}(\overline{X}_\ell = \ell i)\cdot \Pr(X_{t_\ell} \in (\ell+1)j \,|\, X_{t_\ell} \in \ell i)$$

$$= \sum_{i=1}^{b_\ell} \Pr(X_{t_\ell} \in \ell i)\cdot \Pr(X_{t_\ell} \in (\ell+1)j \,|\, X_{t_\ell} \in \ell i)$$

$$= \sum_{i=1}^{b_\ell} \Pr(X_{t_\ell} \in (\ell+1)j \cap \ell i)$$

$$= \Pr(X_{t_\ell} \in (\ell+1)j)$$

is the probability of being in state $(\ell+1)j$ at the beginning of the $\ell+1^{th}$ phase.

Finally, multiplying the product by the $\ell+1^{th}$ phase transition matrix,

$$J(\ell)H(\ell)P(\ell+1) = [d_1,\ldots,d_j,\ldots,d_{b_{\ell+1}}]$$

where

$$d_j = \sum_{i=1}^{b_{\ell+1}} \Pr(X_{t_\ell} \in (\ell+1)i) \cdot \Pr(X_{t_{\ell+1}} \in (\ell+1)j \,|\, X_{t_\ell} \in (\ell+1)i)$$

$$= \sum_{i=1}^{b_{\ell+1}} \Pr(X_{t_{\ell+1}} \in (\ell+1)j \;,\; X_{t_\ell} \in (\ell+1)i)$$

$$= \Pr(X_{t_{\ell+1}} \in (\ell+1)j) = \overline{\Pr}(\overline{X}_{\ell+1} = (\ell+1)j) \quad .$$

Thus

$$J(\ell+1) = J(\ell)H(\ell)P(\ell+1)$$

$$= I(0) \left[\prod_{m=1}^{\ell} P(m)H(m) \right] P(\ell+1)$$

which completes the proof.

If we compare equation 3.4.4 with the formula on page 44 of the Second Semi-Annual Status Report, we note that it does not involve the G and F matrices. It is used solely to compute the probability (mass) function of the random variable $\overline{X}_\ell$ (the state of the lumped process at the ℓth observation time). Also, it is important to note that this formula applies to an arbitrary phased base model X_S and, in particular, the lumped process $\overline{X}_S$ need not be Markov.

Let us now consider the problem addressed at the outset of this subsection, i.e., the probability evaluation of a Cartesian trajectory set

$$V = R_1 \times R_2 \times \ldots \times R_k \subseteq \overline{U}$$

where

$$\overline{Pr}(V) = \overline{Pr}(\overline{X}_1 \in R_1, \overline{X}_2 \in R_2, \ldots, \overline{X}_k \in R_k).$$

Extending the G and F matrices of the previous subsection to the phased base model $\overline{X}_S$, let G(m) denote the characteristic matrix of the mth phase $(1 \leqslant m < k)$, i.e.,

$$G(m) = [g_{ij}(m)]$$

where

$$g_{ij}(m) = \begin{cases} 1 \text{ if } i=j \text{ and } mi \in R_m \\ 0 \text{ otherwise} , \end{cases}$$

and for the final phase (m=k) we define a characteristic vector

$$F(k) = \begin{bmatrix} f_1(k) \\ \vdots \\ f_{b_k}(k) \end{bmatrix}$$

where

$$f_i(k) = \begin{cases} 1 \text{ if } ki \in R_k \\ 0 \text{ otherwise} . \end{cases}$$

We first prove a lemma which is a generalization of the iterative formula

$$\overline{Pr}(\overline{X}_1 \in R_1, \overline{X}_2 \in R_2, \ldots, \overline{X}_k \in R_k)$$

$$= I(0) \left[\prod_{m=1}^{k-1} P(m)G(m)H(m) \right] P(k)F(k) . \qquad (3.4.5)$$

For each R_ℓ, $\ell = 1,2,\ldots,k$, let UR_ℓ be the union of all the equivalence classes contained in R_ℓ, i.e.,

$$UR_\ell = \bigcup_{\ell i \in R_\ell} \ell i .$$

For each phase m, except the final phase, define

$$K(m) = [k_{ij}(m)], \quad m = 1,2,\ldots,k-1,$$

where

$$k_{ij}(m) = Pr(X_{t_m} \varepsilon (m+1)j | X_{t_m} \varepsilon mi, X_{t_{m-1}} \varepsilon UR_{m-1}, \ldots, X_{t_1} \varepsilon UR_1).$$

The matrix K(m) is similar to the interphase transition matrix H(m) except that the interphase transition probabilities are now conditioned by the first m-1 components of the Cartesian set V. Hence K generally depends on V while H does not. (Conditions under which K can be identified with H are a subject of later discussion.)

To compute $\overline{Pr}(V)$ in terms of the matrices P(m), G(m), F(m) and K(m), we assume further that the lumping relations $\equiv_m$ are compatible with the phased model X_S to the extent that transition probabilities are invariant over the states in a lump. More precisely, we say that X_S is <u>strongly lumpable with respect to $\equiv_m$</u> if for all mi, mj ε Q_m, the probabilities

$$Pr(X_{t_m} \varepsilon mj | X_{t_{m-1}} = q)$$

are the same for all q ε mi. A lumped model $\overline{X}_S$ is <u>strongly lumped</u> if X_S is strongly lumpable with respect to all $\equiv_m$, m = 1,2,...,k.

Although we refer to such lumping as "strong", it can be shown that the usual type of stationary Markov chain lumping (i.e., where the Markov property is preserved relative to all initial state distributions) is strong in the above sense (see, for example, [14], p. 124). In particular, all the work we have seen concerning Markov model simplification for reliability

analysis has utilized strong lumping. Thus, strong lumping, as defined above, is not a severe constraint. Indeed, our concept of a strongly lumped process $\overline{X}_S$ is weaker than the usual type of strongly lumped process which presumes the use of a single lumping relation throughout the utilization period.

In terms of the above concepts we are able to prove the following important lemma.

<u>Lemma 1</u>: If $\overline{X}_S$ is strongly lumped then

$$\overline{P}r(X_1 \varepsilon R_1,\ldots,X_k \varepsilon R_k) = I(0)\left[\prod_{m=1}^{k-1} P(m)G(m)K(m)\right]P(k)F(k). \quad (3.4.6)$$

<u>Proof</u>: We show this by induction. When $k = 1$,

$$I(0)P(1)F(1) = \sum_{1j\varepsilon R_1} \overline{P}r(\overline{X}_1 = 1j) = \overline{P}r(\overline{X}_1 \varepsilon R_1).$$

Suppose that the formula holds for $k = \ell$, that is

$$\overline{P}r(\overline{X}_1 \varepsilon R_1,\overline{X}_2 \varepsilon R_2,\ldots,\overline{X}_\ell \varepsilon R_\ell)$$
$$= I(0)\left[\prod_{m=1}^{\ell-1} P(m)G(m)K(m)\right]P(\ell)F(\ell).$$

Then

$$I(0)\left[\prod_{m=1}^{\ell} P(m)G(m)K(m)\right]P(\ell + 1)F(\ell + 1)$$

$$= \left[I(0)\left(\prod_{m=1}^{\ell-1} P(m)G(m)K(m)\right) P(\ell)G(\ell)\right]K(\ell)P(\ell+1)F(\ell+1)$$

$$= A_1K(\ell)P(\ell+1)F(\ell+1)$$

where

$$A_1 = [a_1,\ldots,a_j,\ldots,a_{b_\ell}]$$

and

$$a_j = \begin{cases} \Pr(\overline{X}_1 \in R_1,\ldots,\overline{X}_{\ell-1} \in R_{\ell-1}, \overline{X}_\ell = \ell j) \text{ if } \ell j \in R_\ell \\ 0 \text{ otherwise} , \end{cases}$$

by applying the equation for $k = \ell$.

When we iteratively compute the matrix product, beginning from the left, then the first two terms become

$$A_2 = A_1 K(\ell) = [c_1,\ldots,c_j,\ldots,c_{b_{\ell+1}}]$$

where

$$c_j = \sum_{i=1}^{b_\ell} a_i \, k_{ij}(\ell)$$

$$= \sum_{\ell i \in R_\ell} \Pr(\overline{X}_1 \in R_1,\ldots,\overline{X}_{\ell-1} \in R_{\ell-1}, \overline{X}_\ell = \ell i) \cdot$$

$$\Pr(X_{t_\ell} \in (\ell+1)j \mid X_{t_\ell} \in \ell i, X_{t_{\ell-1}} \in UR_{\ell-1},\ldots,X_{t_1} \in UR_1)$$

$$= \sum_{\ell i \in R_\ell} \Pr(X_{t_1} \in UR_1,\ldots,X_{t_{\ell-1}} \in UR_{\ell-1}, X_{t_\ell} \in \ell i, X_{t_\ell} \in (\ell+1)j)$$

$$= \Pr(X_{t_1} \in UR_1,\ldots,X_{t_\ell} \in UR_\ell, X_{t_\ell} \in (\ell+1)j) .$$

The next partial product is the result of multiplying A_2 by the transition matrix $P(\ell+1)$ which yields:

$$A_3 = A_2(P(\ell+1) = [d_1,\ldots,d_j,\ldots,d_{b_{\ell+1}}]$$

where

$$d_j = \sum_{i=1}^{b_{\ell+1}} c_i P_{ij}(\ell+1)$$

$$= \sum_{i=1}^{b_{\ell+1}} c_i \Pr(X_{t_{\ell+1}} \in (\ell+1)j \mid X_{t_\ell} \in (\ell+1)i) .$$

Since X_S is strongly lumpable with respect to the lumping relation $\equiv_{\ell+1}$,

$$\Pr(X_{t_{\ell+1}} = (\ell+1)j \mid X_{t_\ell} = q)$$

is the same for every $q \in (\ell+1)i$. Let p denote this common probability and consider the events

$$A = X_{t_{\ell+1}} \in (\ell+1)j$$

$$B = X_{t_{\ell+1}} \in (\ell+1)i,\ X_{t_\ell} \in UR_{\ell+1}, \ldots, X_{t_1} \in UR_1$$

$$C = X_{t_\ell} \in (\ell+1)i\ .$$

Then, since $B \subseteq C$, there is a subset R of the lumped state $(\ell+1)i$ such that

$$B = X_{t_\ell} \in R\ .$$

Accordingly,

$$\Pr(A|B) = \frac{\Pr(AB)}{\Pr(B)}$$

where

$$\Pr(AB) = \sum_{q \in R} \Pr(A|X_{t_\ell} = q)\Pr(X_{t_\ell} = q)\ .$$

Since $\Pr(A|X_{t_\ell} = q) = p$ for all $q \in R$,

$$\Pr(AB) = \sum_{q \in R} p \cdot \Pr(X_{t_\ell} = q)$$

$$= p \cdot \sum_{q \in R} \Pr(X_{t_\ell} = q)$$

$$= p \cdot \Pr(B)\ .$$

Accordingly

$$\Pr(A|B) = \frac{p \cdot \Pr(B)}{\Pr(B)} = p \ .$$

In particular, when B = C

$$\Pr(A|C) = p$$

and, hence

$$\Pr(A|C) = \Pr(A|B) \ .$$

In other words,

$$\Pr(X_{t_{\ell+1}} \varepsilon (\ell+1)j | X_{t_\ell} \varepsilon (\ell+1)i) = \Pr(X_{t_{\ell+1}} \varepsilon (\ell+1)j | X_{t_\ell} \varepsilon (\ell+1)i,$$

$$X_{t_\ell} \varepsilon UR_\ell, \ldots, X_{t_1} \varepsilon UR_1) \ .$$

Strong lumpability therefore allows us to forget the past history when determining the intraphase transition probabilities. Accordingly, by replacing $\Pr(A|C)$ with $\Pr(A|B)$ in d_j,

$$d_j = \sum_{i=1}^{b_{\ell+1}} \Pr(X_{t_\ell} \varepsilon (\ell+1)i, X_{t_\ell} \varepsilon UR_\ell, \ldots, X_{t_1} \varepsilon UR_1)$$

$$\Pr(X_{t_{\ell+1}} \varepsilon (\ell+1)j | X_{t_\ell} \varepsilon (\ell+1)i, X_{t_\ell} \varepsilon UR_\ell, \ldots, X_{t_1} \varepsilon UR_1)$$

$$= \sum_{i=1}^{b_{\ell+1}} \Pr(X_{t_{\ell+1}} \varepsilon (\ell+1)j, X_{t_\ell} \varepsilon (\ell+1)i, X_{t_\ell} \varepsilon UR_\ell, \ldots, X_{t_1} \varepsilon UR_1)$$

$$= \Pr(X_{t_{\ell+1}} \varepsilon (\ell+1)j, X_{t_\ell} \varepsilon UR_\ell, \ldots, X_{t_1} \varepsilon UR_1)$$

$$= \overline{\Pr}(\overline{X}_1 \varepsilon R_1, \overline{X}_2 \varepsilon R_2, \ldots, \overline{X}_\ell \varepsilon R_\ell, \overline{X}_{\ell+1} = (\ell+1)j) \ .$$

The product is completed by multiplying A_3 by the characteristic vector $F(\ell+1)$ of the final phase, that is,

$$I(0)\left[\prod_{m=1}^{\ell} P(m)G(m)K(m)\right]P(\ell+1)F(\ell+1)$$

$$= A_3F(\ell+1)$$

$$= \sum_{(\ell+1)j\varepsilon R_{\ell+1}} \overline{Pr}(\overline{X}_1 \varepsilon R_1, \overline{X}_2 \varepsilon R_2 \ldots, \overline{X}_\ell \varepsilon R_\ell, \overline{X}_{\ell+1} = (\ell+1)j)$$

$$= \overline{Pr}(\overline{X}_1 \varepsilon R_1, \overline{X}_2 \varepsilon R_2, \ldots, \overline{X}_{\ell+1} \varepsilon R_{\ell+1}) .$$

Thus, equation 3.4.6 holds for all $\ell \leq k$, which completes the proof of Lemma 1.

Note that in proving the lemma, we did not use the assumption that X_S is strongly lumpable with respect to $\equiv_1$ and hence we can relax the hypothesis and require only that X_S be strongly lumpable with respect to $\equiv_m$, $m = 2,3,\ldots,k$. However, in order to simplify the calculation of the transition probability matrix P(1) associated with the first phase, it is convenient to assume that X_S is strongly lumpable for all phases. This remark applies as well to the subsequent results concerning the evaluation of $\overline{Pr}(V)$.

Although Lemma 1 provides us with relatively unrestricted closed form formulation of $\overline{Pr}(V)$, its disadvantages derive from the fact that the K(m) matrices may be difficult to obtain in practical applications. In particular, K(m) will generally depend on V as well as X_S and, moreover, will generally depend on the history of X_S prior to phase m. The latter objection disappears when the lumping relations are such that

$$Pr(X_{t_m} \varepsilon (m+1)j | X_{t_m} \varepsilon mi, X_{t_{m-1}} \varepsilon UR_{m-1}, \ldots, X_{t_1} \varepsilon UR_1) =$$

$$Pr(X_{t_m} \varepsilon (m+1)j | X_{t_m} \varepsilon mi) \qquad (3.4.7)$$

for all $(m+1)j \varepsilon Q_{m+1}$ and $mi \varepsilon Q_m$.

Recalling the definitions of K(m) and H(m), the preceding condition is just the condition which guarantees that K(m) = H(m). Accordingly, we obtain the following specialization of Lemma 1.

<u>Theorem 2</u>: If $\overline{X}_S$ is strongly lumped, $V = R_1 \times R_2 \times \ldots \times R_k$ and equation (3.4.7) holds for m = 1,2,...,k-1, then

$$\overline{P}r(V) = I(0)\left[\prod_{m=1}^{k-1} P(m)G(m)H(m)\right]P(k)F(k) \ . \qquad (3.4.8)$$

Since equation (3.4.7) depends on the specific nature of the Cartesian set V, for a fixed strongly lumped model $\overline{X}_S$, the hypothesis of Theorem 2 may hold for certain trajectory sets V but not for others. Accordingly, we have sought to identify even stronger conditions under which equation 3.4.8 will hold for arbitrary Cartesian trajectory sets.

<u>Lemma 2</u>: If $\overline{X}_S$ is strongly lumped then equation 3.4.7 holds for all Cartesian sets V and for all phases m, if and only if

$$Pr(X_{t_m} \varepsilon (m+1)i_{m+1} | X_{t_m} \varepsilon mi_m, X_{t_{m-1}} \varepsilon (m-1)i_{m-1}, \ldots, X_{t_1} \varepsilon 1i_1)$$

$$= Pr(X_{t_m} \varepsilon (m+1)i_{m+1} | X_{t_m} \varepsilon mi_m) \qquad (3.4.9)$$

for all m = 1,2,...,k-1, and for all $(m+1)i_{m+1} \varepsilon Q_{m+1}, mi_m \varepsilon Q_m$.

<u>Proof</u>: Suppose equation 3.4.7 holds for all Cartesian sets $V = \mathop{X}_{i=1}^{k} Ri$, then by taking R_i to be the singleton set $\{\ell i_\ell\}$, $\ell = 1,2,\ldots,m+1$,

$$Pr(X_{t_m} \varepsilon (m+1)i_{m+1} | X_{t_m} \varepsilon mi_m, X_{t_{m-1}} \varepsilon (m-1)i_{m-1}, \ldots, X_{t_1} \varepsilon 1i_1)$$

$$= Pr(X_{t_m} \varepsilon (m+1)i_{m+1} | X_{t_m} \varepsilon mi_m) \ .$$

Conversely, when equation 3.4.9 holds for every $\ell i_\ell \in Q_\ell$, $\ell = 1,2,\ldots,m+1$, then

$$\Pr(X_{t_m} \in (m+1)j \mid X_{t_m} \in mi_m, X_{t_{m-1}} \in UR_{m-1},\ldots,X_{t_1} \in UR_1)$$

$$= \sum_{\substack{\ell i_\ell \in R_\ell \\ \ell=1,\ldots,m}} \Pr(X_{t_m} \in (m+1)j \mid X_{t_m} \in mi_m, X_{t_{m-1}} \in (m-1)i_{m-1},\ldots,$$

$$X_{t_1} \in 1i_1) \cdot \frac{\Pr(X_{t_{m-1}} \in (m-1)i_{m-1},\ldots,X_{t_1} \in 1i_1)}{\Pr(X_{t_{m-1}} \in UR_{m-1},\ldots,X_{t_1} \in UR_1}$$

$$= \Pr(X_{t_m} \in (m+1)j \mid X_{t_m} \in mi) \cdot$$

$$\sum_{\substack{\ell i_\ell \in R_\ell \\ \ell=1,\ldots,m}} \frac{\Pr(X_{t_{m-1}} \in (m-1)i_{m-1},\ldots,X_{t_1} \in 1i_1)}{\Pr(X_{t_{m-1}} \in UR_{m-1},\ldots,X_{t_1} \in UR_1}$$

$$= \Pr(X_{t_m} \in (m+i)j \mid X_{t_m} \in mi) \cdot 1 = \Pr(X_{t_m} \in (m+1)j \mid X_{t_m} \in mi)$$

which shows equation 3.4.7.

Combining Lemma 2 with Theorem 2, we obtain the following result:

<u>Theorem 3</u>: If $\overline{X}_S$ is strongly lumped and equation 3.4.9 holds for each phase m, m=1,...,k-1, then for any Cartesian trajectory set V

$$\overline{\Pr}(V) = I(0)\left[\prod_{m=1}^{k-1} P(m)G(m)H(m)\right]P(k)F(k) \ .$$

Under the conditions of Theorem 3, we observe that $\overline{X}_S$ is a Markov process (but not necessarily stationary). This can be demonstrated as follows.

$$\overline{Pr}(\overline{X}_{m+1} = (m+1)j \,|\, \overline{X}_m = mi_m, \overline{X}_{m-1} = (m-1)i_{m-1}, \ldots, \overline{X}_1 = 1i_1)$$

$$= Pr(X_{t_{m+1}} \varepsilon (m+1)j \,|\, X_{t_m} \varepsilon mi_m, X_{t_{m-1}} \varepsilon (m-1)i_{m-1}, \ldots X_{t_1} \varepsilon 1i_1)$$

$$= \sum_{(m+1)i \varepsilon Q_{m+1}} Pr(X_{t_{m+1}} \varepsilon (m+1)j \,|\, X_{t_m} \varepsilon (m+1)i, X_{t_m} \varepsilon mi_m, \ldots,$$

$$X_{t_1} \varepsilon 1i_1) \cdot Pr(X_{t_m} \varepsilon (m+1)i \,|\, X_{t_m} \varepsilon mi_m, \ldots, X_{t_1} \varepsilon 1i_1)$$

$$= \sum_{(m+1)i \varepsilon Q_{m+1}} Pr(X_{t_{m+1}} \varepsilon (m+1)j \,|\, X_{t_m} \varepsilon (m+1)i)$$

$$\cdot Pr(X_{t_m} \varepsilon (m+1)i \,|\, X_{t_m} \varepsilon mi_m)$$

$$= Pr(X_{t_{m+1}} \varepsilon (m+1)j \,|\, X_{t_m} \varepsilon mi_m)$$

$$= \overline{Pr}(\overline{X}_{m+1} = (m+1)i \,|\, \overline{X}_m = mi_m)$$

Hence, $\overline{X}_S$ satisfies the Markov property.

Moreover, if we extend the definition of the interphase transition matrices so that H(0) is the identity matrix, i.e.,

$$H(0) = [h_{ij}(0)]$$

where

$$h_{ij}(0) = \begin{cases} 1 \text{ if } i=j \\ 0 \text{ otherwise,} \end{cases}$$

then the transition probabilities of $\overline{X}_S$ associated with phase m can be expressed as a matrix

$$\overline{P}(m) = (\overline{p}_{ij}(m))$$

where

$$\begin{aligned}\overline{p}_{ij}(m) &= \overline{Pr}(\overline{X}_m = mj \mid \overline{X}_{m-1} = (m-1)) \\ &= Pr(X_{t_m} \varepsilon\ mj \mid X_{t_{m-1}} \varepsilon\ (m-1)i) \\ &= \sum_{m\ell \varepsilon Q_m} Pr(X_{t_m} \varepsilon\ mj \mid X_{t_{m-1}} \varepsilon\ m\ell) \cdot Pr(X_{t_{m-1}} \varepsilon\ m\ell \mid X_{t_{m-1}} \varepsilon (m-1)i) \\ &= \sum_{\ell=1}^{b_m} p_{\ell j}(m) h_{i\ell}(m-1) \ .\end{aligned}$$

Accordingly,

$$\overline{P}(m) = H(m-1)P(m)$$

and equation 3.4.8 can be represented in a more convenient form:

$$\overline{Pr}(V) = I(0)\left[\prod_{m=1}^{k-1} \overline{P}(m)G(m)\right]\overline{P}(k)F(k) \quad . \qquad (3.4.10)$$

Since $h_{ij}(m)$ generally depends on the observation time t_m even when X_S is stationary, the transition probabilities $\overline{p}_{ij}(m)$ may not be the same for different phases. Hence $\overline{X}_S$ is a time varying Markov process.

Although Theorem 3 provides us with a formula for evaluating the probability of an arbitrary Cartesian trajectory set V, it has the disadvantage that equation 3.4.9 has to be verified with respect to all possible sequences of lumped states $X_{t_m} \varepsilon\ mi_m$ where $mi_m \varepsilon\ Q_m$, $m = 1,2,\ldots,k$. Thus in order to further simplify the computation, we have identified the following stronger condition.

By applying arguments similar to the proof of Lemma 1, we show that equation 3.4.9 holds when the probabilities

$$Pr(X_{t_m} \varepsilon\ (m+1)i_{m+1} \mid X_{t_m} = q)$$

are the same for all $q \varepsilon\ mi_m$. Hence, we obtain the following important result.

<u>Theorem 4</u>: If $\overline{X}_S$ is strongly lumped and for all $m \in \{1,2,\ldots,m-1\}$, $(m+1)j \in Q_{m+1}$, and $mi \in Q_m$ the probabilities

$$\Pr(X_{t_m} \in (m+1)j \mid X_{t_m} = q)$$

are the same for all $q \in mi$, then

$$\overline{\Pr}(V) = I(0)\left[\prod_{m=1}^{k-1} P(m)G(m)H(m)\right]P(k)F(k)$$

for all Cartesian trajectory sets V.

Theorems 2-4 tell us, under successively more stringent conditions, how the probability of a Cartesian set V may be iteratively computed from knowledge of the intraphase processes (the P matrices), the interphase transitions (the K or H matrices), and the set V (the G and F matrices). Under the conditions of Theorem 4, the P and H matrices are relatively easy to obtain. Under the weaker conditions of Theorem 3, it appears difficult to determine whether these conditions are indeed satisfied, although we have not as yet had enough experience with such calculations to judge the extent of the difficulty. A similar comment applies to the even weaker conditions of Theorem 2. However, we do believe that the theory developed above demonstrates the feasibility of "Michigan lumping", i.e., lumping a phased model according to the computational requirements of each phase as opposed to "homogeneous lumping" which uses the same lumping relation throughout the utilization interval.

During the next reporting period we intend to more fully explore the practical implications of these lumping methods by

experimenting with various types of base models and various types of Cartesian sets. We also wish to explore weaker types of lumping which result in nonequivalent models, but where the performability of the lumped model closely approximates that of the unlumped model.

3.5 Hierarchical Modeling of an Air Transport Mission

Several prototype air transport models have been examined in the course of the present reporting period. Below we report in detail on one such model. This is a comprehensive example and should serve to illustrate some of the concepts discussed in the previous sections. In particular, the uses of capability functions (Section 3.2.2), partial capability functions (Section 3.3.1) and interlevel translations (Section 3.3.1) are demonstrated, while state spaces, utilization periods, trajectory spaces and trajectories (Section 3.1.1) are explicitly shown. In addition, the evaluation of performability is exhibited.

This section is organized as follows. First, some notational conventions are set forth (Section 3.5.1). Then, starting from an informal general description (or concept) of a specific air transport mission, an accomplishment set is defined in Section 3.5.2. The particular mission is an extension of the mission discussed in the second Semi-Annual Status Report [2]. With some broad assumptions concerning the aircraft, the upper level models of a model hierarchy were constructed. Section 3.5.3 describes the resulting models. This part of the hierarchy consists of three levels -- the mission level, the aircraft task level, and the computational task level. (A fourth level, the computational hardware level, will be discussed later.) Some of the techniques used to characterize the models at the upper levels have been delineated in the first two Semi-Annual Status Reports [1-2]. However, in the presentation given in this report, interlevel translations have been introduced,

other defining quantities such as state spaces have been explicitly stated, and the overall discussion has been formalized.

Next, Section 3.5.4 reports on a calculus being developed which uses the interlevel translations to determine the trajectory preimages of the capability function, i.e., γ^{-1}. (See Section 3.3.1.) With this calculus, the partial capability functions at each of the first three levels were derived. These are presented in Section 3.5.5.

The final segment of Section 3.5 discusses the performability evaluation (over the total system) of three computers. The three computers are different configurations of four modules of equal computational power. A computer hardware level model was constructed for each computer and placed in the hierarchy. Section 3.5.6 describes these models. Thus, a separate hierarchy was evolved for each computer. From these hierarchies, three capability functions were determined, as reported in Section 3.5.7. The capability functions were then evaluated over several sets of utilization intervals and computer failure rates. Section 3.5.8 discusses METAPHOR, a software package being developed to aid in performability evaluation. Finally, METAPHOR is used to evaluate the three computers.

3.5.1. Notational Conventions

Several conventions concerning notation used in this mission model have been adopted. These are presented here for convenience.

In Sections 3.1.1 and 3.3.1, important foundations were established for the trajectory spaces U and U^i employed in the evaluation of the capability function. The existence of a probability space (Ω, E, P) underlying the total system was postulated and the stochastic processes X^i supporting the trajectory spaces U^i were characterized. Recognition of these quantities is important, particularly to understand the stochastic nature of the models used and to differentiate between trajectories and the random processes which define them.

In the discussion of this mission model, no explicit description of the probability space will be presented other than assigning probabilities to certain events. In particular, specific references to outcomes $\omega \varepsilon \Omega$ will be dropped except where necessary for definition purposes. This is a standard convention for random processes. Furthermore, the random process X^i underlying a trajectory space U^i will not be expressly stated except again where necessary. Therefore a trajectory $u \varepsilon U^i$ implicitly refers to a random process X^i evaluated at some sample $\omega \varepsilon \Omega$ such that $X^i(\omega)=u$. (See Section 3.1.1.)

For this treatment then, a composite trajectory at level i is a function $u_c^i : T_c^i \rightarrow Q_c^i$, where $u_c^i(t) = X_{c,t}^i = X_{c,t}^i(\omega)$ for some $\omega \varepsilon \Omega$. T_c^i is the ith level composite utilization period while Q_c^i is the ith level composite state space. The ith level composite trajectory space is the set $U_c^i = \{u_c^i\} = \{u_{c,\omega}^i | \omega \varepsilon \Omega\}$. In addition, $u_b^i : T_b^i \rightarrow Q_b^i$, $u_b^i(t) = X_{b,t}^i$, T_b^i, Q_b^i, and U_b^i are analogously defined for the basic process X_b^i. The ith level trajectory space is thus $U^i = U_c^i \otimes U_b^i = \{(u_{c,\omega}^i, u_{b,\omega}^i) | \omega \varepsilon \Omega\}$. (See Section 3.1.2).

In this mission model, the random processes X_c^i and X_b^i generally delineate several system components, i.e., features such as hardware subsystems or behavioral functions which are identifiable and helpful in describing the system. As noted in Section 3.3. , Q_c^i and Q_b^i can be coordinatized; the projection of $X_{c,t}^i$ $(X_{b,t}^i)$ on a particular coordinate is called a composite (basic) variable. For the trajectories used, two coordinates are employed. One coordinate is the particular component being observed, while the other coordinate is the observation time.

More precisely, a trajectory $u^i \varepsilon U^i$ is first written as a column array:

$$u^i = \begin{bmatrix} u_c^i \\ \hdashline u_b^i \end{bmatrix}$$

where U_c^i is the composite trajectory and U_b^i is the base trajectory. In case the number of observation times at level i is finite, expansion along the time coordinate yields the representation

$$u^i = \begin{bmatrix} u_c^i(t_1) & u_c^i(t_2) & \dots & u_c^i(t_n) \\ \hdashline u_b^i(t_1) & u_b^i(t_2) & \dots & u_b^i(t_n) \end{bmatrix}$$

where $T^i = \{t_1, t_2, \dots, t_n\}$. If the composite and basic components are respectively $u_{c_1}^i, u_{c_2}^i, \dots, u_{c_m}^i$, and $u_{b_1}^i, u_{b_2}^i, \dots, u_{b_p}^i$, then expansion along the component coordinate yields:

$$u^i = \left[\begin{array}{c} u^i_{c_1} \\ u^i_{c_2} \\ \vdots \\ u^i_{c_m} \\ \hline u^i_{b_1} \\ u^i_{b_2} \\ \vdots \\ u^i_{b_p} \end{array}\right].$$

Thus along both coordinates, the expanded representation is:

$$u^i = \left[\begin{array}{cccc} u^i_{c_1}(t_1) & u^i_{c_1}(t_2) & \cdots & u^i_{c_1}(t_n) \\ u^i_{c_2}(t_1) & u^i_{c_2}(t_2) & \cdots & u^i_{c_2}(t_n) \\ \vdots & \vdots & & \vdots \\ u^i_{c_m}(t_1) & u^i_{c_m}(t_2) & \cdots & u^i_{c_m}(t_n) \\ \hline u^i_{b_1}(t_1) & u^i_{b_1}(t_2) & \cdots & u^i_{b_1}(t_n) \\ u^i_{b_2}(t_1) & u^i_{b_2}(t_2) & \cdots & u^i_{b_2}(t_n) \\ \vdots & \vdots & & \vdots \\ u^i_{b_p}(t_1) & u^i_{b_p}(t_2) & \cdots & u^i_{b_p}(t_n) \end{array}\right].$$

A projection along a single time coordinate is referred to as a <u>trajectory observation</u>. Similarly, a projection along a single component will be called a <u>component trajectory</u>. For instance

$$u^i(t_k) = \begin{bmatrix} u^i_{c_1}(t_k) \\ u^i_{c_2}(t_k) \\ \vdots \\ u^i_{c_m}(t_k) \\ \hline u^i_{b_1}(t_k) \\ u^i_{b_2}(t_k) \\ \vdots \\ u^i_{b_p}(t_k) \end{bmatrix}$$

is a trajectory observation at time t_k, while

$$u^i_{c_j} = \begin{bmatrix} u^i_{c_j}(t_1) & u^i_{c_j}(t_2) & \dots & u^i_{c_j}(t_n) \end{bmatrix}$$

is a component trajectory of the jth composite component. The interval between the kth and $(k+1)^{th}$ sample is called the $(k+1)^{th}$ <u>phase</u>.

As an example to clarify this notation and illustrate its use, consider the following. Suppose, at hierarchy level 2, a model with two composite components (flight control system (FCS) and navigation (NAV)) and a single basic component (air traffic control (ATC)) has been constructed. Suppose further that

the utilization period involves three samples $T = \{t_1, t_2, t_3\}$. To prevent the necessity of writing the level number with every variable of the model, the trajectory space at level 2 is called Y. A trajectory in Y is denoted y, while a variable in y denoting the jth composite component at the kth observation time is written y_{c_j,t_k}. Thus, a trajectory is represented by

$$y = \begin{array}{c} \left[\begin{array}{ccc} y_{c_1,t_1} & y_{c_1,t_2} & y_{c_1,t_3} \\ y_{c_2,t_1} & y_{c_2,t_2} & y_{c_2,t_3} \\ \hline y_{b_1,t_1} & y_{b_1,t_2} & y_{b_1,t_3} \end{array}\right] \end{array} \begin{array}{l} \text{FCS} \\ \text{NAV} \\ \\ \text{ATC} \end{array} .$$

Then with the obvious correspondence $c_1=1$, $c_2=2$, $b_1=3$, $t_1=1$, $t_2=2$, and $t_3=3$, we write

$$y = \left[\begin{array}{ccc} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & y_{33} \end{array}\right] \begin{array}{l} \text{FCS} \\ \text{NAV} \\ \\ \text{ATC} \end{array} .$$

Here the composite trajectory is

$$y_c = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \end{bmatrix}$$

while the basic trajectory is

$$y_b = [y_{31} \quad y_{32} \quad y_{33}] ,$$

where, for example, y_{23} is the state of the navigation system at the third observation time.

Finally a projection function $\xi_{jk}(A) = a_{jk}$, where A is the matrix $[a_{pq}]$ (see Section 3.3.2.1), is frequently composed with an interlevel translation. This is done to extract the particular portion of the function range which is of interest. As an illustration, consider the level i composite model having m components

$$U_c^i = U_{c_1}^i \otimes \ldots \otimes U_{c_m}^i = \{(u_{c_1,\omega}^i, \ldots, u_{c_m,\omega}^i) \mid \omega \varepsilon \Omega\},$$

where the $U_{c_j}^i$ can be further coordinatized along time, $U_{c_j}^i(t_k)$, where $t_k \varepsilon T^i$. Using the projection function, $U_{c_j}^i(t_k)$ will be written $\xi_{jk}U_c^i$. The interlevel translation from level i+1 to level i (assuming level i+1 exists) is

$$\kappa_{i+1}: U_c^{i+1} \otimes U_b^{i+1} \rightarrow U_c^i$$

and to select the function mapping $U_c^{i+1} \otimes U_b^{i+1}$ into $U_{c_j}^i(t_k) = \xi_{jk}U_c^i$, we write

$$\xi_{jk}\kappa_{i+1}: \quad U_c^{i+1} \otimes U_b^{i+1} \rightarrow \xi_{jk}U_c^i \ .$$

We shall refer to $\xi_{jk}\kappa_{i+1}$ as the jth <u>component function at observation k</u>.

3.5.2 Mission Description and Accomplishment Set

We consider a basic air transport mission which can be characterized as follows:

Mission Statement: "Transport passengers between two points safely, conveniently and with minimal fuel consumption."

The total system S = (C,E,P,L) is a flight control computer C operated in the environment E of a portal-to-portal flight of a commercial aircraft P within an airline L. Specifically, C is the object system, E is the environment system, P is the set of related systems, and L is the demand system. The user is interested in fuel efficiency, timeliness and safety; accordingly, the mission statement entails three actions which must be monitored to judge mission performance.

Mission Requirement Set

i) A given safety rate is to be attained.
ii) Inconveniences (diversions) are to be minimized.
iii) Fuel consumption is to be minimized.

Now we can specify a set of accomplishment levels

$$A = \{a_0, a_1, a_2, a_3, a_4\}$$

where in general terms the following correspondences hold:

a_0 = low fuel consumption, no diversion to an alternate landing site, and no fatalities,

a_1 = high fuel consumption, no diversion, and no fatalities,

a_2 = low fuel consumption, diversion, and no fatalities,

a_3 = high fuel consumption, diversion, and no fatalities,

a_4 = fatal crash.

The utilization period of the mission is taken to be T = [0,T]. To develop the model hierarchy, a top down approach

is used. Thus, the model at level i is generated and enlarged before the model at level i+1 (the next lower level) is developed. In the process of characterizing level i, we may not know which variables are to be expanded at level i+1 (i.e., which variables are composite) and which variables will not depend on lower level variables (i.e., which variables are basic). Thus when variables are introduced no claim is made as to whether they are composite or basic. Only when the next lower level is constructed do we make such classifications.

3.5.3. Higher Level Models

This section develops the models used at the first three levels. These are the mission level, the aircraft task level and the computational task level. The model description presented at each level consists of a set of random variables characterizing the system at that level, the state space, the sample time set and the trajectory set. In addition, an interlevel translation is defined at each level to connect that level with the next higher level.

3.5.3.1 Mission Level Model Development

Level 0, the top level, describes those aspects of the total system performance that the user considers important. The model at this level thus characterizes the relevant factors deemed pertinent for a mission. In particular, the model must have a scope broad enough and a level of abstraction high enough to support the accomplishment level descriptions.

For the given accomplishment levels then, an appropriate

scope for the top level is the air carrier, while a suitable abstraction level corresponds to the mission itself. We will therefore refer to this level as the "mission level." After examining the verbal definitions of the accomplishment levels, the mission level can be formally represented by a single variable random process $Z = X_T^0$ taking values in the state space

$$Q^0 = \{0,1\}^3$$

where the values $Z = (z_1, z_2, z_3)$ of Z are interpreted as follows:

$$z_1 = \begin{cases} 0 & \text{if mission is fuel efficient} \\ 1 & \text{otherwise,} \end{cases}$$

$$z_2 = \begin{cases} 0 & \text{if mission is not diverted} \\ 1 & \text{otherwise,} \end{cases}$$

$$z_3 = \begin{cases} 0 & \text{if mission is safe} \\ 1 & \text{otherwise (fatal crash).} \end{cases}$$

Since the model at this level is a single random variable, the trajectory space coincides with the state space, that is

$$Z = U^0 = Q^0 = \{0,1\}^3 \quad .$$

We can now determine the interlevel translation κ_0 between the mission level trajectory space Z and the accomplishment set A. Table 1 specifies κ_0. Thus if we know the value z of the mission variable Z, we know the mission's level of accomplishment. For example, employing the array representation scheme discussed in Section 3.5.1, the (degenerate) trajectory

$$z = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

says the mission resulted in low fuel consumption, no diversion and no fatalities, and accordingly $\kappa_0(z) = a_1$.

$z = (z_1, z_2, z_3)$	$\kappa_0(z)$
0 0 0	a_0
0 0 1	a_4
0 1 0	a_2
0 1 1	a_4
1 0 0	a_1
1 0 1	a_4
1 1 0	a_3
1 1 0	a_4

Table 1

Definition table for $\kappa_0 : Z \to A$.

3.5.3.2 Aircraft Functional Task Level Model Development

The next lower level, level 1, is an intermediate level describing the performance of the aircraft with regard to the total system. For this level, the scope will thus be the aircraft, while the level of abstraction will be the functional tasks of the aircraft. Level 1 will therefore be referred to as the "aircraft functional task level."

To construct the aircraft functional task level, we must first determine an appropriate set of random processes with which to describe the level.
However, to select such a set, we must first examine the system properties we wish the processes to reflect.

For this simple model, assume that we know the following characteristics about the aircraft in which the computer is to be used:

a) For the missions in which the aircraft is utilized, the fuel capacity is such that if fuel consumption is high for more than half of the mission time, the aircraft runs out of fuel and crashes.

b) The aircraft has an autoland system which, if working, will land the plane in any weather. If autoland is being used and fails, the aircraft crashes.

c) The autoland system is used only in Category III weather.

d) If at the initiation of landing, the fuel consumption has been high for any part of the mission, autoland will not be attempted. Instead, the aircraft will be diverted if Category III weather occurs.

e) If the aircraft crashes, fatalities will occur.

At this time, we will not consider factors affecting the mission variables other than those mentioned above. In particular, we ignore those elements of the total system upon which the computer has no effect. Then from the above specifications, we can write the following conditions for the mission variables Z:

$$z_1 = \begin{cases} 0 & \text{if fuel regulation works for the entire mission} \\ 1 & \text{otherwise,} \end{cases}$$

$$z_2 = \begin{cases} 1 & \text{if weather is bad at initiation of landing and autoland is not available} \\ 0 & \text{otherwise,} \end{cases}$$

$$z_3 = \begin{cases} 1 & \text{if either} \\ & \quad \text{a) fuel regulation works for less than half of the mission time} \\ & \text{or b) weather is bad at initiation of landing, and autoland is available at that time but fails during landing} \\ 0 & \text{otherwise.} \end{cases}$$

To characterize the aircraft level, we will utilize a random process with two variables, $Y = \{X^1_{T_L}, X^1_T\}$, where T_L is the time at which landing is initiated. The state space is

$$Q^1 = \{0,1,2,3,4,5,6,7\} \times \{0,1\}$$

where the values $Y = (y_1, y_2)$ of Y have the following meanings:

$$y_1 = \begin{cases} 0 & \text{if fuel regulation works for entire phase and autoland available at end of cruise} \\ 1 & \text{if fuel regulation works entire phase but autoland not available at end of cruise} \\ 2 & \text{if fuel regulation works for time } T_L/2 \le t < T_L \\ 3 & \text{if fuel regulation works for time } t < T_L/2 \\ 4 & \text{if fuel regulation works for entire phase, and autoland successful} \\ 5 & \text{if fuel regulation works for entire phase, but autoland not successful} \\ 6 & \text{if fuel regulation fails, but autoland successful} \\ 7 & \text{if fuel regulation fails and autoland fails,} \end{cases}$$

$$y_2 = \begin{cases} 0 & \text{if non-Category III weather at end of cruise} \\ 1 & \text{otherwise.} \end{cases}$$

With the array representation of Section 3.5.1, a trajectory $y \varepsilon Y$ will be written in the form

$$y = \begin{array}{cc} \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} & \begin{array}{l} \text{Control} \\ \text{Weather} \end{array} \\ \begin{array}{cc} \text{Takeoff/Cruise } t=T_L & \text{Landing } t=T \end{array} & \end{array}$$

where $y_1 = (y_{11}, y_{12})$ are the variables denoting the control systems of the aircraft (i.e., fuel regulation and autoland) and $y_2 = (y_{21}, y_{22})$ are the variables denoting weather the mission encounters.

To be logically consistent with the requirements of the mission variables Z noted above, we can restrict the values that

the y_{ij} can take on. Thus,

$$y_{11} \in \{0,1,2,3\}$$
$$y_{12} \in \{4,5,6,7\}$$
$$y_{21} \in \{0,1\}$$
$$y_{22} = \not{c} \quad \text{(i.e., the fictious state, see Section 3. .) .}$$

That is, y_{11} and y_{12} describe the control performance of the aircraft in terms of the plane's fuel regulation and autoland performance. The variable y_{21} samples the weather at the end of the cruise phase. Since we are unconcerned with the weather during the landing phase, the variable y_{22} is assigned the trivial state $\not{c}$. The aircraft trajectory space is hence

$$Y = U^1 = \{0,1,2,3\} \times \{4,5,6,7\} \times \{0,1\} .$$

With the above definition of Y, we are now able to state the interlevel translation function between the aircraft task level and the mission level, i.e., $\kappa_1 : Y \to Z_c$. Because each component of Z is decomposed within level 1, $Z = Z_c = U_c^0$ and so each component at level 0 is composite. Thus, κ_1 maps into Z, that is, $\kappa_1 : Y \to Z$. Now κ_1 can be broken into its component functions

$$\xi_1\kappa_1 : Y \to \xi_1 Z$$
$$\xi_2\kappa_1 : Y \to \xi_2 Z$$
$$\xi_3\kappa_1 : Y \to \xi_3 Z.$$

Then by matching the definitions of the y_{ij}'s with the definitions of z_1, z_2, and z_3, the following functions are obtained:

$$\xi_1\kappa_1(y) = \begin{cases} 0 \text{ if } y_{11}\varepsilon\{0,1\} \text{ and } y_{12}\varepsilon\{4,5\} \\ 1 \text{ otherwise} \end{cases}$$

$$\xi_2\kappa_1(y) = \begin{cases} 1 \text{ if } y_{11}\varepsilon\{1,2,3\} \text{and } y_{21} = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\xi_3\kappa_1(y) = \begin{cases} 1 \text{ if either} \\ \quad \text{a) } y_{11}=3 \text{ or } (y_{11}=2 \text{ and } y_{12}\varepsilon\{6,7\}) \\ \text{or b) } y_{11}=0 \text{ and } y_{12}\varepsilon\{5,7\} \text{ and } y_{21}=1 \\ 0 \text{ otherwise.} \end{cases}$$

Note that we make two pessimistic and simplifying assumptions regarding $\xi_3\kappa_1$. First, if the fuel regulation works for less than half of the takeoff/cruise phase (hence $y_{11} = 3$), then we assume that the fuel regulation works for less than half of the mission (hence $\xi_1\kappa_1(y) = 1$). This assumption is justified if

$$\begin{aligned} &\Pr\text{ (fuel regulation works for } t < T_L/2 \\ &\quad \text{during } [0,T_L]) \\ &\quad \cong \Pr\text{ (fuel regulation works for } t < T/2 \\ &\qquad \text{during } [0,T]). \end{aligned}$$

Some basis for this claim lies in the fact that the takeoff/cruise phase is usually significantly longer than the landing phase, that is, $T_L >> T-T_L$.

The second assumption is that if the fuel regulation fails at all during the takeoff/cruise phase and then fails at all during the landing phase, then the fuel regulation fails for at least half of the mission. In other words, if $y_{11}\varepsilon\{2,3\}$ and $y_{12}\varepsilon\{5,7\}$, then $\xi_3\kappa_1(y) = 1$.

Both assumptions are pessimistic in that some non-fatal missions will be associated with fatalities (z_3=1). These assumptions must be made because the resolution of the aircraft function variables does not allow the exact determination of the time that the fuel regulation works. Such a determination could be made by simply modifying the aircraft function variables to have one variable monitor the autoland and a second variable (continuous, ranging over [0,T]) keep track of the time the fuel regulation works. However, for this illustrative example, we adopt the simpler view.

3.5.3.3 Computational Task Level Model Development

In continuing the decomposition of the mission and the corresponding development of the hierarchy, we must next describe the variables composing the aircraft functions described in the previous section. Because we wish to evaluate the computer's effectiveness, we ignore non-computer related components. Thus, the scope of this level will be the computer while the level of abstraction will be the functional tasks of the computer. Hence level 2 will be called the "computational task level."

For this mission, we assume that the aircraft tasks fuel regulation and autoland each has a computational task: fuel regulation computations and autoland computations. Furthermore, we assume that in order for the aircraft task to be successful, its corresponding computational task must also be successful. If, for example, not all of the autoland computations are done, then the autoland task is not achieved, and so autoland can not be

performed. The weather variable at the aircraft functional task level (y_{21} and y_{22}) has no computer support and therefore is a basic variable.

The computational task level can be described by a three variable random process $X = \{X^2_{T_H}, X^2_{T_L}, X^2_T\}$, where T_H is an observation time halfway between 0 and T_L, that is, $T_H = T_L/2$. The state space is

$$Q = \{0,1\}^2$$

where the values $x = (x_1, x_2)$ of X have the following interpretations:

$$x_1 = \begin{cases} 0 & \text{if fuel regulation computations are successful} \\ 1 & \text{otherwise,} \end{cases}$$

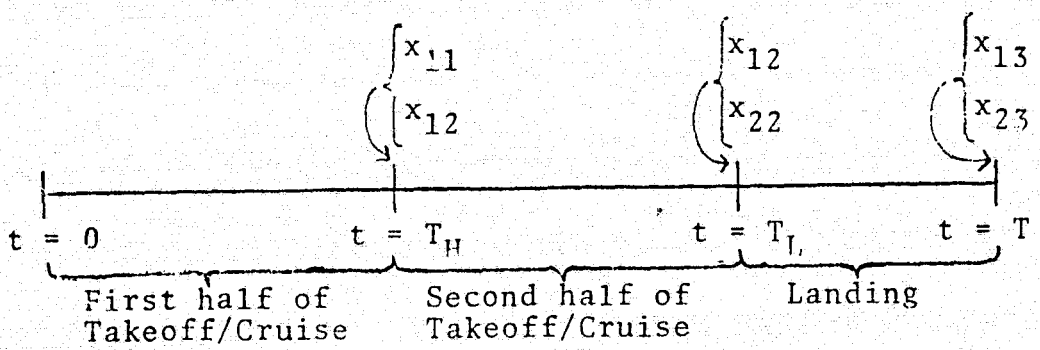
$$x_2 = \begin{cases} 0 & \text{if autoland computations are successful} \\ 1 & \text{otherwise.} \end{cases}$$

In the representation of Section 3.5.1, a trajectory $x \varepsilon X$ will be written as

$$x = \begin{array}{cl} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} & \begin{array}{l} \text{Fuel regulation computations} \\ \text{Autoland computations} \end{array} \\ \underbrace{\qquad\qquad}_{\text{Takeoff/cruise}} \quad \underbrace{\qquad}_{\text{Landing}} & \end{array}$$

where $x_1 = (x_{11}, x_{12}, x_{13})$ are variables representing the fuel regulation computations while $x_2 = (x_{21}, x_{22}, x_{23})$ are variables representing the autoland computations.

The observations x_{ij} are made as follows:



Because success or failure of the autoland computations at the halfway point of the cruise phase cannot affect the availability of the computations at the end of the cruise phase, the variable x_{21} is assigned ¢, the fictitious state. All other variables can be either 0 or 1. Thus, the computational level trajectory space is $X = U^2 = \{0,1\}^5$. All computational tasks are assumed successful at the beginning of the mission (t = 0). Also, knowledge of the system's behavior at the observations T_H, T_L and T is assumed to yield sufficient information to infer the values of the level 1 variables.

With the above assumptions, we can then construct the relation between X and Y:

$$\kappa_2 : X \to Y_c$$

by first constructing:

$$\xi_{11}\kappa_2 : X \to Y_{11}$$

$$\text{and} \quad \xi_{12}\kappa_2 : X \to Y_{12}$$

Thus,

$$\xi_{11}\kappa_2(x) = \begin{cases} 0 \text{ if } x_{11} = x_{12} = x_{22} = 0 \\ 1 \text{ if } x_{11} = x_{12} = 0 \text{ and } x_{22} = 1 \\ 2 \text{ if } (x_{11} = 1 \text{ and } x_{12} = 0) \text{ or} \\ \qquad (x_{11} = 0 \text{ and } x_{12} = 1) \\ \qquad [\text{i.e., if } x_{11} \oplus x_{12} = 1, \text{ where } \oplus \\ \qquad \text{denotes the "exclusive or" operation}] \\ 3 \text{ if } x_{11} = x_{12} = 1 \end{cases}$$

and

$$\xi_{12}\kappa_2(x) = \begin{cases} 4 \text{ if } x_{13} = x_{23} = 0 \\ 5 \text{ if } x_{13} = 0 \text{ and } x_{23} = 1 \\ 6 \text{ if } x_{13} = 1 \text{ and } x_{23} = 0 \\ 7 \text{ if } x_{13} = 1 \text{ and } x_{23} = 1. \end{cases}$$

3.5.4 A Calculus of Trajectory Sets

Although a fourth level (the bottom computer hardware and software level) has yet to be discussed, it is convenient at this time to introduce a calculus being developed which is of great use in determining the γ-induced trajectory sets, i.e., γ^{-1}. This calculus will be used to simplify the upper level models of the hierarchy before any lower level models are examined. Also, the calculus is used to assimilate lower levels as they are developed. After the lowest level of the hierarchy has been operated upon, the result is γ^{-1}. Derivation of γ^{-1} is important because, using the techniques of Section 3.4.2 on γ^{-1}, performability calculations for the system can then be effected.

The calculus presented here is part of an ongoing effort to produce general tools for performability evaluation. Although the description in this section is oriented to this particular mission model, the technique is generalizable. Section 3.5.4.1 below furnishes an algorithm (based on the given mission model; see Section 3.3.1 for the general case) for extracting γ^{-1}, while Section 3.5.4.2 gives the actual trajectory set calculus in terms of a representation for trajectory sets and some basic operations on those sets.

C-2

3.5.4.1 An Algorithm for Determining γ^{-1}

An immediate goal within this mission model discussion is a characterization of the capability function

$$\gamma: W \times X_b \times Y_b \times Z_b \to A$$

where W is the fourth level trajectory space to be described in Section 3.5.6. From γ, we then determine the preimage sets of γ (i.e., the "γ-induced trajectory sets"). That is, we wish to find $\gamma^{-1}(a) = \{u | \gamma(u) = a$ and $u \in W \times X_b \times Y_b \times Z_b$, i.e., u is a mission trajectory$\}$ for all $a \in A$. Then using the techniques outlined in Section 3.4, we can determine the probability distribution of the accomplishment set for the mission, that is, the performability:

$$p_S(a) = Pr(\{u | \gamma(u) = a\}) = Pr(\gamma^{-1}(a))$$

for all $a \in A$.

Below is an algorithm for determining γ^{-1}. Based on the discussion in Section 3.3.1, the algorithm constructs γ^{-1} iteratively employing partial capability functions and interlevel translations. The symbol $\forall$ denotes "for all."

1) Find $\gamma_0^{-1}(a) = \{(z) | \gamma_0(z) = a\}$ $\forall$ $a \in A$. Since $\gamma_0 = \kappa_0$, this is equivalent to finding $\kappa_0^{-1}(a)$ $\forall$ $a \in A$.

2) Find $\gamma_1^{-1}(a) = \{(y,z_b) | \gamma_1(y,z_b) = a\}$ $\forall$ $a \in A$. This is achieved by finding $\kappa_1^{-1}(z_c)$ for each $z = \begin{bmatrix} z_c \\ \hline z_b \end{bmatrix} \in \gamma_1^{-1}(a)$. Since in this hierarchy, z_b is empty, then $z_c = z$ and so we need to find a $\kappa_1^{-1}(z)$ $\forall$ $z \in Z$. Once κ_1^{-1} is known,

$$\gamma_1^{-1}(a) = \{(y,z_b) | \gamma_1(y,z_b) = a\}$$
$$= \{(\kappa_1^{-1}(z) | \gamma_0(z) = a\}.$$

3) Find $\gamma_2^{-1}(a) = \{(x,y_b,z_b) | \gamma_2(x,y_b,z_b) = a\}$ $\forall$ $a \in A$. Here, we must find $\kappa_2^{-1}(y_c)$ for each $y = \begin{bmatrix} y_c \\ \hline y_b \end{bmatrix} \in \gamma_1^{-1}(a)$. Then

$$\gamma_2^{-1}(a) = \{(x,y_b,z_b) | \gamma_2(x,y_b,z_b) = a\}$$
$$= \{(\kappa_2^{-1}(y_c),\ y_b\ ) | \gamma_1(\ y\ ) = a\}.$$

4) Find $\gamma^{-1}(a) = \gamma_3^{-1}(a) = \{(w,x_b,y_b,z_b) | \gamma_3(w,x_b,y_b,z_b) = a\}$ $\forall\ a \in A$. Similar to the method in step 3, determine $\kappa_3^{-1}(x_c)$ for each $x = \begin{bmatrix} x_c \\ \hdashline x_b \end{bmatrix} \in \gamma_2^{-1}(a)$. From this,

$$\gamma^{-1}(a) = \gamma_3^{-1}(a)$$
$$= \{(w,x_b,y_b,z_b) | \gamma_3(w,x_b,y_b,z_b) = a\}$$
$$= \{(\kappa_3^{-1}(x_c),x_b,\ y_b\ ) | \gamma_2(x,\ y_b\ ) = a\}.$$

Note that since $\kappa_i : U^i \to U_c^{i+1}$, then both the domain and range of κ_i can be represented by arrays as discussed in Section 3.5.1. Also, the component functions $\xi_{jk}\kappa_i : U^i \to \xi_{jk}U_c^{i+1}$ exist and are identifiable. Now the κ_i inverse of $v \in U_c^{i+1}$, $\kappa_i^{-1}(v)$, is simply the set of all trajectories $u \in U^i$ such that $\kappa_i(u) = v$. Furthermore, it is plain to see that for u to map into v, each component function $\xi_{jk}\kappa_i(u)$ must map into the corresponding component $\xi_{jk}v$. That is,

$$\kappa_i(u) = v \text{ if and only if } \xi_{jk}\kappa_i(u) = \xi_{jk}v,$$
$$\text{for all proper } j \text{ and } k.$$

Thus, the inverse of v must be the intersection of all the inverses of $\xi_{jk}v$. Hence,

$$\kappa_i^{-1}(v) = \bigcap_{j,k} \xi_{jk}\kappa_i^{-1}(v).$$

For instance, if

u	$\kappa_i(u) = v$
1	a a
2	a b
3	b a
4	b b

then

$\xi_1 v$	$(\xi_1\kappa_i)^{-1}(v)$
a	{1, 2}
b	{3, 4}

$\xi_2 v$	$(\xi_2\kappa_i)^{-1}(v)$
a	{1, 3}
b	{2, 4}

and so

v	$(\xi_1\kappa_i)^{-1}(v) \cap (\xi_2\kappa_i)^{-1}(v) = \kappa_i^{-1}(v)$
a a	{1, 2} ∩ {1, 3} = 1
a b	{1, 2} ∩ {2, 4} = 2
b a	{3, 4} ∩ {1, 3} = 3
b b	{3, 4} ∩ {2, 4} = 4

The next section introduces some methods examined during the reporting period which facilitate writing these preimage sets.

3.5.4.2 Trajectory Sets, Array Products, and Intersections

Manipulation of sets of trajectories is necessary to derive the preimage sets of γ. However, handling such sets can be awkward because of their size. Therefore, we have been investigating techniques of operating with sets of trajectories in a convenient and compact manner. This section reports on the most promising calculus investigated.

A set of trajectories will be called a trajectory set. We first introduce a simple representation of a trajectory set. Consider the trajectory

$$u = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}$$

where each u_{ij} can assume values in a set of states Q_{ij}. Each u_{ij} is a "variable." For example, we may have

$$y = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

where $y_{11} \in \{1,2,3\}$, $y_{12} \in \{0,2,4\}$, $y_{21} \in \{-1,-2\}$, and $y_{22} \in \{a,b,c\}$. Suppose we have two trajectories u_1 and u_2 such that u_1 and u_2 are equal variable-by-variable except for a single variable. That is

$$u_1 = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \cdots & u_{ij} & \cdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}$$

$$u_2 = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \cdots & u'_{ij} & \cdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}.$$

where $u_{ij} \neq u'_{ij}$. We then write the trajectory set $\{u_1, u_2\}$ as

$$\{u_1,u_2\} = \left\{ \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \cdots & u_{ij} & \cdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix}, \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \cdots & u'_{ij} & \cdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \right\}$$

$$= \begin{bmatrix} \{u_{11}\} & \cdots & \{u_{1n}\} \\ \cdots & \{u_{ij}, u'_{ij}\} & \cdots \\ \{u_{m1}\} & \cdots & \{u_{mn}\} \end{bmatrix}.$$

This representation is called an <u>array product</u>. Note that the concept is similar to that of a cross product. Of course, the idea can be generalized:

$$\begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & & \vdots \\ R_{m1} & \cdots & R_{mn} \end{bmatrix} = \left\{ \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \;\middle|\; u_{ij} \in R_{ij} \subseteq Q_{ij} \right\}.$$

As an illustration, suppose that

$$y_1 = \begin{bmatrix} 1 & 2 \\ -2 & a \end{bmatrix} \qquad y_2 = \begin{bmatrix} 3 & 2 \\ -2 & a \end{bmatrix}$$

$$y_3 = \begin{bmatrix} 1 & 2 \\ -2 & b \end{bmatrix} \qquad y_4 = \begin{bmatrix} 3 & 2 \\ -2 & a \end{bmatrix} ,$$

then

$$\{y_1, y_2, y_3, y_4\} = \begin{bmatrix} \{1,3\} & \{2\} \\ \{-2\} & \{a,b\} \end{bmatrix} .$$

Because the use of array products has been so widespread in our work with trajectory sets, we have adopted the simplifying convention of writing array product elements which are singleton sets as elements without set brackets. Thus

$$\{y_1, y_2, y_3, y_4\} = \begin{bmatrix} \{1,3\} & 2 \\ -2 & \{a,b\} \end{bmatrix} .$$

No confusion should result since context will make clear whether an object is a array product (and hence a set) or a single trajectory. Furthermore, this convention makes array products easier to read by cutting down on the number of brackets that a reader must wade through.

Often a single array product cannot by itself represent all the trajectories within a trajectory set. In that instance, the union of several array products must be employed to represent the trajectory set. Thus, for the general case, we write a trajectory set as the union of p array products, P_i:

$$\{u_1,\ldots,u_\ell\} = P_1 \cup \ldots \cup P_p$$

$$= \begin{bmatrix} R^1_{11} & \ldots & R^1_{1n} \\ \vdots & & \vdots \\ R^1_{m1} & \ldots & R^1_{mn} \end{bmatrix} \cup \ldots \cup \begin{bmatrix} R^p_{11} & \ldots & R^p_{1n} \\ \vdots & & \vdots \\ R^p_{m1} & \ldots & R^p_{mn} \end{bmatrix}$$

$$= \left\{ \begin{bmatrix} u^i_{11} & \ldots & u^i_{1n} \\ \vdots & & \vdots \\ u^i_{m1} & \ldots & u^i_{mn} \end{bmatrix} \;\middle|\; \begin{array}{l} i \in \{1, \ldots, p\}, \\ \quad u_{jk} \in R_{jk} \subset Q_{jk} \end{array} \right\}.$$

For example, in addition to y_1, y_2, y_3, and y_4 above, let

$$y_5 = \begin{bmatrix} 3 & 0 \\ -2 & a \end{bmatrix} \qquad y_6 = \begin{bmatrix} 3 & 0 \\ -2 & b \end{bmatrix}.$$

Then

$$\{y_1,y_2,y_3,y_4,y_5,y_6\} = \begin{bmatrix} \{1,3\} & 2 \\ -2 & \{a,b\} \end{bmatrix} \cup \begin{bmatrix} 3 & 0 \\ \{-2\} & \{a,b\} \end{bmatrix}.$$

In passing, note that this representation is not unique, e.g., the set above can also be written

$$\{y_1,y_2,y_3,y_4,y_5,y_6\} = \begin{bmatrix} 3 & \{0,2\} \\ -2 & \{a,b\} \end{bmatrix} \cup \begin{bmatrix} 1 & 2 \\ -2 & \{a,b\} \end{bmatrix}.$$

A canonical form can be easily defined. For instance, an ordering of the trajectories in the set can be made and used as a basis for constructing the array products. However, for the mission model example discussed in this report, a unique representation is not required and so will not be formalized.

Two special sets should be mentioned. One is the <u>empty set</u> (or null set) $\emptyset$, the set containing no elements. The other is the <u>full set</u> (or universe) $*$ which represents the set containing

all elements "of interest." For trajectory sets, this is the set of all possible states a variable can assume. For instance, if

$$y_7 = \begin{bmatrix} 1 & 2 \\ -1 & a \end{bmatrix},$$

then

$$\{y_1, y_7\} = \begin{bmatrix} 1 & 2 \\ * & a \end{bmatrix}.$$

Another frequently used item is the <u>null array</u> Φ. This is defined to be any array product which contains the empty set ϕ as an element. As an instance,

$$\Phi = \begin{bmatrix} \{1,2\} & \phi \\ * & \{a,b\} \end{bmatrix}.$$

We now define the operation of intersection on the class of array products. The intersection $\cap$ of two array products P_1 and P_2 is the element-by-element intersection of the two arrays. P_1 and P_2 must have the same dimensions.

$$P_1 \cap P_2 = \begin{bmatrix} R^1_{11} & \cdots & R^1_{1n} \\ \vdots & & \vdots \\ R^1_{m1} & \cdots & R^1_{mn} \end{bmatrix} \cap \begin{bmatrix} R^2_{11} & \cdots & R^2_{1n} \\ \vdots & & \vdots \\ R^2_{m1} & \cdots & R^2_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} R^1_{11} \cap R^2_{11} & \cdots & R^1_{1n} \cap R^2_{1n} \\ \vdots & & \vdots \\ R^1_{m1} \cap R^2_{m1} & \cdots & R^1_{mn} \cap R^2_{mn} \end{bmatrix}.$$

The following table defines the element intersection $R^1_{ij} \cap R^2_{ij}$:

	a_1	ϕ	$*$	$\not\in$
a_2	$a_1 \cap a_2$	ϕ	a_2	a_2
ϕ	ϕ	ϕ	ϕ	ϕ
$*$	a_1	ϕ	$*$	$*$
$\not\in$	a_1	ϕ	$*$	$\not\in$

where a_1 and a_2 are any sets and $a_1 \cap a_2$ is standard set intersection. Thus

$$\begin{bmatrix} \{1,2\} & \phi \\ * & \{a,b\} \end{bmatrix} \cap \begin{bmatrix} \{1,3\} & \{0,2\} \\ -2 & \not\in \end{bmatrix} = \begin{bmatrix} \{1,2\} \cap \{1,3\} & \phi \cap \{0,2\} \\ * \cap -2 & \{a,b\} \cap \not\in \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \phi \\ -2 & \{a,b\} \end{bmatrix}$$

$$= \Phi .$$

Array product intersection is distributive over set union. For instance:

$$\begin{bmatrix} \{1,2\} & \phi \\ * & \{a,b\} \end{bmatrix} \cap \left(\begin{bmatrix} \{1,2\} & 0 \\ -2 & \{b,c\} \end{bmatrix} \cup \begin{bmatrix} \{1,3\} & \{0,2\} \\ -2 & \not\in \end{bmatrix} \right)$$

$$= \left(\begin{bmatrix} \{1,2\} & \phi \\ * & \{a,b\} \end{bmatrix} \cap \begin{bmatrix} \{1,2\} & 0 \\ -2 & \{b,c\} \end{bmatrix} \right) \cup \left(\begin{bmatrix} \{1,2\} & \phi \\ * & \{a,b\} \end{bmatrix} \cap \begin{bmatrix} \{1,3\} & \{0,2\} \\ -2 & \not\in \end{bmatrix} \right)$$

$$= \begin{bmatrix} \{1,2\} & \phi \\ -2 & b \end{bmatrix} \cup \begin{bmatrix} 1 & \phi \\ -2 & \{a,b\} \end{bmatrix}$$

The complement P^c of an array product P is the set of all arrays not represented by P. This can be found as follows:

$$P^c = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & & \vdots \\ R_{m1} & \cdots & R_{mn} \end{bmatrix}^c$$

$$= \begin{bmatrix} R_{11}^c & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{bmatrix} \cup \cdots \cup \begin{bmatrix} * & \cdots & R_{1n}^c \\ \vdots & & \vdots \\ * & \cdots & * \end{bmatrix} \cup \cdots \cup$$

$$\begin{bmatrix} * & \cdots & * \\ \vdots & & \vdots \\ R_{m1}^c & \cdots & * \end{bmatrix} \cup \cdots \cup \begin{bmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & R_{mn}^c \end{bmatrix}$$

$$= \bigcup_{\substack{k=1 \\ \ell=1}}^{\substack{k=m \\ \ell=n}} \begin{bmatrix} \overline{R}_{11} & \cdots & \overline{R}_{1n} \\ \cdots & \overline{R}_{k\ell} & \cdots \\ \overline{R}_{m1} & \cdots & \overline{R}_{mn} \end{bmatrix},$$

where

$$\overline{R}_{ij} = \begin{cases} R_{ij}^c & \text{if } k=i,\ \ell=j \\ * & \text{otherwise} \end{cases},$$

$R_{ij} \subseteq Q_{ij}$, and $R_{ij}^c = \{q \mid q \in Q_{ij} \text{ and } q \notin R_{ij}\}$. Also, $*^c = \phi$, $\phi^c = *$, and $\phi^c = \phi$.

To determine the complement of a trajectory set, De Morgan's Law could be used. Suppose V is a trajectory set composed of p array products. Then

$$V^c = (P_1 \cup \cdots \cup P_p)^c$$
$$= P_1^c \cap \cdots \cap P_p^c \quad .$$

As an example,

$$\left(\begin{bmatrix}\{1,3\} & 2\\ -2 & \{a,b\}\end{bmatrix} \cup \begin{bmatrix}\{1,2\} & 2\\ -1 & *\end{bmatrix}\right)^c = \begin{bmatrix}\{1,3\} & 2\\ -2 & \{a,b\}\end{bmatrix}^c \cap \begin{bmatrix}\{1,2\} & 2\\ -1 & *\end{bmatrix}^c$$

$$= \left(\begin{bmatrix}2 & *\\ * & *\end{bmatrix} \cup \begin{bmatrix}* & \{0,4\}\\ * & *\end{bmatrix} \cup \begin{bmatrix}* & *\\ -1 & *\end{bmatrix} \cup \begin{bmatrix}* & *\\ * & c\end{bmatrix}\right) \cap$$

$$\left(\begin{bmatrix}3 & *\\ * & *\end{bmatrix} \cup \begin{bmatrix}* & \{0,4\}\\ * & *\end{bmatrix} \cup \begin{bmatrix}* & *\\ -2 & *\end{bmatrix} \cup \begin{bmatrix}* & *\\ * & \phi\end{bmatrix}\right)$$

$$= \begin{bmatrix}2 & \{0,4\}\\ * & *\end{bmatrix} \cup \begin{bmatrix}2 & *\\ -2 & *\end{bmatrix} \cup \begin{bmatrix}3 & \{0,4\}\\ * & *\end{bmatrix} \cup \begin{bmatrix}* & \{0,4\}\\ * & *\end{bmatrix} \cup$$

$$\begin{bmatrix}* & \{0,4\}\\ -2 & *\end{bmatrix} \cup \begin{bmatrix}3 & *\\ -1 & *\end{bmatrix} \cup \begin{bmatrix}* & \{0,4\}\\ -1 & *\end{bmatrix} \cup \begin{bmatrix}3 & *\\ * & c\end{bmatrix} \cup$$

$$\begin{bmatrix}* & \{0,4\}\\ * & c\end{bmatrix} \cup \begin{bmatrix}* & *\\ -2 & c\end{bmatrix} .$$

We have found, however, that the evaluation of performability ($Pr(\gamma^{-1}(a))$) is often simpler if the γ-induced trajectory sets ($\gamma^{-1}(a)$) are represented as the union of disjoint sets. The set above, for instance, can be written

$$\left(\begin{bmatrix}\{1,3\} & 2\\ -2 & \{a,b\}\end{bmatrix} \cup \begin{bmatrix}\{1,2\} & 2\\ -1 & *\end{bmatrix}\right)^c = \begin{bmatrix}* & \{0,4\}\\ * & *\end{bmatrix} \cup \begin{bmatrix}2 & 2\\ -2 & *\end{bmatrix} \cup \begin{bmatrix}3 & 2\\ -1 & *\end{bmatrix} \cup \begin{bmatrix}\{1,3\} & 2\\ -2 & c\end{bmatrix} .$$

Representing a trajectory set as a union of disjoint array products has analogies with representing a Boolean function in disjunctive normal form. Thus, we see the possibility of generalizing Roth's cubical calculus (see [15] for instance) to handle sets other than 0,1 and so manipulate trajectory sets. However, we have not yet formalized these techniques, and so they will not be discussed in depth within this report. During the next reporting period, we intend to continue this effort.

3.5.5 Higher Level Partial Capability Function Preimage Sets

Using the techniques given in the previous sections, γ_3^{-1} was derived. The following sections outline that process. We have found that simplifying the higher level models (by their partial capability function preimage sets) makes the insertion of several different bottom level models more wieldly. Thus, discussion of the bottom levels is postponed until Section 3.5.6.

As a review, Figure 2 displays the hierarchical structure for the mission as defined thus far in the report. Figure 3 summarizes the variables employed in those higher level models.

3.5.5.1 γ_0-Induced Trajectory Sets

From the definition of κ_0 in Section 3.5.3.1, $\kappa_0^{-1} = \gamma_0^{-1}$ is immediately obtained. Table 2 gives the γ_0 preimage sets. Note that each $\gamma_0^{-1}(a)$ is expressed as an array product.

3.5.5.2 γ_1-Induced Trajectory Sets

To determine γ_1^{-1}, we first reexamine κ_1. From Section 3.5.3.2:

$$\xi_1\kappa_1(y) = \begin{cases} 0 \text{ if } y_{11} \in \{0,1\} \text{ and } y_{12} \in \{4,5\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } y \in \begin{bmatrix} \{0,1\} & \{4,5\} \\ * & ¢ \end{bmatrix} \\ 1 \text{ otherwise} \end{cases}$$

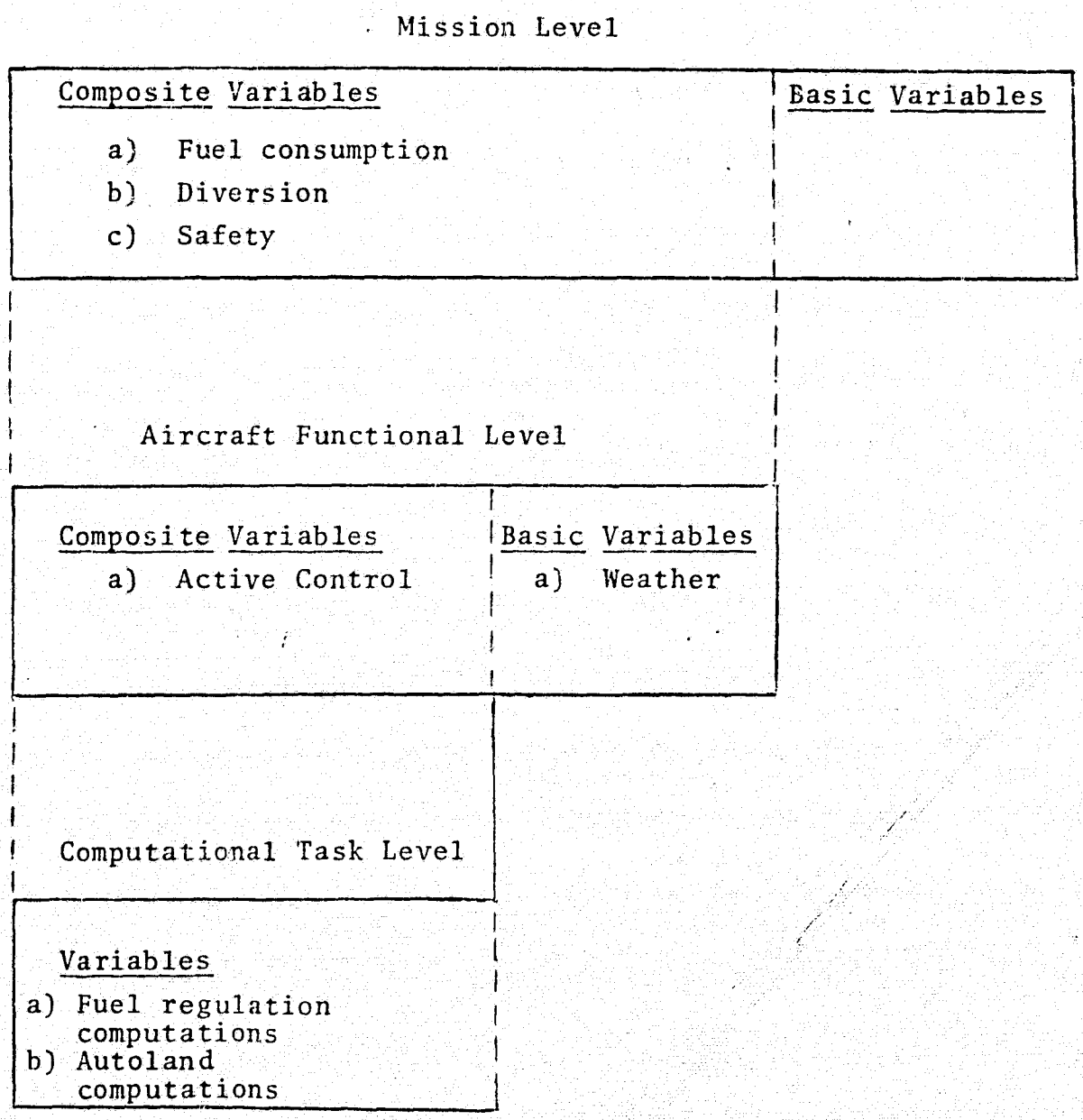


Figure 2

Mission variables defined in the upper levels of an air transport mission model hierarchy.

Accomplishment Set

$$A = \{a_0, a_1, a_2, a_3, a_4\}$$

Mission Level

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \begin{array}{l} \text{Fuel consumption} \\ \text{Diversion} \\ \text{Safety} \end{array}$$

$$Z = \{z\} = \begin{bmatrix} \{0,1\} \\ \{0,1\} \\ \{0,1\} \end{bmatrix}$$

Aircraft Functional Level

$$y = \begin{bmatrix} y_{11} & y_{12} \\ \hline y_{21} & y_{22} \end{bmatrix} \begin{array}{l} \text{Control} \\ \text{Weather} \end{array}$$

$$Y = \{y\} = \begin{bmatrix} \{0,1,2,3\} & \{4,5,6,7\} \\ \{0,1\} & ¢ \end{bmatrix}$$

Computational Task Level

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} \begin{array}{l} \text{Fuel regulation computations} \\ \text{Autoland computations} \end{array}$$

$$X = \{x\} = \begin{bmatrix} \{0,1\} & \{0,1\} & \{0,1\} \\ ¢ & \{0,1\} & \{0,1\} \end{bmatrix}$$

Figure 3

Variables employed in the higher levels of the model hierarchy.

$a \in A$	$\gamma_0^{-1}(a) = \kappa_0^{-1}(a) \subseteq Z$
a_0	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
a_1	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
a_2	$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$
a_3	$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$
a_4	$\left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\} = \begin{bmatrix} * \\ * \\ 1 \end{bmatrix}, \quad * = \{0,1\}$

Table 2

Pre-images of γ_0

$$\xi_2\kappa_1(y) = \begin{cases} 1 \text{ if } y_{11} \in \{1,2,3\} \text{ and } y_{21} = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } y \in \begin{bmatrix} \{1,2,3\} & * \\ 1 & ¢ \end{bmatrix} \\ 0 \text{ otherwise,} \end{cases}$$

$$\xi_3\kappa_1(y) = \begin{cases} 1 \text{ if either a) } y_{11} = 3 \text{ or } (y_{11} = 2 \text{ and } y_{12} \in \{6,7\}) \\ \qquad\qquad \text{or b) } y_{11} = 0 \text{ and } y_{21} = 1 \text{ and } y_{12} \in \{5,7\} \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } y \in \begin{bmatrix} 3 & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 2 & \{6,7\} \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & \{5,7\} \\ 1 & ¢ \end{bmatrix} \\ 0 \text{ otherwise.} \end{cases}$$

The component inverses are:

$$(\xi_1\kappa_1)^{-1}(z_1) = \begin{cases} \begin{bmatrix} \{0,1\} & \{4,5\} \\ * & ¢ \end{bmatrix} & \text{if } z_1 = 0 \\ \begin{bmatrix} \{2,3\} & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} \{0,1\} & \{6,7\} \\ * & ¢ \end{bmatrix} & \text{otherwise,} \end{cases}$$

$$(\xi_2\kappa_1)^{-1}(z_2) = \begin{cases} \begin{bmatrix} \{1,2,3\} & * \\ 1 & ¢ \end{bmatrix} & \text{if } z_2 = 1 \\ \begin{bmatrix} 0 & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} \{1,2,3\} & * \\ 0 & ¢ \end{bmatrix} & \text{otherwise,} \end{cases}$$

$$(\xi_3\kappa_1)^{-1}(z_3) = \begin{cases} \begin{bmatrix} 3 & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 2 & \{6,7\} \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & \{5,7\} \\ 1 & ¢ \end{bmatrix} & \text{if } z_3 = 1 \\ \begin{bmatrix} 1 & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & \{4,6\} \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & \{5,7\} \\ 0 & ¢ \end{bmatrix} \cup \begin{bmatrix} 2 & \{4,5\} \\ * & ¢ \end{bmatrix} & \text{otherwise.} \end{cases}$$

The derivations of $(\xi_1\kappa_1)^{-1}(0)$, $(\xi_2\kappa_1)^{-1}(1)$, and $(\xi_3\kappa_1)^{-1}(1)$ follow immediately from the definitions of $\xi_1\kappa_1(0)$, $\xi_2\kappa_1(1)$, and $\xi_3\kappa_1(1)$ respectively, while $(\xi_1\kappa_1)^{-1}(1)$, $(\xi_2\kappa_1)^{-1}(0)$, and $(\xi_3\kappa_1)^{-1}(0)$ were found by taking the complements of $(\xi_1\kappa_1)^{-1}(0)$, $(\xi_2\kappa_1)^{-1}(1)$, and $(\xi_3\kappa_1)^{-1}(1)$.

Armed with the $(\xi_i\kappa_i)^{-1}$ relations above, we can now state γ_1^{-1} using

$$\gamma_1^{-1}(a) = \{(\kappa_1^{-1}(z), z) \mid \gamma_0(z) = a\}$$

$$\kappa_1^{-1}(z) = (\xi_1\kappa_1)^{-1}(z_1) \cap (\xi_2\kappa_1)^{-1}(z_2) \cap (\xi_3\kappa_1)^{-1}(z_3).$$

As an example of the computations involved, consider

$$z = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Then

$$\begin{aligned}
\kappa_1^{-1}(z) &= (\xi_1\kappa_1)^{-1}(0) \cap (\xi_2\kappa_1)^{-1}(0) \cap (\xi_3\kappa_1)^{-1}(0) \\
&= \left(\begin{bmatrix} \{0,1\} & \{4,5\} \\ * & \not\in \end{bmatrix}\right) \cap \left(\begin{bmatrix} 0 & * \\ * & \not\in \end{bmatrix} \cup \begin{bmatrix} \{1,2,3\} & * \\ 0 & \not\in \end{bmatrix}\right) \cap \left(\begin{bmatrix} 1 & * \\ * & \not\in \end{bmatrix}\right. \\
&\qquad \left. \cup \begin{bmatrix} 0 & \{4,6\} \\ * & \not\in \end{bmatrix} \cup \begin{bmatrix} 0 & \{5,7\} \\ 0 & \not\in \end{bmatrix} \cup \begin{bmatrix} 2 & \{4,5\} \\ * & \not\in \end{bmatrix}\right) \\
&= \begin{bmatrix} 0 & 4 \\ * & \not\in \end{bmatrix} \cup \begin{bmatrix} 0 & 5 \\ 0 & \not\in \end{bmatrix} \cup \begin{bmatrix} 1 & \{4,5\} \\ 0 & \not\in \end{bmatrix}.
\end{aligned}$$

So, since $\{z \mid \gamma_0(z) = a_0\} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, then

$$\gamma_1^{-1}(z) = \begin{bmatrix} 0 & 4 \\ * & \not\in \end{bmatrix} \cup \begin{bmatrix} 0 & 5 \\ 0 & \not\in \end{bmatrix} \cup \begin{bmatrix} 1 & \{4,5\} \\ 0 & \not\in \end{bmatrix}.$$

Table 3 displays the γ_1 preimage sets for all $a \in A$.

$a \in A$	$\gamma_1^{-1}(a) \in Y$
a_0	$\begin{bmatrix} 0 & 4 \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & 5 \\ 0 & ¢ \end{bmatrix} \cup \begin{bmatrix} 1 & \{4,5\} \\ 0 & ¢ \end{bmatrix}$
a_1	$\begin{bmatrix} 2 & \{4,5\} \\ 0 & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & 6 \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & 7 \\ 0 & ¢ \end{bmatrix} \cup \begin{bmatrix} 1 & \{6,7\} \\ 0 & ¢ \end{bmatrix}$
a_2	$\begin{bmatrix} 1 & \{4,5\} \\ 1 & ¢ \end{bmatrix}$
a_3	$\begin{bmatrix} 2 & \{4,5\} \\ 1 & ¢ \end{bmatrix} \cup \begin{bmatrix} 1 & \{6,7\} \\ 1 & ¢ \end{bmatrix}$
a_4	$\begin{bmatrix} 3 & * \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 2 & \{6,7\} \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & \{5,7\} \\ 1 & ¢ \end{bmatrix}$

Table 3
Preimages of γ_1

3.5.5.3 γ_2-Induced Trajectory Sets

The derivation of γ_2^{-1} is similar to that of γ_1^{-1}. First, consider κ_2 from Section 3.5.3.3:

$$\xi_{11}\kappa_2(x) = \begin{cases} 0 \text{ if } x_{11} = x_{12} = x_{22} = 0 \\ 1 \text{ if } x_{11} = x_{12} = 0 \text{ and } x_{22} = 1 \\ 2 \text{ if } (x_{11} = 1 \text{ and } x_{12} = 0) \\ \qquad \text{or } (x_{11} = 0 \text{ and } x_{12} = 1) \\ 3 \text{ if } x_{11} = x_{12} = 1 \end{cases}$$

$$= \begin{cases} 0 \text{ if } x \in \begin{bmatrix} 0 & 0 & * \\ ¢ & 0 & * \end{bmatrix} \\ 1 \text{ if } x \in \begin{bmatrix} 0 & 0 & * \\ ¢ & 1 & * \end{bmatrix} \\ 2 \text{ if } x \in \begin{bmatrix} 1 & 0 & * \\ ¢ & * & * \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & * \\ ¢ & * & * \end{bmatrix} \\ 3 \text{ if } x \in \begin{bmatrix} 1 & 1 & * \\ ¢ & * & * \end{bmatrix}, \end{cases}$$

$$\xi_{12}\kappa_2(x) = \begin{cases} 4 \text{ if } x_{13} = x_{23} = 0 \\ 5 \text{ if } x_{13} = 0 \text{ and } x_{23} = 1 \\ 6 \text{ if } x_{13} = 1 \text{ and } x_{23} = 0 \\ 7 \text{ if } x_{13} = 1 \text{ and } x_{23} = 1 \end{cases}$$

$$= \begin{cases} 4 \text{ if } x \in \begin{bmatrix} * & * & 0 \\ ¢ & * & 0 \end{bmatrix} \\ 5 \text{ if } x \in \begin{bmatrix} * & * & 0 \\ ¢ & * & 1 \end{bmatrix} \\ 6 \text{ if } x \in \begin{bmatrix} * & * & 1 \\ ¢ & * & 0 \end{bmatrix} \\ 7 \text{ if } x \in \begin{bmatrix} * & * & 1 \\ ¢ & * & 1 \end{bmatrix} \end{cases}.$$

The inverses of $\xi_{11}\kappa_2$ and $\xi_{12}\kappa_2$ can then be stated as below:

$$(\xi_{11}\kappa_2)^{-1}(y_{12}) = \begin{cases} \begin{bmatrix} 0 & 0 & * \\ ¢ & 0 & * \end{bmatrix} & \text{if } y_{11} = 0 \\ \begin{bmatrix} 0 & 0 & * \\ ¢ & 1 & * \end{bmatrix} & \text{if } y_{11} = 1 \\ \begin{bmatrix} 1 & 0 & * \\ ¢ & * & * \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & * \\ ¢ & * & * \end{bmatrix} & \text{if } y_{11} = 2 \\ \begin{bmatrix} 1 & 1 & * \\ ¢ & * & * \end{bmatrix} & \text{if } y_{11} = 3 \end{cases}$$

$$(\xi_{12}\kappa_2)^{-1}(y_{12}) = \begin{cases} \begin{bmatrix} * & * & 0 \\ ¢ & * & 0 \end{bmatrix} & \text{if } y_{12} = 4 \\ \begin{bmatrix} * & * & 0 \\ ¢ & * & 1 \end{bmatrix} & \text{if } y_{12} = 5 \\ \begin{bmatrix} * & * & 1 \\ ¢ & * & 0 \end{bmatrix} & \text{if } y_{12} = 6 \\ \begin{bmatrix} * & * & 1 \\ ¢ & * & 1 \end{bmatrix} & \text{if } y_{12} = 7. \end{cases}$$

From Section 3.5.4.1, we note that

$$\kappa_2(y_c) = (\xi_{11}\kappa_2)^{-1}(y_{11}) \cap (\xi_{12}\kappa_2)^{-1}(y_{12})$$

where $y_c = \begin{bmatrix} y_{11} & y_{12} \end{bmatrix}$. For example,

$$\kappa_2^{-1}([0 \quad 4]) = (\xi_{11}\kappa_2)^{-1}(0) \cap (\xi_{12}\kappa_2)^{-1}(4)$$

$$= \begin{bmatrix} 0 & 0 & * \\ ¢ & 0 & * \end{bmatrix} \cap \begin{bmatrix} * & * & 0 \\ ¢ & * & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix}.$$

The inverse κ_2^{-1} is shown in Table 4.

Next, we must determine

$$\gamma_2^{-1}(a) = \{(\kappa_2^{-1}(y_c), y_b) \mid \gamma_1(y) = a\}$$

for all $a \in A$. As an illustration, consider $\gamma_2^{-1}(a_0)$. From Table 3,

$$\gamma_1^{-1}(a_0) = \begin{bmatrix} 0 & 4 \\ * & ¢ \end{bmatrix} \cup \begin{bmatrix} 0 & 5 \\ 0 & ¢ \end{bmatrix} \cup \begin{bmatrix} 1 & \{4,5\} \\ 0 & ¢ \end{bmatrix}.$$

Then for each y_c in $\gamma_1^{-1}(a_0)$ we find from Table 4:

$$\kappa_2^{-1}([0 \quad 4]) = \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix},$$

$$\kappa_2^{-1}([0 \quad 5]) = \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 1 \end{bmatrix},$$

$$\kappa_2^{-1}([1 \quad \{4,5\}]) = (\xi_{11}\kappa_2)^{-1}(1) \cap \left((\xi_{12}\kappa_2)^{-1}(4) \cup (\xi_{12}\kappa_2)^{-1}(5)\right)$$

$$= \begin{bmatrix} 0 & 0 & * \\ ¢ & 1 & * \end{bmatrix} \cap \left(\begin{bmatrix} * & * & 0 \\ ¢ & * & 0 \end{bmatrix} \cup \begin{bmatrix} * & * & 0 \\ ¢ & * & 1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & * \end{bmatrix}.$$

Hence,

$$\gamma_2^{-1}(a_0) = \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix} \times [* \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 1 \end{bmatrix} \times [0 \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & * \end{bmatrix} \times [0 \quad ¢]\right)$$

where $\times$ denotes the Cartesian (cross) product, e.g.,

$$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix} \times [* \quad ¢] = \left\{\left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix}, [0 \quad ¢]\right), \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix}, [1 \quad ¢]\right)\right\}.$$

Table 5 gives the complete relation γ_2^{-1}.

y_c	$\kappa_2^{-1}(y_c) \subseteq X$
[0 4]	$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix}$
[0 5]	$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 1 \end{bmatrix}$
[0 6]	$\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 0 & 0 \end{bmatrix}$
[0 7]	$\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 0 & 1 \end{bmatrix}$
[1 4]	$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & 0 \end{bmatrix}$
[1 5]	$\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & 1 \end{bmatrix}$
[1 6]	$\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 1 & 0 \end{bmatrix}$
[1 7]	$\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 1 & 1 \end{bmatrix}$
[2 4]	$\begin{bmatrix} 1 & 0 & 0 \\ ¢ & * & 0 \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & 0 \\ ¢ & * & 0 \end{bmatrix}$
[2 5]	$\begin{bmatrix} 1 & 0 & 0 \\ ¢ & * & 1 \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & 0 \\ ¢ & * & 1 \end{bmatrix}$
[2 6]	$\begin{bmatrix} 1 & 0 & 1 \\ ¢ & * & 0 \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & 1 \\ ¢ & * & 0 \end{bmatrix}$
[2 7]	$\begin{bmatrix} 1 & 0 & 1 \\ ¢ & * & 1 \end{bmatrix} \cup \begin{bmatrix} 0 & 1 & 1 \\ ¢ & * & 1 \end{bmatrix}$
[3 4]	$\begin{bmatrix} 1 & 1 & 0 \\ ¢ & * & 0 \end{bmatrix}$
[3 5]	$\begin{bmatrix} 1 & 1 & 0 \\ ¢ & * & 1 \end{bmatrix}$
[3 6]	$\begin{bmatrix} 1 & 1 & 1 \\ ¢ & * & 0 \end{bmatrix}$
[3 7]	$\begin{bmatrix} 1 & 1 & 1 \\ ¢ & * & 1 \end{bmatrix}$

Table 4
Preimages of κ_2

$a \in A$	$\kappa_2^{-1}(a) \subseteq X \times Y_c$
a_0	$\left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 0 \end{bmatrix} \times [* \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & 1 \end{bmatrix} \times [0 \quad ¢]\right)$
	$\cup \left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 0 & * \end{bmatrix} \times [0 \quad ¢]\right)$
a_1	$\left(\begin{bmatrix} 1 & 0 & 0 \\ ¢ & * & * \end{bmatrix} \times [0 \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 1 & 0 \\ ¢ & * & * \end{bmatrix} \times [0 \quad ¢]\right)$
	$\cup \left(\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 0 & 0 \end{bmatrix} \times [* \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 0 & 1 \end{bmatrix} \times [0 \quad ¢]\right)$
	$\cup \left(\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 1 & * \end{bmatrix} \times [0 \quad ¢]\right)$
a_2	$\left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & * \end{bmatrix} \times [1 \quad ¢]\right)$
a_3	$\left(\begin{bmatrix} 1 & 0 & 0 \\ ¢ & * & * \end{bmatrix} \times [1 \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 1 & 0 \\ ¢ & * & * \end{bmatrix} \times [1 \quad ¢]\right)$
	$\cup \left(\begin{bmatrix} 0 & 0 & 1 \\ ¢ & 1 & * \end{bmatrix} \times [1 \quad ¢]\right)$
a_4	$\left(\begin{bmatrix} 1 & 1 & * \\ ¢ & * & * \end{bmatrix} \times [* \quad ¢]\right) \cup \left(\begin{bmatrix} 1 & 0 & 1 \\ ¢ & * & * \end{bmatrix} \times [* \quad ¢]\right)$
	$\cup \left(\begin{bmatrix} 0 & 1 & 1 \\ ¢ & * & * \end{bmatrix} \times [* \quad ¢]\right) \cup \left(\begin{bmatrix} 0 & 0 & * \\ ¢ & 0 & 1 \end{bmatrix} \times [1 \quad ¢]\right)$

Table 5

Preimages of γ_2

3.5.6 Bottom Level Models

To this point, we have presented the foundation for the effectiveness evaluation of various computers for a simple mission. We now introduce several computer models, and then using the techniques described in Sections 3.4.2-3, determine their performability.

We emphasize that any computer model could now be placed in the hierarchy and evaluated for its performability in the given mission, provided that a suitable translation κ_3 is constructed. The computer models which illustrate this example analysis have been chosen because of their simplicity, their diversity, and the fact that they are based on the same building blocks. This latter quality makes comparison of various configurations easier.

Three computers will be evaluated and compared. Each will be composed of four processor modules, where each module has processing power P (i.e., has the ability to do P work units of usable computation per unit time). Also, each module fails independently with a Poisson distribution having a constant failure rate λ (thus, Pr (a given module fails during an interval of length T) = $1 - e^{-\lambda T}$). We assume every module has sufficient internal checking so that a module can diagnose itself as failed. This could be accomplished for example if every module were composed of components in a triple module redundant (TMR) configuration. If a module fails, the P units of processing power associated with that module are lost. Finally, we make the

assumption that 2P units of processing power are required to perform fuel regulation computations during the take-off and cruise phases, 1P unit is required to perform the fuel regulation computations during the landing phase, while 1P unit is needed for autoland checkout and preparation (i.e., availability) during the last portion of the cruise phase, and 2P units of processing power are required to perform autoland computations during the landing phase. In summary:

		Phases		
		(Take-off, first part of cruise)	(Second part of cruise)	(Landing)
Tasks	Fuel Regulation Computations	2P	2P	P
	Autoland Computations	0	1P	2P

Required Processing Power.

The three computer models will be denoted:

S_1 Dedicated Processor Model,

S_2 Dedicated Group Processor Model,

S_3 Gracefully Degrading Processor Model.

In S_1, each of the four processors is dedicated at all times to a given task. S_2 configures the four processors into two groups of two processors; these groups are then dedicated to a given task. Finally, S_3 allows any processor to perform any task, with a priority system specifying the particular tasks to be done.

The following sections describe the models in detail and give the κ_3 translations. Also, some model simplification via lumping is performed and is reflected in the construction of appropriate interphase transition matrices (H-matrices).

The four processor modules are denoted M_1, M_2, M_3, and M_4. The phases for the bottom level are the same as for the computational task level, i.e.,

$$T^4 = \{T_H, T_L, T\}.$$

3.5.6.1 Dedicated Processor Model

The first computer to be considered, S_1, consists of four modules, each dedicated to a computational task: two for fuel regulation computations and two for autoland computations. During the take-off and cruise phases, the configuration is as follows:

S_1, Phases 1 and 2:

$\boxed{M_1}$, $\boxed{M_2}$ } Fuel regulation computations

$\boxed{M_3}$ Autoland checkout and preparation

$\boxed{M_4}$ Inactive .

Here, modules M_1 and M_2 are dedicated to fuel regulation computations while module M_3 is reserved for autoland availability activities. Module M_4 is inactive. The landing phase configuration is

S_1, Phase 3:

$\boxed{M_1}$ Inactive

$\boxed{M_2}$ Fuel regulation computations

$\boxed{M_3}$, $\boxed{M_4}$ } Autoland computations .

In this phase, module M_1 is no longer needed and so is inactive. M_2 performs fuel regulation computations, while M_3 and M_4 do the autoland computations.

Because these modules are dedicated, there is no reconfiguration if one module should fail. Let us write the state of the system as the collection of unfailed modules and enclose the set with angle brackets. (This is done to prevent confusion with "{" and "}" .) For example, if no modules are failed, then the system state is $\langle M_1, M_2, M_3, M_4 \rangle$. The computer S_1 can then be represented during all phases as the Markov model denoted by the transition diagram in Figure 4. These states are Q_3 for S_1.

However, note that during phase 1, we are unconcerned with M_3 and M_4, during phase 2 we do not care about M_4, and during phase 3 we are unconcerned with M_1. Thus we can significantly reduce the state space of the model by not considering M_3 or M_4 during phase 1, M_4 during phase 2, or M_1 during phase 3. Accordingly, the transition diagrams applicable during the three phases are shown in Figure 5. (See Section 3.4.3 on model simplification.)

To account for the possible failures of modules not examined during some phase, two interphase transition matrices H(1) and H(2) must be constructed (see Section 3.4.3.) Indeed, these are easily defined as follows. Since each module fails independently of the others, we need introduce only the probability that a module fails during the interval it is not observed. Let T_1 and T_2 be the lengths of phases 1 and 2 respectively. Then the probability that a module fails by the end of phase 1 given it was

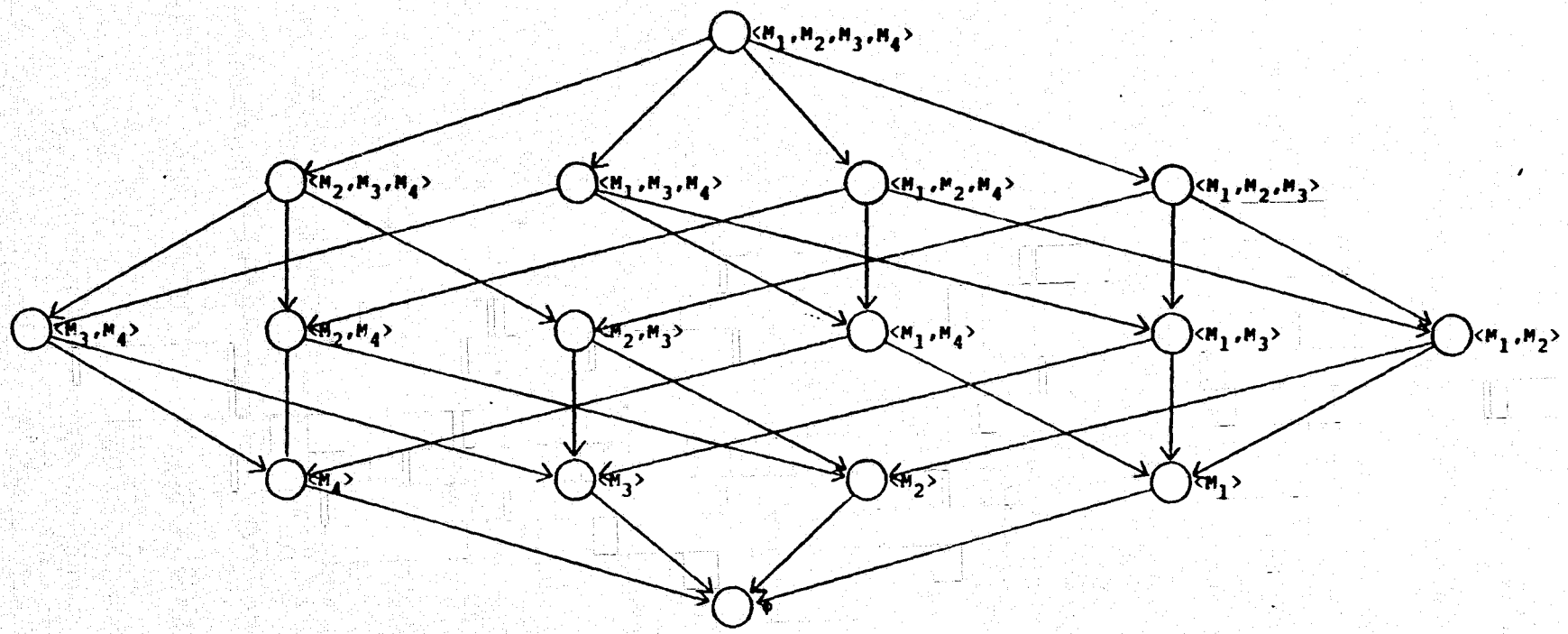
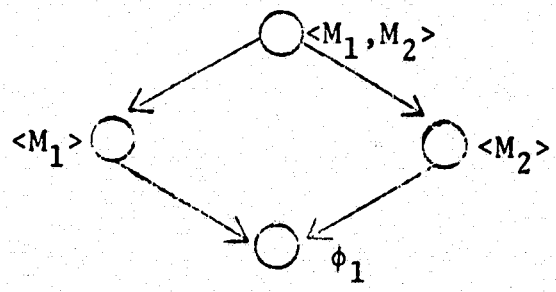
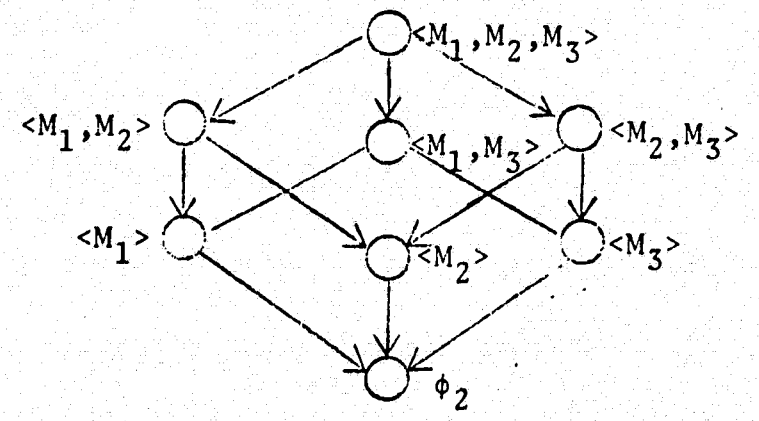


Figure 4

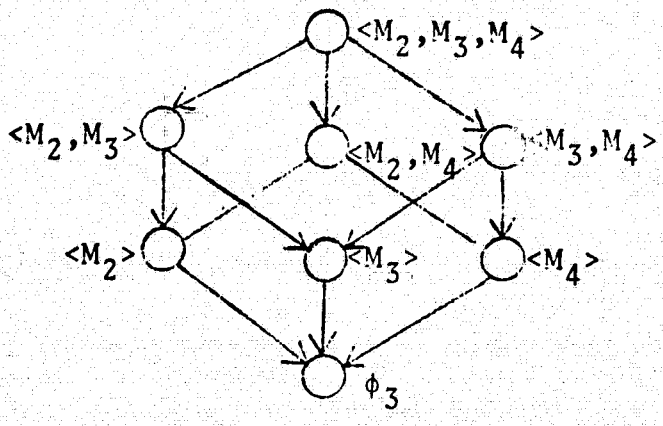
Markov Model Transition Diagram For the Dedicated Processor Model S_1. Each Transition Has Transition Rate λ.



a) Phase 1



b) Phase 2



c) Phase 3

Figure 5

Reduced Markov model transition diagrams for S_1 for phases 1, 2 and 3. Each transition has transition rate λ.

good at the beginning of phase 1 is $1-e^{-\lambda T_1}$. Similarly, $1-e^{-\lambda T_2}$ denotes the probability that a module fails by the end of phase 2 and $1-e^{-\lambda(T_1 + T_2)}$ is the probability that a module fails by the end of phase 2 given it was good at the beginning of phase 1. Thus, we can write H(1) and H(2) as shown in Figure 6. Note that H(1) is conditioned on S_1 initially being in state $\langle M_1,M_2,M_3,M_4\rangle$.

Each entry in the H matrix denotes the probability of transfering from some state in phase i (listed on the left hand side of the matrix) to some state in phase i + 1 (listed below the matrix). For example, we see that if S_1 is in state $\langle M_1,M_3\rangle$ at the end of phase 2, we then transfer to state $\langle M_3,M_4\rangle$ with probability $e^{-\lambda(T_1 + T_2)}$ (i.e., with the probability that M_4 has not failed during phases 1 and 2) and to state $\langle M_3\rangle$ with probability $1-e^{-\lambda(T_1 + T_2)}$ (i.e., with the probability that M_4 has failed).

Next we can specify the transition matrices P(1), P(2), and P(3) for each phase utilizing the transition diagrams in Figure 5. The P(i) appear in Figure 7. Again, each element of P(i) represents the probability of transfering from the state listed along the left hand column to the state listed below the bottom row during the phase.

3.5.6.2 Dedicated Group Processor Model

For the second computer S_2, we again have four modules, and again connect them such that two are dedicated to fuel regulation computations and two are dedicated to autoland computations. However, within the two groups, if one processor fails, the second processor can take over the first processor's function if the second processor is inactive. Hence, during all phases

$$
H(1) = \begin{array}{c} \langle M_1,M_2\rangle \\ \langle M_1\rangle \\ \langle M_2\rangle \\ \phi_1 \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-\lambda T_1} & 1-e^{-\lambda T_1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & e^{-\lambda T_1} & 0 & 1-e^{-\lambda T_1} & 0 & 0 & 0 \\
0 & 0 & 0 & e^{-\lambda T_1} & 0 & 1-e^{-\lambda T_1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\lambda T_1} & 1-e^{-\lambda T_1}
\end{bmatrix} \\
\begin{array}{cccccccc}
\langle M_1,M_2,M_3\rangle & \langle M_1,M_2\rangle & \langle M_1,M_3\rangle & \langle M_2,M_3\rangle & \langle M_1\rangle & \langle M_2\rangle & \langle M_3\rangle & \phi_2
\end{array}
\end{array}
$$

$$
H(2) = \begin{array}{c} \langle M_1,M_2,M_3\rangle \\ \langle M_1,M_2\rangle \\ \langle M_1,M_3\rangle \\ \langle M_2,M_3\rangle \\ \langle M_1\rangle \\ \langle M_2\rangle \\ \langle M_3\rangle \\ \phi_2 \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-\lambda(T_1+T_2)} & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & e^{-\lambda(T_1+T_2)} & 0 & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 & 0 \\
0 & 0 & 0 & e^{-\lambda(T_1+T_2)} & 0 & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 \\
e^{-\lambda(T_1+T_2)} & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\lambda(T_1+T_2)} & 1-e^{-\lambda(T_1+T_2)} \\
0 & 0 & e^{-\lambda(T_1+T_2)} & 0 & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 & 0 \\
0 & 0 & 0 & e^{-\lambda(T_1+T_2)} & 0 & 1-e^{-\lambda(T_1+T_2)} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\lambda(T_1+T_2)} & 1-e^{-\lambda(T_1+T_2)}
\end{bmatrix} \\
\begin{array}{cccccccc}
\langle M_2,M_3,M_4\rangle & \langle M_2,M_3\rangle & \langle M_2,M_4\rangle & \langle M_3,M_4\rangle & \langle M_2\rangle & \langle M_3\rangle & \langle M_4\rangle & \phi_3
\end{array}
\end{array}
$$

Figure 6

Interphase Transition Matrices for the Dedicated Processor Model

ORIGINAL PAGE IS
OF POOR QUALITY

$$
P(1) = \begin{array}{c} \langle M_1,M_2\rangle \\ \langle M_1\rangle \\ \langle M_2\rangle \\ \phi_1 \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-2\lambda T_1} & e^{-\lambda T_1}(1-e^{-\lambda T_1}) & e^{-\lambda T_1}(1-e^{-\lambda T_1}) & \left(1-e^{-\lambda T_1}\right)^2 \\
0 & e^{-\lambda T_1} & 0 & 1-e^{-\lambda T_1} \\
0 & 0 & e^{-\lambda T_1} & 1-e^{-\lambda T_1} \\
0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{cccc} \langle M_1,M_2\rangle & \langle M_1\rangle & \langle M_2\rangle & \phi_1 \end{array}
\end{array}
$$

$$
P(2) = \begin{array}{c} \langle M_1,M_2,M_3\rangle \\ \langle M_1,M_2\rangle \\ \langle M_1,M_3\rangle \\ \langle M_2,M_3\rangle \\ \langle M_1\rangle \\ \langle M_2\rangle \\ \langle M_3\rangle \\ \phi_2 \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-3\lambda T_2} & e^{-2\lambda T_2}(1-e^{-\lambda T_2}) & e^{-2\lambda T_2}(1-e^{-\lambda T_2}) & e^{-2\lambda T_2}(1-e^{-\lambda T_2}) & e^{-\lambda T_2}\left(1-e^{-\lambda T_2}\right)^2 & e^{-\lambda T_2}\left(1-e^{-\lambda T_2}\right)^2 & e^{-\lambda T_2}\left(1-e^{-\lambda T_2}\right)^2 & \left(1-e^{-\lambda T_2}\right)^3 \\
0 & e^{-2\lambda T_2} & 0 & 0 & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & 0 & \left(1-e^{-\lambda T_2}\right)^2 \\
0 & 0 & e^{-2\lambda T_2} & 0 & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & 0 & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & \left(1-e^{-\lambda T_2}\right)^2 \\
0 & 0 & 0 & e^{-2\lambda T_2} & 0 & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & e^{-\lambda T_2}(1-e^{-\lambda T_2}) & \left(1-e^{-\lambda T_2}\right)^2 \\
0 & 0 & 0 & 0 & e^{-\lambda T_2} & 0 & 0 & 1-e^{-\lambda T_2} \\
0 & 0 & 0 & 0 & 0 & e^{-\lambda T_2} & 0 & 1-e^{-\lambda T_2} \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\lambda T_2} & 1-e^{-\lambda T_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{cccccccc} \langle M_1,M_2,M_3\rangle & \langle M_1,M_2\rangle & \langle M_1,M_3\rangle & \langle M_2,M_3\rangle & \langle M_1\rangle & \langle M_2\rangle & \langle M_3\rangle & \phi_2 \end{array}
\end{array}
$$

$$
P(3) = \begin{array}{c} \langle M_2,M_3,M_4\rangle \\ \langle M_2,M_3\rangle \\ \langle M_2,M_4\rangle \\ \langle M_3,M_4\rangle \\ \langle M_2\rangle \\ \langle M_3\rangle \\ \langle M_4\rangle \\ \phi_3 \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-3\lambda T_3} & e^{-2\lambda T_3}(1-e^{-\lambda T_3}) & e^{-2\lambda T_3}(1-e^{-\lambda T_3}) & e^{-2\lambda T_3}(1-e^{-\lambda T_3}) & e^{-\lambda T_3}\left(1-e^{-\lambda T_3}\right)^2 & e^{-\lambda T_3}\left(1-e^{-\lambda T_3}\right)^2 & e^{-\lambda T_3}\left(1-e^{-\lambda T_3}\right)^2 & \left(1-e^{-\lambda T_3}\right)^3 \\
0 & e^{-2\lambda T_3} & 0 & 0 & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & 0 & \left(1-e^{-\lambda T_3}\right)^2 \\
0 & 0 & e^{-2\lambda T_3} & 0 & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & 0 & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & \left(1-e^{-\lambda T_3}\right)^2 \\
0 & 0 & 0 & e^{-2\lambda T_3} & 0 & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & e^{-\lambda T_3}(1-e^{-\lambda T_3}) & \left(1-e^{-\lambda T_3}\right)^2 \\
0 & 0 & 0 & 0 & e^{-\lambda T_3} & 0 & 0 & 1-e^{-\lambda T_3} \\
0 & 0 & 0 & 0 & 0 & e^{-\lambda T_3} & 0 & 1-e^{-\lambda T_3} \\
0 & 0 & 0 & 0 & 0 & 0 & e^{-\lambda T_3} & 1-e^{-\lambda T_3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{cccccccc} \langle M_2,M_3,M_4\rangle & \langle M_2,M_3\rangle & \langle M_2,M_4\rangle & \langle M_3,M_4\rangle & \langle M_2\rangle & \langle M_3\rangle & \langle M_4\rangle & \phi_3 \end{array}
\end{array}
$$

Figure 7

State Transition Matrices for the Dedicated Processor Model

the configuration looks as follows:

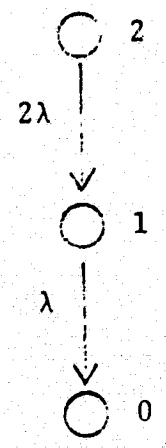
S_2:

M_1, M_2 — Fuel regulation computations

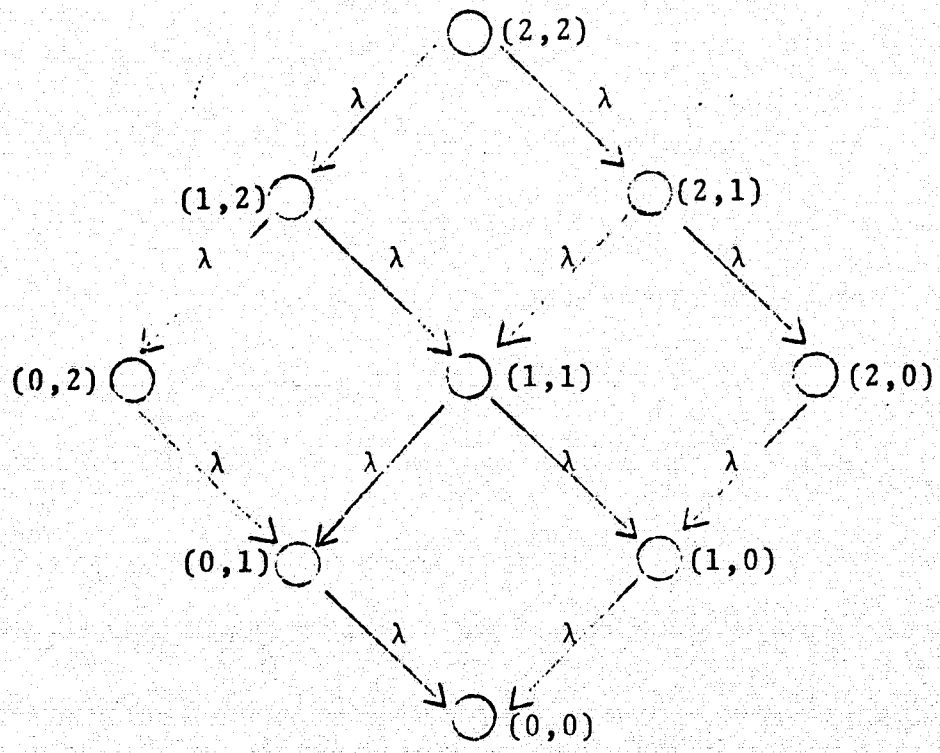
M_3, M_4 — Autoland checkout, preparation, and computations.

If we again write the model's state as the set of unfailed modules, then the Markov model transition diagram for S_2 is the same as shown in Figure 4. However, once again during different phases we are unconcerned with various phases of the modules. Thus, during phase 1 we do not care about M_3 or M_4, during phase 2 we are unconcerned with which of M_3 or M_4 is operational, while in phase 3 we are unconcerned with which of M_1 or M_2 is working. Hence, let us write the state of the model in phase 1 as the number of fuel regulation modules working f, $f \in \{0,1,2\}$. Also, the state of model in phases 2 and 3 will be written as the ordered pair (f,a), where $f \in \{0,1,2\}$ is the number of working fuel regulation modules and $a \in \{0,1,2\}$ is the number of working autoland modules. The Markov diagrams for phases 1,2, and 3 are shown in Figure 8. The model for phases 2 and 3 is not easily reducible since we must keep track of the number of functioning units at all times to determine when none are left.

Because the states of phase 2 and phase 3 are identical and no reconfiguration occurs between the phases, the interphase transition matrix H(2) is the 9 x 9 identity matrix. The inter-



a) Phase 1



b) Phases 2 and 3

Figure 8

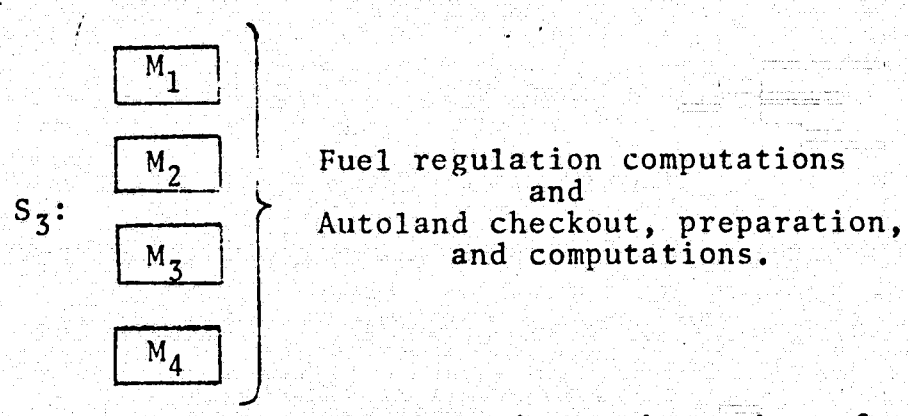
Markov Model Transition Diagram for S_2.

phase transition matrix between phases 1 and 2 must take into account the probabilities that one or both of the autoland modules have failed. H(1) is conditioned on S_2 being initially in state (2,2). Figure 9 shows H(1) and H(2).

The transition matrices P(1), P(2) and P(3) are given in Figure 10.

3.5.6.3 Gracefully Degrading Processor Model

The third computer to be discussed, S_3, is once more comprised of 4 modules, but with no specific processor assignments. Any processor can perform any other processor's task. One processor will be used to help co-ordinate the other processors, and if necessary, can be used as a spare. The configuration during both phases is thus:



The state of this system is then simply the number of processors working. Hence, the state ranges from 0 to 4. Figure 11 shows the general Markov model transition diagram for S_3. $Q^3 = \{0,1,2,3,4\}$.

Although the transition diagram could be reduced by examination of the computer task requirements and by a proper choice of the interphase transition matrices, we choose not to do that at this time. Hence the model represented in Figure 11 will be used uniformly in all three phases. Furthermore, since no hardware

$$
H(1) \quad \begin{array}{c} 2 \\ 1 \\ 3 \end{array}
\begin{bmatrix}
e^{-2\lambda T_1} & 0 & 2e^{-\lambda T_1}(1-e^{-\lambda T_1}) & 0 & 0 & \left(1-e^{-\lambda T_1}\right)^2 & 0 & 0 & 0 \\
0 & e^{-2\lambda T_1} & 0 & 0 & 2e^{-\lambda T_1}(1-e^{-\lambda T_1}) & 0 & 0 & \left(1-e^{-\lambda T_1}\right)^2 & 0 \\
0 & 0 & 0 & e^{-2\lambda T_1} & 0 & 0 & 2e^{-\lambda T_1}(1-e^{-\lambda T_1}) & 0 & \left(1-e^{-\lambda T_1}\right)^2
\end{bmatrix}
$$

$$
\begin{array}{cc}
 & \begin{array}{ccccccccc} (2,2) & (1,2) & (2,1) & (0,2) & (1,1) & (2,0) & (0,1) & (1,0) & (0,0) \end{array} \\
H(2) = \begin{array}{c} (2,2) \\ (1,2) \\ (2,1) \\ (0,2) \\ (1,1) \\ (2,0) \\ (0,1) \\ (1,0) \\ (0,0) \end{array} &
\begin{bmatrix}
1 & & & & & & & & \\
 & 1 & & & & & & & \\
 & & 1 & & & & & & \\
 & & & 1 & & & & & \\
 & & & & 1 & & & & \\
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1
\end{bmatrix}
\end{array}
$$

Figure 9

Interphase Transition Matrices for the Dedicated Group Processor

$$
P(1) = \begin{array}{c} 2 \\ 1 \\ 3 \\ \\ \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-2\lambda T_1} & 2e^{-\lambda T_1}(1-e^{-\lambda T_1}) & \left(1-e^{-\lambda T_1}\right)^2 \\
0 & e^{-\lambda T_1} & 1-e^{-\lambda T_1} \\
0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccc} 2 & 1 & 0 \end{array}
\end{array}
$$

$$
P(2) = \begin{array}{c}
(2,2) \\ (1,2) \\ (2,1) \\ (0,2) \\ (1,1) \\ (2,0) \\ (0,1) \\ (1,0) \\ (0,0) \\ \\
\end{array}
\begin{array}{c}
\begin{bmatrix}
p^4 & p^3q & p^3q & p^2q^2 & 4p^2q^2 & p^2q^2 & 2pq^3 & 2pq^3 & q^4 \\
0 & p^3 & 0 & p^2q & 2p^2q & 0 & 2pq^2 & pq^2 & q^3 \\
0 & 0 & p^3 & 0 & 0 & p^2q & pq^2 & 2pq^2 & q^3 \\
0 & 0 & 0 & q^2 & 0 & 0 & 2pq & 0 & q^2 \\
0 & 0 & 0 & 0 & p^2 & 0 & pq & pq & q^2 \\
0 & 0 & 0 & 0 & 0 & p^2 & 0 & 2pq & q^2 \\
0 & 0 & 0 & 0 & 0 & 0 & p & 0 & q \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p & q \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccccccccc} (2,2) & (1,2) & (2,1) & (0,2) & (1,1) & (2,0) & (0,1) & (1,0) & (0,0) \end{array}
\end{array}
\qquad
\begin{array}{l} p = e^{-\lambda T_2} \\ q = 1-e^{-\lambda T_2} \end{array}
$$

$$
P(3) = \begin{array}{c}
(2,2) \\ (1,2) \\ (2,1) \\ (0,2) \\ (1,1) \\ (2,0) \\ (0,1) \\ (1,0) \\ (0,0) \\ \\
\end{array}
\begin{array}{c}
\begin{bmatrix}
p^4 & p^3q & p^3q & p^2q^2 & 4p^2q^2 & p^2q^2 & 2pq^3 & 2pq^3 & q^4 \\
0 & p^3 & 0 & p^2q & 2p^2q & 0 & 2pq^2 & pq^2 & q^3 \\
0 & 0 & p^3 & 0 & 0 & p^2q & pq^2 & 2pq^2 & q^3 \\
0 & 0 & 0 & q^2 & 0 & 0 & 2pq & 0 & q^2 \\
0 & 0 & 0 & 0 & p^2 & 0 & pq & pq & q^2 \\
0 & 0 & 0 & 0 & 0 & p^2 & 0 & 2pq & q^2 \\
0 & 0 & 0 & 0 & 0 & 0 & p & 0 & q \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p & q \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccccccccc} (2,2) & (1,2) & (2,1) & (0,2) & (1,1) & (2,0) & (0,1) & (1,0) & (0,0) \end{array}
\end{array}
\qquad
\begin{array}{l} p = e^{-\lambda T_3} \\ q = 1-e^{-\lambda T_3} \end{array}
$$

Figure 10

Transition Matrices for the Dedicated Group Processor

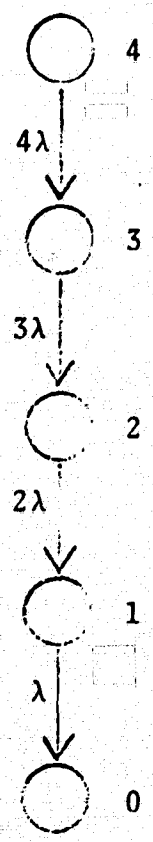


Figure 11

General Markov Model Transition Diagram for S_3.

reconfiguration occurs between phases, the interphase transition matrices are the 5 x 5 identity matrix:

$$H(1) = H(2) = \begin{bmatrix} 1 & & & & \\ & 1 & & 0 & \\ & & 1 & & \\ & 0 & & 1 & \\ & & & & 1 \end{bmatrix}.$$

Also, the transition matrices P(1), P(2), and P(3) are the same and are shown in Figure 12.

For each state within each phase, we establish a set of tasks, the <u>task set</u>, which will be performed within that state. This set is necessary to allocate the available processing resources to the highest priority computations that are achievable. In particular, if a processor performing a highly important task fails, then a processor performing a less important task must assume the computations of the first, more important, task. Also, situations could arise such that insufficient processing power may be available to perform high priority tasks. In this case, lower priority tasks are then performed.

Specifying a set of tasks to be operating in each state resolves these problems. In general, the set is based both on the number of processing units required to accomplish each task in the set as well as on some task cost function. For this example, an appropriate cost function is based on a simple priority list for each phase. For phase 1, the priority ordering is:

1) Fuel regulation computations
2) Coordination and spare,

$$
P(1) = \begin{array}{c} 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ \\ \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-4\lambda T_1} & 4e^{-3\lambda T_1}(1-e^{-\lambda T_1}) & 6e^{-2\lambda T_1}\left(1-e^{-\lambda T_1}\right)^2 & 4e^{-\lambda T_1}\left(1-e^{-\lambda T_1}\right)^3 & \left(1-e^{-\lambda T_1}\right)^4 \\
0 & e^{-3\lambda T_1} & 3e^{-2\lambda T_1}(1-e^{-\lambda T_1}) & 3e^{-\lambda T_1}\left(1-e^{-\lambda T_1}\right)^2 & \left(1-e^{-\lambda T_1}\right)^3 \\
0 & 0 & e^{-2\lambda T_1} & 2e^{-\lambda T_1}(1-e^{-\lambda T_1}) & \left(1-e^{-\lambda T_1}\right)^2 \\
0 & 0 & 0 & e^{-\lambda T_1} & 1-e^{-\lambda T_1} \\
0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \end{array}
\end{array}
$$

$$
P(2) = \begin{array}{c} 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ \\ \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-4\lambda T_2} & 4e^{-3\lambda T_2}(1-e^{-\lambda T_2}) & 6e^{-2\lambda T_2}\left(1-e^{-\lambda T_2}\right)^2 & 4e^{-\lambda T_2}\left(1-e^{-\lambda T_2}\right)^3 & \left(1-e^{-\lambda T_2}\right)^4 \\
0 & e^{-3\lambda T_2} & 3e^{-2\lambda T_2}(1-e^{-\lambda T_2}) & 3e^{-\lambda T_2}\left(1-e^{-\lambda T_2}\right)^2 & \left(1-e^{-\lambda T_2}\right)^3 \\
0 & 0 & e^{-2\lambda T_2} & 2e^{-\lambda T_2}(1-e^{-\lambda T_2}) & \left(1-e^{-\lambda T_2}\right)^2 \\
0 & 0 & 0 & e^{-\lambda T_2} & 1-e^{-\lambda T_2} \\
0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \end{array}
\end{array}
$$

$$
P(3) = \begin{array}{c} 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ \\ \end{array}
\begin{array}{c}
\begin{bmatrix}
e^{-4\lambda T_2} & 4e^{-3\lambda T_3}(1-e^{-\lambda T_3}) & 6e^{-2\lambda T_3}\left(1-e^{-\lambda T_3}\right)^2 & 4e^{-\lambda T_3}\left(1-e^{-\lambda T_3}\right)^3 & \left(1-e^{-\lambda T_3}\right)^4 \\
0 & e^{-3\lambda T_2} & 3e^{-2\lambda T_3}(1-e^{-\lambda T_3}) & 3e^{-\lambda T_3}\left(1-e^{-\lambda T_3}\right)^2 & \left(1-e^{-\lambda T_3}\right)^3 \\
0 & 0 & e^{-2\lambda T_2} & 2e^{-\lambda T_3}(1-e^{-\lambda T_3}) & \left(1-e^{-\lambda T_3}\right)^2 \\
0 & 0 & 0 & e^{-\lambda T_2} & 1-e^{-\lambda T_3} \\
0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\begin{array}{ccccc} 4 & 3 & 2 & 1 & 0 \end{array}
\end{array}
$$

Figure 12

State Transition Martices for the Gracefully Degrading Processor Model

for phase 2 the list is

1) Fuel regulation computations
2) Autoland checkout and preparation
3) Coordination and spare,

and for phase 3 the ordering is

1) Autoland computations
2) Fuel regulation computations
3) Coordination and spare.

The earlier in the list a task is named, the higher its priority.

For a given state and phase, we choose the tasks to be performed according to the following algorithm. Starting with the highest priority task, each task is examined in turn. If enough resources are available to perform both the examined task and all previously chosen tasks, then the examined task is also chosen. Using this rule, the task sets for each state during each phase are shown in Table 6.

3.5.7 Capability Function Preimage Sets

We are now prepared to derive the capability function preimage sets γ^{-1} for each of the three bottom computational hardware level models. First the interlevel translations κ_3 between the bottom level and the computational task level are determined. These are then combined with the preimage sets of γ_3 (Section 3.5.5) using the algorithm of Section 3.5.4.1 to arrive at γ^{-1}.

Each of the three bottom levels will be represented by a three-variable random process $w = \{X^3_{T_H}, X^3_{T_L}, X^3_T\}$ taking values in

Phase	State	Task Set
1	4	Fuel regulation computations Coordination and spare
	3	Fuel regulation computations Coordination and spare
	2	Fuel regulation computations
	1	None
	0	None
2	4	Fuel regulation computations Autoland checkout and preparation Coordination and spare
	3	Fuel regulation computations Autoland checkout and preparation
	2	Fuel regulation computations
	1	Autoland checkout and preparation
	0	None
3	4	Autoland computations Fuel regulation computations Coordination and spare
	3	Autoland computations Fuel regulation computations
	2	Autoland computations
	1	Fuel regulation computations
	0	None

Table 6

Task Sets for S_3 by Phase and State.

the state spaces Q^3 described in the preceeding sections. The trajectory space for each model is

$$W = U^3 = Q^3 \times Q^3 \times Q^3 ,$$

and in our array notation, a trajectory $w \varepsilon W$ is written

$$w = [w_1 \quad w_2 \quad w_3] \quad .$$

The interlevel translation $\kappa_3 : W \rightarrow X_c$ can be decomposed into

$$\xi_{11}\kappa_3 : W \rightarrow X_{11}$$

$$\xi_{12}\kappa_3 : W \rightarrow X_{12}$$

$$\xi_{13}\kappa_3 : W \rightarrow X_{13}$$

$$\xi_{21}\kappa_3 : W \rightarrow X_{21}$$

$$\xi_{22}\kappa_3 : W \rightarrow X_{22}$$

$$\xi_{23}\kappa_3 : W \rightarrow X_{23} \ .$$

The κ are constructed as follows. From Section 3.5.3.3

$$x_{ij} = \begin{cases} 0 \text{ if computation } i \text{ successful at observation } j \\ 1 \text{ otherwise} \end{cases}$$

for $i = 1,2$

$j = 1,2,3$

$j \neq 1$ when $i = 1$

where $i = 1$ fuel regulation computations

$i = 2$ autoland computations, and

$x_{21} = ¢$.

Furthermore, from Section 3.5.6, the required processing power for each phase broken down by task is given as:

Tasks	Phases: (Take-off, first part of cruise)	(Second part of cruise)	(Landing)
Fuel Regulation Computations	2P	2P	P
Autoland Computations	0	1P	2P

Required Processing Power

where P is the processing power of one processor. Every state in each bottom level model in Section 3.5.6 has associated with it the number of processors applied to each task. Hence, given a bottom level state, determining whether a particular computational task is achieved is mechanical, and so the function $\xi_{ij}\kappa_3$ is straightforward, namely

$$\xi_{ij}\kappa_3(w) = \begin{cases} 0 \text{ if } w_i \text{ allocates sufficient processing power to task } j \text{ so that } j \text{ is achieved} \\ 1 \text{ otherwise.} \end{cases}$$

In addition, it is plain that $\xi_{ij}\kappa_3$ has an inverse, $(\xi_{ij}\kappa_3)^{-1}$. From Section 3.5.4.1,

$$\gamma^{-1}(a) = \{(\kappa_3^{-1}(x_c), x_b, y_b) \mid \gamma_2(x, y_b) = a\}.$$

But x has no basic variables since all variables in x are decomposed at the computer hardware level, so $x_b = \Phi$, $x = x_c$, and

$$\gamma^{-1}(a) = \{(\kappa_3^{-1}(x), y_b) \mid \gamma_2(x, y_b) = a\}.$$

In the sections that follow, γ^{-1} is derived for each of the bottom models given in Section 3.5.6.

3.5.7.1 γ^{-1} for the Dedicated Processor Model

For the dedicated processor model S_1, the interlevel translation κ_3 can be seen to be composed of the following $\xi_{ij}\kappa_3$:

$$\xi_{11}\kappa_3(w) = \begin{cases} 0 \text{ if } w_1 = \langle M_1, M_2\rangle \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [\langle M_1,M_2\rangle \ * \ \ *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{12}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_2\rangle\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_2\rangle\} \ *], \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{13}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{\langle M_2,M_3,M_4\rangle, \langle M_2,M_3\rangle, \langle M_2,M_4\rangle, \langle M_2\rangle\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* * \{\langle M_2,M_3,M_4\rangle, \langle M_2,M_3\rangle, \langle M_2,M_4\rangle, \langle M_2\rangle\}] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{21}\kappa_3(w) = \phi \ ,$$

$$\xi_{22}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_3\rangle, \langle M_2,M_3\rangle, \langle M_3\rangle\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_3\rangle, \langle M_2,M_3\rangle, \langle M_3\rangle\} \ *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{23}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{\langle M_2,M_3,M_4\rangle, \langle M_3,M_4\rangle\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* * \langle M_2,M_3,M_4\rangle, \langle M_3,M_4\rangle\}] \\ 1 \text{ otherwise} \ . \end{cases}$$

From these components, we can now write the inverses.

$(\xi_{ij}\kappa_3)^{-1}$:

$$(\xi_{11}\kappa_3)^{-1}(x_{11}) = \begin{cases} [\langle M_1,M_2\rangle \ * \ \ *] \text{ if } x_{11}=0 \\ [\{\langle M_1\rangle, \langle M_2\rangle, \phi_1\} * \ \ *] \text{ if } x_{11}=1, \end{cases}$$

$$(\xi_{12}\kappa_3)^{-1}(x_{12}) = \begin{cases} [* \ \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_2\rangle\} \ *] \text{ if } x_{12}=0 \\ [* \ \{\langle M_1,M_3\rangle, \langle M_2,M_3\rangle, \langle M_1\rangle, \langle M_2\rangle, \\ \ \langle M_3\rangle, \phi_2\} \ \ *] \text{ if } x_{12}=1, \end{cases}$$

$$(\xi_{13}\kappa_3)^{-1}(x_{13}) = \begin{cases} [* \ \ * \ \{\langle M_2,M_3,M_4\rangle, \langle M_2,M_3\rangle, \langle M_2,M_4\rangle, \\ \ \langle M_2\rangle\} \ *] \text{ if } x_{13}=0 \\ [* \ \ * \ \{\langle M_3,M_4\rangle, \langle M_3\rangle, \langle M_4\rangle, \phi_3\}] \text{if } x_{13}=1 \end{cases}$$

$$(\xi_{21}\kappa_3)^{-1}(x_{21}) = [* \ \ * \ \ *],$$

$$(\xi_{22}\kappa_3)^{-1}(x_{22}) = \begin{cases} [* \ \ \{\langle M_1,M_2,M_3\rangle, \langle M_1,M_3\rangle, \langle M_2,M_3\rangle, \\ \ \langle M_3\rangle \ \}] \text{if } x_{22}=0 \\ [* \ \ \{\langle M_1,M_2\rangle, \langle M_1\rangle, \langle M_2\rangle, \phi_2\} \ *] \text{ if } x_{22}=0, \end{cases}$$

$$(\xi_{23}\kappa_3)^{-1}(x_{23}) = \begin{cases} [* \ \ * \ \{\langle M_2,M_3,M_4\rangle, \langle M_3,M_4\rangle\}] \text{ if } x_{23}=0 \\ [* \ \ * \ \{\langle M_2,M_3\rangle, \langle M_2,M_4\rangle, \langle M_2\rangle, \\ \ \langle M_3\rangle, \langle M_4\rangle, \phi_3\}] \text{ if } x_{23}=1. \end{cases}$$

Then, using the method of Section 3.5.4.1, κ_3^{-1} was found. This relation is given in Table 7. Finally, using Table 7 and Table 5, γ^{-1} was found. As an example of the procedure utilized:

$$\begin{aligned} \gamma^{-1}(a_2) &= \{(\kappa_3^{-1}(x), y_b) \mid \gamma(x,y_b) = a_2\} \\ &= \left(\kappa_3^{-1}\left(\begin{bmatrix} 0 & 0 & 0 \\ ¢ & 1 & * \end{bmatrix}\right) \times [1 \quad ¢]\right) \\ &= \left([\langle M_1,M_2\rangle \quad \langle M_1,M_2\rangle \quad \{\langle M_2,M_3,M_4\rangle, \right. \\ &\qquad \left. \langle M_2,M_3\rangle, \langle M_2,M_4\rangle, \langle M_2\rangle\}] \times [1 \quad ¢]\right). \end{aligned}$$

Table 8 lists γ^{-1} for S_1.

x_c	$\kappa_3^{-1}(x_c) \subseteq W$
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 0 \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2,M_3\rangle \quad \langle M_2,M_3,M_4\rangle]$
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 1 \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2,M_3\rangle \quad \{\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}]$
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 1 & * \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2\rangle \quad \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}]$
$\begin{bmatrix} 1 & 0 & 0 \\ \not\in & * & * \end{bmatrix}$	$[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \quad \{\langle M_1,M_2,M_3\rangle,\langle M_1,M_2\rangle\} \quad \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}]$
$\begin{bmatrix} 0 & 1 & 0 \\ \not\in & * & * \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \quad \{\langle M_1,M_2,M_3\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}]$
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 0 \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2,M_3\rangle \quad \langle M_3,M_4\rangle]$
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 1 \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2,M_3\rangle \quad \{\langle M_3\rangle,\langle M_4\rangle,\phi_3\}]$
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 1 & * \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2\rangle \quad \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}]$
$\begin{bmatrix} 1 & 1 & * \\ \not\in & * & * \end{bmatrix}$	$[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \quad \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \quad *]$
$\begin{bmatrix} 1 & 0 & 1 \\ \not\in & * & * \end{bmatrix}$	$[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \quad \{\langle M_1,M_2,M_3\rangle,\langle M_1,M_2\rangle\} \quad \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}]$
$\begin{bmatrix} 0 & 1 & 1 \\ \not\in & * & * \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \quad \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}]$
$\begin{bmatrix} 0 & 0 & * \\ \not\in & 0 & 1 \end{bmatrix}$	$[\langle M_1,M_2\rangle \quad \langle M_1,M_2,M_3\rangle \quad \{\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}]$

Table 7

Preimages of κ_3 of the Dedicated Processor Model

x_c	$\gamma^{-1}(a) \subseteq W \times Y_c$
a_0	$\{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \langle M_2,M_3,M_4\rangle] \times [* \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [0 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [0 \ \ \phi]\}$
a_1	$\{[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \ \ \{\langle M_1,M_2,M_3\rangle,\langle M_1,M_2\rangle\} \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [0 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [0 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \langle M_3,M_4\rangle] \times [* \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_3\rangle,\langle M_4\rangle,\phi_3\}] \times [0 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}] \times [0 \ \ \phi]\}$
a_2	$[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [1 \ \ \phi]$
a_3	$\{[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \ \ \{\langle M_1,M_2,M_3\rangle,\langle M_1,M_2\rangle\} \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [1 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \ \ \{\langle M_2,M_3,M_4\rangle,\langle M_2,M_3\rangle,\langle M_2,M_4\rangle,\langle M_2\rangle\}] \times [1 \ \ \phi]\}$
	$\cup\ \{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}] \times [1 \ \ \phi]\}$
a_4	$\{[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \ \ \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \ \ *] \times [* \ \ \phi]\}$
	$\cup\{[\{\langle M_1\rangle,\langle M_2\rangle,\phi_1\} \ \ \{\langle M_1,M_2,M_3\rangle,\langle M_1,M_2\rangle\} \ \ \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}] \times [* \ \ \phi]\}$
	$\cup\{[\langle M_1,M_2\rangle \ \ \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_2\} \ \ \{\langle M_3,M_4\rangle,\langle M_3\rangle,\langle M_4\rangle,\phi_3\}] \times [* \ \ \phi]\}$
	$\cup\{[\langle M_1,M_2\rangle \ \ \langle M_1,M_2\rangle \ \ \{\langle M_1,M_3\rangle,\langle M_2,M_3\rangle,\langle M_1\rangle,\langle M_2\rangle,\langle M_3\rangle,\phi_3\}] \times [1 \ \ \phi]\}$

Table : 8

Preimages of γ for the Dedicated Processor Model

3.5.7.2 γ^{-1} for the Dedicated Group Processor Model

The derivation of the γ-induced trajectory sets for the dedicated group processor model S_2 is similar to the derivation in Section 3.5.7.1 for the dedicated processor model. The $\xi_{ij}\kappa_3$ are as follows:

$$\xi_{11}\kappa_3(w) = \begin{cases} 0 \text{ if } w_1=2 \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ of } w \in [2 \quad * \quad *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{12}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{(2,2), (2,1), (2,0)\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \quad \{(2,2), (2,1), (2,0)\} \; *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{13}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{(2,2), (2,1), (2,0), \\ \qquad (1,2), (1,1), (1,0)\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \quad * \quad \{(2,2), (2,1), (2,0), \\ \qquad (1,2), (1,1), (1,0)\}] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{21}\kappa_3(w) = \phi ,$$

$$\xi_{22}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{(2,2), (1,2), (0,2) \\ \qquad (2,1), (1,1), (0,1)\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \quad \{(2,2), (1,2), (0,2) \\ \qquad (2,1), (1,1), (0,1)\} \quad *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{23}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{(2,2), (1,2), (0,2)\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \quad * \ \{(2,2), (1,2), (0,2)\}] \\ 1 \text{ otherwise.} \end{cases}$$

The inverses $(\xi_{ij}\kappa_3)^{-1}$ are as below:

$$(\xi_{11}\kappa_3)^{-1}(x_{11}) = \begin{cases} [2 \quad * \quad *] \text{ if } x_{11}=0 \\ [\{0,1\} \ * \quad *] \text{ if } x_{11}=1, \end{cases}$$

$$(\xi_{12}\kappa_3)^{-1}(x_{12}) = \begin{cases} [* \quad \{(2,2), (2,1), (2,0)\} \quad *] \text{ if } x_{12}=0 \\ [* \quad \{(1,2), (1,1), (1,0), \\ \ (0,2), (0,1), (0,0)\} \quad *] \text{ if } x_{12}=1, \end{cases}$$

$$(\xi_{13}\kappa_3)^{-1}(x_{13}) = \begin{cases} [* \quad * \quad \{(2,2), (2,1), (2,0), \\ \ (1,2), (1,1), (1,0)\}] \text{ if } x_{13}=0 \\ [* \quad * \quad \{(0,2), (0,1), (0,0)\}] \text{ if } x_{13}=1, \end{cases}$$

$$(\xi_{21}\kappa_2)^{-1}(x_{21}) = [* \quad * \quad *],$$

$$(\xi_{22}\kappa_3)^{-1}(x_{22}) = \begin{cases} [* \quad \{(2,2), (1,2), (0,2), \\ \ (2,1), (1,1), (0,1)\} \quad *] \text{ if } x_{22}=0 \\ [* \quad \{(2,0), (1,0), (0,0)\} \quad *] \text{ if } x_{22}=1, \end{cases}$$

$$(\xi_{23}\kappa_3)^{-1}(x_{23}) = \begin{cases} [* \quad * \ \{(2,2), (1,2), (0,2)\}] \text{ if } x_{23}=0 \\ [* \quad * \ \{(2,1), (1,1), (0,1), \\ \ (2,0), (1,0), (0,0)\}] \text{ if } x_{23}=1. \end{cases}$$

Finally, as in the previous section, κ_3^{-1} and γ^{-1} for S_2 were found. These are given in Tables 9 and 10 respectively.

3.5.7.3 γ^{-1} for the Gracefully Degrading Model

The capability function preimage sets for S_3, the gracefully degrading processor model, are derived in the same manner

x_c	$\kappa_3^{-1}(x_c) \subseteq W$
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 0 \end{bmatrix}$	[2 {(2,2),(2,1)} {(2,2),(2,1)}]
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 1 \end{bmatrix}$	[2 {(2,2),(2,1)} {(2,1),(2,0),(1,1),(1,0)}]
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 1 & * \end{bmatrix}$	[2 (2,0) {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}]
$\begin{bmatrix} 1 & 0 & 0 \\ \not\in & * & * \end{bmatrix}$	[(0,1) {(2,2),(2,1),(2,0)} {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}]
$\begin{bmatrix} 0 & 1 & 0 \\ \not\in & * & * \end{bmatrix}$	[2 {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 0 \end{bmatrix}$	[2 {(2,2),(2,1)} (0,2)]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 1 \end{bmatrix}$	[2 {(2,2),(2,1)} {(0,1),(0,0)}]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 1 & * \end{bmatrix}$	[2 (2,0) {(0,2),(0,1),(0,0)}]
$\begin{bmatrix} 1 & 1 & * \\ \not\in & * & * \end{bmatrix}$	[(0,1) {((1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} *]
$\begin{bmatrix} 1 & 0 & 1 \\ \not\in & * & * \end{bmatrix}$	[(0,1) {(2,2),(2,1),(2,0)} {(0,2),(0,1),(0,0)}]
$\begin{bmatrix} 0 & 1 & 1 \\ \not\in & * & * \end{bmatrix}$	[2 {((1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} {(0,2),(0,1),(0,0)}]
$\begin{bmatrix} 0 & 0 & * \\ \not\in & 0 & 1 \end{bmatrix}$	[2 {(2,2),(2,1)} {(2,1),(1,1),(0,1),(2,0),(1,0),(0,0)}]

Table 9

Preimages of κ_3 of the Dedicated Group Processor Model

x_c	$\gamma^{-1}(a) \subseteq W \times Y_c$
a_0	([2 {(2,2),(2,1)} {(2,2),(1,2)}] x [* ¢]) ∪ ([2 {(2,2),(2,1)} {(2,1),(2,0),(1,1),(1,0)}] x [0 ¢]) ∪ ([2 (2,0) {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [0 ¢])
a_1	([{0,1} (2,0) {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [0 ¢]) ∪ ([2 {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [0 ¢]) ∪ ([2 {(2,2),(2,1)} 2] x [* ¢]) ∪ ([2 {(2,2),(2,1)} 0] x [0 ¢]) ∪ ([2 (2,0) {(0,2),(0,1),(0,0)}] x [0 ¢])
a_2	[2 (2,0) {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [1 ¢]
a_3	([{0,1} (2,0) {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [1 ¢]) ∪ ([2 {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} {(2,2),(2,1),(2,0),(1,2),(1,1),(1,0)}] x [1 ¢]) ∪ ([2 (2,0) {(0,2),(0,1),(0,0)}] x [1 ¢])
a_4	([{0,1} {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} *] x [* ¢]) ∪ ([{0,1} (2,0) {(0,2),(0,1),(0,0)}] x [* ¢]) ∪ ([2 {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)} {(0,2),(0,1),(0,0)}] x [* ¢]) ∪ ([2 {(2,2),(2,1)} {(1,2),(1,1),(1,0),(0,2),(0,1),(0,0)}] x [1 ¢])

Table 10

Preimages of γ for the Dedicated Group Processor Model

as the sets for the other two models (Sections 3.5.7.1-2). The $\xi_{ij}\kappa_3$ are below.

$$\xi_{11}\kappa_3(w) = \begin{cases} 0 \text{ if } w_1 \in \{4,3,2\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [\{4,3,2\} \ * \ \ *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{12}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{4,3,2\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \ \{4,3,2\} \ \ *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{13}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{4,3,1\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \ \ * \ \{4,3,1\}] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{21}\kappa_3(w) = \phi$$

$$\xi_{22}\kappa_3(w) = \begin{cases} 0 \text{ if } w_2 \in \{4,3,1\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \ \{4,3,1\} \ \ *] \\ 1 \text{ otherwise,} \end{cases}$$

$$\xi_{23}\kappa_3(w) = \begin{cases} 0 \text{ if } w_3 \in \{4,3,2\} \\ 1 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 0 \text{ if } w \in [* \ \ * \ \{4,3,2\}] \\ 1 \text{ otherwise} \end{cases} .$$

From κ_3 the component inverses $(\xi_{ij}\kappa_3)^{-1}$ can be derived:

$$(\xi_{11}\kappa_3)^{-1}(x_{11}) = \begin{cases} [\{4,3,2\} \quad * \quad *] \text{ if } x_{11}=0 \\ [\{0,1\} \quad * \quad *] \text{ if } x_{11}=1, \end{cases}$$

$$(\xi_{12}\kappa_3)^{-1}(x_{12}) = \begin{cases} [* \quad \{4,3,2\} \quad *] \text{ if } x_{12}=0 \\ [* \quad \{0,1\} \quad *] \text{ if } x_{12}=1, \end{cases}$$

$$(\xi_{13}\kappa_3)^{-1}(x_{13}) = \begin{cases} [* \quad * \quad \{4,3,1\}] \text{ if } x_{13}=0 \\ [* \quad * \quad \{0,2\}] \text{ if } x_{13}=1, \end{cases}$$

$$(\xi_{21}\kappa_3)^{-1}(x_{21}) = [* \quad * \quad *],$$

$$(\xi_{22}\kappa_3)^{-1}(x_{22}) = \begin{cases} [* \quad \{4,3,1\} \quad *] \text{ if } x_{22}=0 \\ [* \quad \{0,2\} \quad *] \text{ if } x_{22}=1, \end{cases}$$

$$(\xi_{23}\kappa_3)^{-1}(x_{23}) = \begin{cases} [* \quad * \quad \{4,3,2\}] \text{ if } x_{23}=0 \\ [* \quad * \quad \{0,1\}] \text{ if } x_{23}=1. \end{cases}$$

In Tables 11 and 12 are shown the relations κ_3^{-1} and γ^{-1} for S_3.

3.5.8 Performability Evaluation

Once the γ-induced trajectory sets of a computer system over a mission have been determined, the computer's performability is obtained by calculating the probability of each of the trajectory sets. That is,

$$p_s(a) = Pr(\gamma^{-1}(a)).$$

This section describes how these probabilities were determined for the example mission under discussion. In particular, Section 3.5.8.1 examines METAPHOR, a software package we are currently developing to aid in performability evaluation. METAPHOR was used to determine the performability of each computer S_1, S_2, and S_3 under several sets of conditions. These conditions involve the failure rate of the computer modules, the length of

x_c	$\kappa_3^{-1}(x_c) \subseteq W$
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 0 \end{bmatrix}$	[{4,3,2} {4,3} {4,3}]
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 0 & 1 \end{bmatrix}$	[{4,3,2} {4,3} 1]
$\begin{bmatrix} 0 & 0 & 0 \\ \not\in & 1 & * \end{bmatrix}$	[{4,3,2} 2 {4,3,1}]
$\begin{bmatrix} 1 & 0 & 0 \\ \not\in & * & * \end{bmatrix}$	[{0,1} {4,3,2} {4,3,1}]
$\begin{bmatrix} 0 & 1 & 0 \\ \not\in & * & * \end{bmatrix}$	[{4,3,2} {0,1} {4,3,1}]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 0 \end{bmatrix}$	[{4,3,2} {4,3} 2]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 0 & 1 \end{bmatrix}$	[{4,3,2} {4,3} 0]
$\begin{bmatrix} 0 & 0 & 1 \\ \not\in & 1 & * \end{bmatrix}$	[{4,3,2} 2 {0,2}]
$\begin{bmatrix} 1 & 1 & * \\ \not\in & * & * \end{bmatrix}$	[{0,1} {0,1} *]
$\begin{bmatrix} 1 & 0 & 1 \\ \not\in & * & * \end{bmatrix}$	[{0,1} {4,3,2} {0,2}]
$\begin{bmatrix} 0 & 1 & 1 \\ \not\in & * & * \end{bmatrix}$	[{4,3,2} {0,1} {0,2}]
$\begin{bmatrix} 0 & 0 & * \\ \not\in & 0 & 1 \end{bmatrix}$	[{4,3,2} {4,3} {0,1}]

Table 11

Preimages of κ_3 of the Gracefully Degrading Processor Model

x_c	$\gamma^{-1}(a) \subseteq W \times Y_c$
a_0	([{4,3,2} {4,3} {4,3}] x [* ¢]) ∪([{4,3,2} {4,3} 1] x [0 ¢]) ∪([{4,3,2} 2 {4,3,1}] x [0 ¢])
a_1	([{1,0} {4,3,2} {4,3,1}] x [0 ¢]) ∪([{4,3,2} {1,0} {4,3,1}] x [0 ¢]) ∪([{4,3,2} {4,3} 2] x [* ¢]) ∪([{4,3,2} {4,3} 0] x [0 ¢]) ∪([{4,3,2} 2 {2,0}] x [0 ¢])
a_2	[{4,3,2} 2 {4,3,1}] x [1 ¢]
a_3	([{1,0} {4,3,2} {4,3,1}] x [1 ¢]) ∪([{4,3,2} {1,0} {4,3,1}] x [1 ¢]) ∪([{4,3,2} 2 {2,0}] x [1 ¢])
a_4	([{1,0} {1,0} *] x [* ¢]) ∪([{1,0} {4,3,2} {2,0}] x [* ¢]) ∪([{4,3,2} {1,0} {2,0}] x [* ¢]) ∪([{4,3,2} {4,3} {1,0}] x [1 ¢])

Table 12

Preimages of γ for the Gracefully Degrading Processor Model

mission, and the probability of Category III weather at the destination. Section 3.5.8.2 gives the results of these calculations.

3.5.8.1. METAPHOR

To aid in the evaluation and analysis of performability, we are developing a software package called METAPHOR (Michigan Evaluation Aid for Perphormability). We envision METAPHOR ultimately as a tool to be used at all stages of performability analysis, from the definition of model levels and interlevel translations to the determination of γ-induced trajectory sets to the evaluation of the probability of those sets. At present, only the last function has been implemented.

Because of the design nature of constructing the model hierarchy, we believe METAPHOR must be an interactive facility. Hence we are incorporating into METAPHOR a command language which will enable the user to call desired functions, enter data, display results, and seek help or explanations of any function. Among the commands already implemented for computing the probability of a trajectory set are the following:

DATA	asks the user which input data he would like to see and then displays it
ALTER	asks the user which input data he would like to alter and then performs the alteration
HELP	when typed in response to a question, METAPHOR replies with an explanation of the question
CALC	allows the user to utilize the APL calculator mode
END	exits METAPHOR.

At present, METAPHOR defaults to calculating the probability of trajectory sets. The user is queried for the number of phases in the bottom model and the number of states in each phase. Then the program asks for the transition matrix P of each phase as well as the interphase transition matrix H between each phase. (See Section 3.4.3 .) METAPHOR is capable of generating several classes of P matrices corresponding to various classes of Markov models. The user need supply only the model type, the failure rate of each module involved, and the length of each phase. Alternatively, the user can enter the P matrices directly. Currently, the H matrices must be entered directly.

Next, METAPHOR requests the number of basic variables as well as their probabilities. At present, METAPHOR can handle only a string of base variables each of which consists of a single obserbation. The user is then asked the number of accomplishment levels, and for each accomplishment level, the user must input the number of array products used to describe the corresponding trajectory set $\gamma^{-1}(a)$. The trajectory sets must be disjoint; likewise, the array products must also be disjoint.

Finally, for each trajectory product array, the user must supply the initial state vector I(0), the characteristic matrix G for each phase, the characteristic vector F, and the basic variable values. METAPHOR then calculates the probability of achieving each trajectory product array V using the relation

$$Pr(V) = I(0)\left[\prod_{i=1}^{k-1} P(i)G(i)H(i)\right] P(k)F(k)$$

where k is the number of phases and the product operation is matrix multiplication. (See Section 3.4.3 .) The effect of the non-bottom level basic variables and their associated probabilities are also taken into account by METAPHOR.

Throughout METAPHOR, extensive error checking is provided on all inputs, to insure both proper data types (e.g., numeric vs. character, or scaler vs. vector) as well as logical consistency (e.g., probabilities summing to one). If an error does occur, the user is prompted and the question is asked again.

For this preliminary study of METAPHOR, the language APL[16] was chosen for the prototype program because of its compactness and array handling abilities. Once the feasability of the program has been demonstrated, however, translating the package into a faster and more portable language such as FORTRAN may be desired. At present, METAPHOR contains approximately fifty APL functions and about 700 lines of code. Also, internal documentation is generous. External documentation, on the other hand, is not as thoroughly developed. Because of this and because METAPHOR is still in an early developmental state, we do not include a listing of the package in this report. Figure 13 shows the output for a run evaluating the performability of the gracefully degrading computer S_3. The next section discusses the input in more detail. During the next reporting period, we intend to continue our efforts in developing METAPHOR.

3.5.8.2 Performability Results

Using both the performability models constructed earlier in

```
METAPHOR
VERSION 1
11/77


NUMBER OF PHASES?
⎕:  3

NUMBER OF STATES PER PHASE?  (SPACE BETWEEN EACH NUMBER)
⎕:  5  5  5


SPECIFY THE P MATRICES FOR EACH PHASE, 1 PHASE AT A TIME

PHASE 1:

WHAT TYPE OF P MATRIX?
⎕:  2
ENTER PHASE LENGTH
⎕:  2.5
ENTER COMPONENT FAILURE RATE
⎕:  0.0001
ENTER NUMBER OF GROUPS
⎕:  1
ENTER NUMBER OF COMPONENTS PER GROUP  (SPACE BETWEEN EACH NUMBER):
⎕:  4

PHASE 2:

WHAT TYPE OF P MATRIX?
⎕:  2
ENTER PHASE LENGTH
⎕:  2.5
ENTER COMPONENT FAILURE RATE
⎕:  0.0001
ENTER NUMBER OF GROUPS
⎕:  1
ENTER NUMBER OF COMPONENTS PER GROUP  (SPACE BETWEEN EACH NUMBER):
⎕:  4

PHASE 3:

WHAT TYPE OF P MATRIX?
⎕:  2
ENTER PHASE LENGTH
⎕:  2.5
ENTER COMPONENT FAILURE RATE
⎕:  0.0001
```

FIGURE 13

SAMPLE SESSION WITH METAPHOR

```
ENTER NUMBER OF GROUPS
⎕:  1
ENTER NUMBER OF COMPONENTS PER GROUP  (SPACE BETWEEN EACH NUMBER):
⎕:  4
ENTER THE H MATRICES FOR EACH PHASE, 1 PHASE AT A TIME


PHASE 1:

ROW 1:
⎕:  1  0  0  0  0
ROW 2:
⎕:  0  1  0  0  0
ROW 3:
⎕:  0  0  1  0  0
ROW 4:
⎕:  0  0  0  1  0
ROW 5:
⎕:  0  0  0  0  1


PHASE 2:

ROW 1:
⎕:  1  0  0  0  0
ROW 2:
⎕:  0  1  0  0  0
ROW 3:
⎕:  0  0  1  0  0
ROW 4:
⎕:  0  0  0  1  0
ROW 5:
⎕:  0  0  0  0  1

NUMBER OF CONSTANT BASIC VARIABLES?
⎕:  1

PROBABILITIES OF EACH CONSTANT VARIABLE?  (SPACE BETWEEN EACH NUMBER)
⎕:  0.0019

NUMBER OF ACCOMPLISHMENT LEVELS?
⎕:  5

ACCOMPLISHMENT LEVEL 0
NUMBER OF TRAJECTORY SETS FOR THIS ACCOMPLISHMENT LEVEL?
⎕:  3
TRAJECTORY SET 1
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

```
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  2
TRAJECTORY SET 2
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  0
TRAJECTORY SET 3
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  0

ACCOMPLISHMENT LEVEL 1
NUMBER OF TRAJECTORY SETS FOR THIS ACCOMPLISHMENT LEVEL?
⎕:  5
TRAJECTORY SET 1
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

```
⎕:  0
TRAJECTORY SET 2
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  0
TRAJECTORY SET 3
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  2
TRAJECTORY SET 4
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  0  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  0
TRAJECTORY SET 5
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  1
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

```
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  0

ACCOMPLISHMENT LEVEL 2
NUMBER OF TRAJECTORY SETS FOR THIS ACCOMPLISHMENT LEVEL?
⎕:  1
TRAJECTORY SET 1
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  1

ACCOMPLISHMENT LEVEL 3
NUMBER OF TRAJECTORY SETS FOR THIS ACCOMPLISHMENT LEVEL?
⎕:  3
TRAJECTORY SET 1
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  1
TRAJECTORY SET 2
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  1  0
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  1
TRAJECTORY SET 3
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

```
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  1

ACCOMPLISHMENT LEVEL 4
NUMBER OF TRAJECTORY SETS FOR THIS ACCOMPLISHMENT LEVEL?
⎕:  4
TRAJECTORY SET 1
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  1  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  2
TRAJECTORY SET 2
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  1  0  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  2
TRAJECTORY SET 3
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

```
⎕:  0  0  1  0  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  2
TRAJECTORY SET 4
ENTER THE I VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  1  0  0  0  0
PHASE 1:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  1  0  0
PHASE 2:
ENTER THE G DIAGONAL (SPACE BETWEEN EACH ENTRY):
⎕:  1  1  0  0  0
ENTER THE F VECTOR (SPACE BETWEEN EACH ENTRY):
⎕:  0  0  0  1  1
ENTER THE BASIC VARIABLE VECTOR  (SPACE BETWEEN EACH ENTRY):
⎕:  1


PERFORMABILITY FOR THIS MISSION = 0.9999966309  1.873257051E¯6
    7.471727544E¯10  1.494594808E¯6  4.983160269E¯10
END
```

FIGURE 13 (CONT)

SAMPLE SESSION WITH METAPHOR

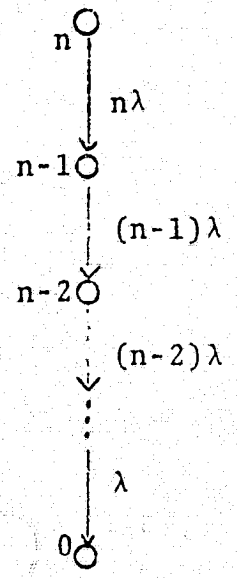
this section as well as METAPHOR, the performabilities of S_1, S_2, and S_3 in several environments were determined. The environments and performability results are described below.

Two user environments were considered. The first was a moderately long flight of 5.5 hours from Washington, D.C. to Los Angeles, California, while the second was a short flight of 1.0 hour from Washington, D.C. to the J.F. Kennedy Airport in New York. The probability of Category III weather in Los Angeles was assumed to be 0.019, and for New York, it was assumed to be 0.011. (These probabilities are from Table 16 of [17].) In addition, it was supposed that three types of processor modules were available with failure rates of 0.001, 0.0001, and 0.00001 failures/hour respectively, but identical in all other regards. (Of course a different cost will be associated with each type of module, the least reliable one being the cheapest.)

For the Los Angeles flight, phase 1 is 2.5 hours, phase 2 is 2.5 hours, and phase 3 is 0.5 hours. The New York flight has corresponding phase lengths of 1/3, 1/3, and 1/3 hours. The landing phase of the New York flight is shorter because we assume the New York plane flies at a lower altitude than the Los Angeles plane due to the shorter trip.

Figure 13 shows a sample session with METAPHOR used to determine the performability in the Los Angeles flight environment of S_3 having modules with a failure rate of 0.001 failures/hour. In the session, the analyst first tells METAPHOR that there are 3 phases in the mission with 5 states in each phase. In phase 1, the P matrix is of type 2. This is one of the types of P matrices

which METAPHOR will automatically generate if the proper parameters are input. Type 2 signifies a system with m groups of n components each. Every group has a transition diagram of the form



where the state name is the number of surviving components in the group. For S_3, there is 1 group with 4 processors. The failure rate of the processors is input as 0.0001 failures/hour while the phase length (for phase 1) is given to be 2.5 hours. Similar information is presented to METAPHOR for phases 2 and 3.

Next, the analyst informs METAPHOR that the H matrices are the 5 x 5 identity matrices, that the single non-bottom level basic variable is constant and has probability 0.019, and that 5 accomplishment levels are to be evaluated. For accomplishment level 0, METAPHOR is told that there are 3 trajectory sets, and the analyst inputs each set by first entering the I vector, then the main diagonal of the characteristic (G) matrix for phases 1 and 2, the characteristic (F) vector, and the condition of the

weather variable. 0 means non-Category III weather, 1 denotes Category III weather, and 2 signifies either 0 or 1 (i.e., a "don't care"). METAPHOR calculates the performability "on the fly", i.e., METAPHOR does the necessary calculations on the data as it is input and then discards the data it will no longer require. Once the analyst has completed entering all the trajectory sets for each accomplishment level, METAPHOR prints the performability.

METAPHOR was used to evaluate the three models S_1, S_2, and S_3 for each of the failure rates in both user environments. The results of these computations are in Table 13. Employing bar graphs, Figure 14 compares the performability of each processor and environment vis-a-vis the processor module failure rate. Because of the wide range in magnitude of the probabilities, a logarithmic axis was used. For accomplishment levels a_1 through a_4, the axis is labeled in increments of 10^{-1}, while for a_0, the axis is labeled in terms of "n 9's." This phrase denotes $1 - 10^{-n}$; for example, 5 9's = 0.9999 3 9's = 0.999.

As is to be expected, the gracefully degrading processor model, S_3, has a higher probability of accomplishing a_0 and lower probabilities of achieving a_1 through a_4 than the other two processor models. Consider however the interesting results regarding S_1 and S_2. In particular, note that the probability of a crash, a_4, is greater for the dedicated group processor model S_2, while the values for a_0 are the same. This outcome is somewhat surprising since S_2 has some form of reconfiguration and so seemingly should be more reliable. However, examination of the entire performability spectrum reveals the reasons for this discrepancy. Note that for

Mission Enviornment	Computer Model	Module Failure Rate (failures per hour)	Accomplishment Levels a_0	a_1	a_2	a_3	a_4
Washington, D. C. to New York (JFK) $B = 0.011$	S_1	0.001	0.997	3.4×10^{-4}	6.6×10^{-4}	3.3×10^{-4}	1.7×10^{-3}
		0.0001	0.9997	3.4×10^{-5}	6.6×10^{-5}	3.3×10^{-5}	1.7×10^{-4}
		0.00001	0.99997	3.4×10^{-6}	6.6×10^{-6}	3.3×10^{-6}	1.7×10^{-5}
	S_2	0.001	0.997	7.4×10^{-6}	4.4×10^{-7}	6.6×10^{-4}	2.6×10^{-3}
		0.0001	0.9997	7.3×10^{-7}	4.4×10^{-9}	6.6×10^{-5}	2.6×10^{-4}
		0.00001	0.99997	7.3×10^{-8}	4.4×10^{-11}	6.6×10^{-6}	2.5×10^{-5}
	S_3	0.001	0.999994	3.4×10^{-6}	1.8×10^{-9}	2.6×10^{-6}	1.2×10^{-9}
		0.0001	0.99999994	3.4×10^{-8}	1.8×10^{-12}	2.6×10^{-8}	1.2×10^{-12}
		0.00001	0.9999999994	3.4×10^{-10}	1.8×10^{-15}	2.6×10^{-10}	1.2×10^{-15}
Washington, D. C. to Los Angeles $B = 0.019$	S_1	0.001	0.98	5.4×10^{-4}	4.8×10^{-3}	2.4×10^{-3}	8.4×10^{-3}
		0.0001	0.998	5.5×10^{-5}	4.9×10^{-4}	2.5×10^{-4}	8.5×10^{-4}
		0.00001	0.9998	5.5×10^{-6}	4.9×10^{-5}	2.5×10^{-5}	8.5×10^{-5}
	S_2	0.001	0.98	9.4×10^{-5}	1.2×10^{-5}	4.9×10^{-3}	1.3×10^{-2}
		0.0001	0.998	9.5×10^{-6}	1.2×10^{-7}	4.9×10^{-4}	1.2×10^{-3}
		0.00001	0.9998	9.5×10^{-7}	1.2×10^{-9}	4.9×10^{-5}	1.3×10^{-4}
	S_3	0.001	0.9998	3.4×10^{-5}	1.4×10^{-7}	1.5×10^{-4}	7.8×10^{-8}
		0.0001	0.999998	3.4×10^{-7}	1.5×10^{-10}	1.5×10^{-6}	7.8×10^{-11}
		0.00001	0.99999998	3.4×10^{-9}	1.5×10^{-13}	1.5×10^{-8}	7.8×10^{-14}

Table 13

Performability for S_1, S_2, and S_3. Modules have three failure rates, and two mission enviornments are considered.

B = Pr(Category III weather at initiation of landing).

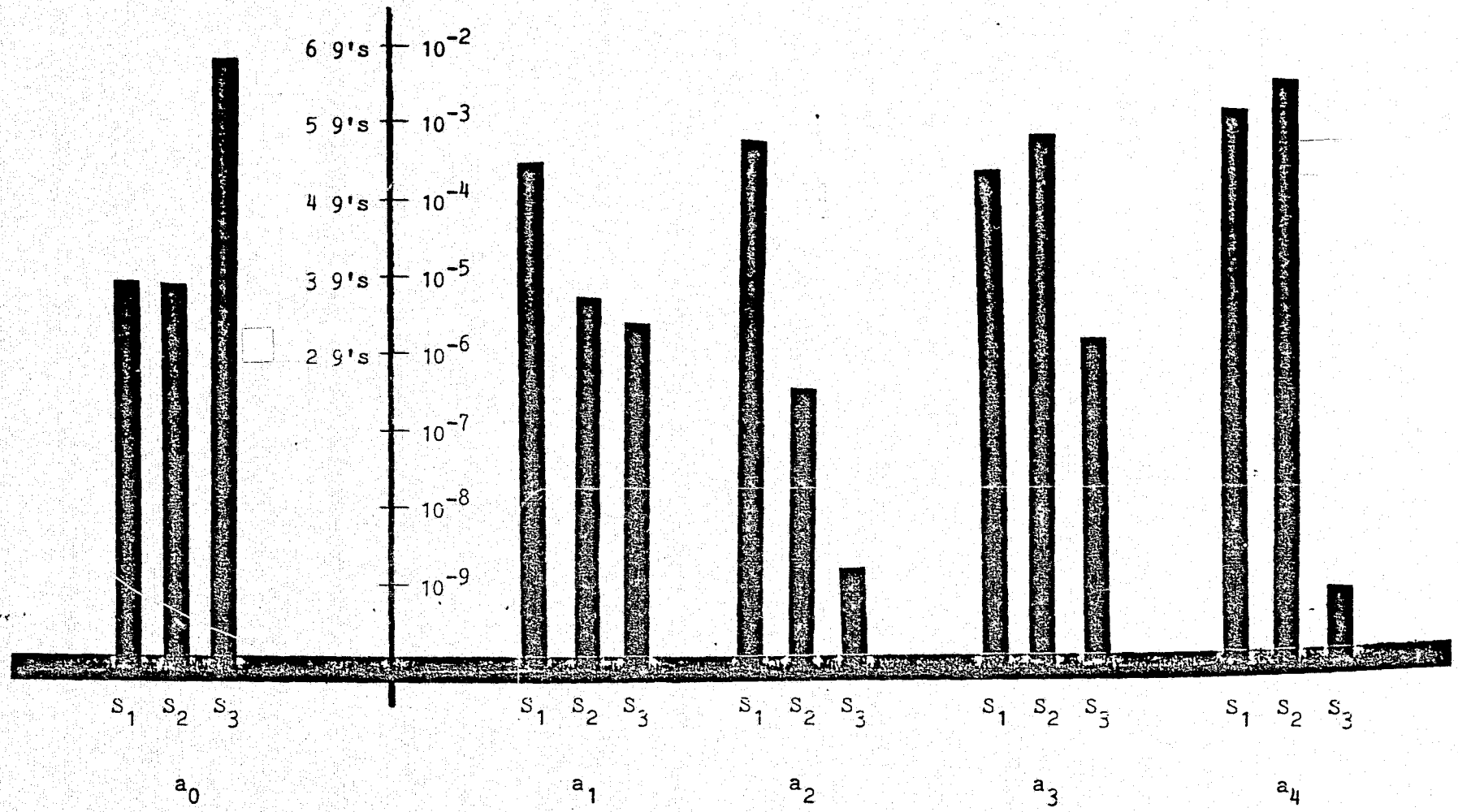


Figure 14

Performability for the Example of Section 3.5.

a) Processor Module Failure Rate = 0.001 failures/hour
Washington, D. C. to New York Mission

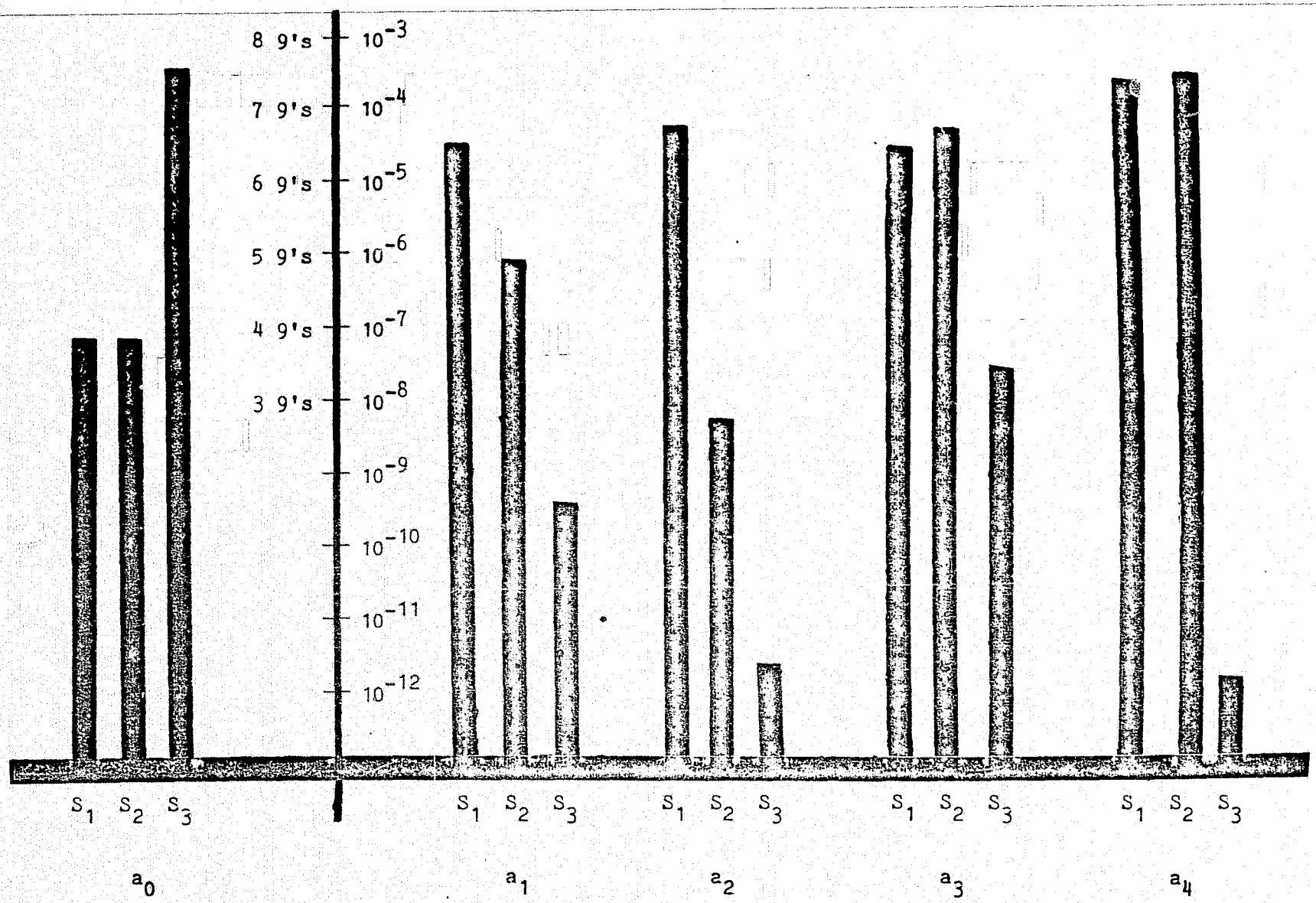


Figure 14 cont.)

Performability for the Example of Section 3.5.

b) Processor Module Failure Rate = 0.0001 failures/hour
Washington, D. C. to New York Mission

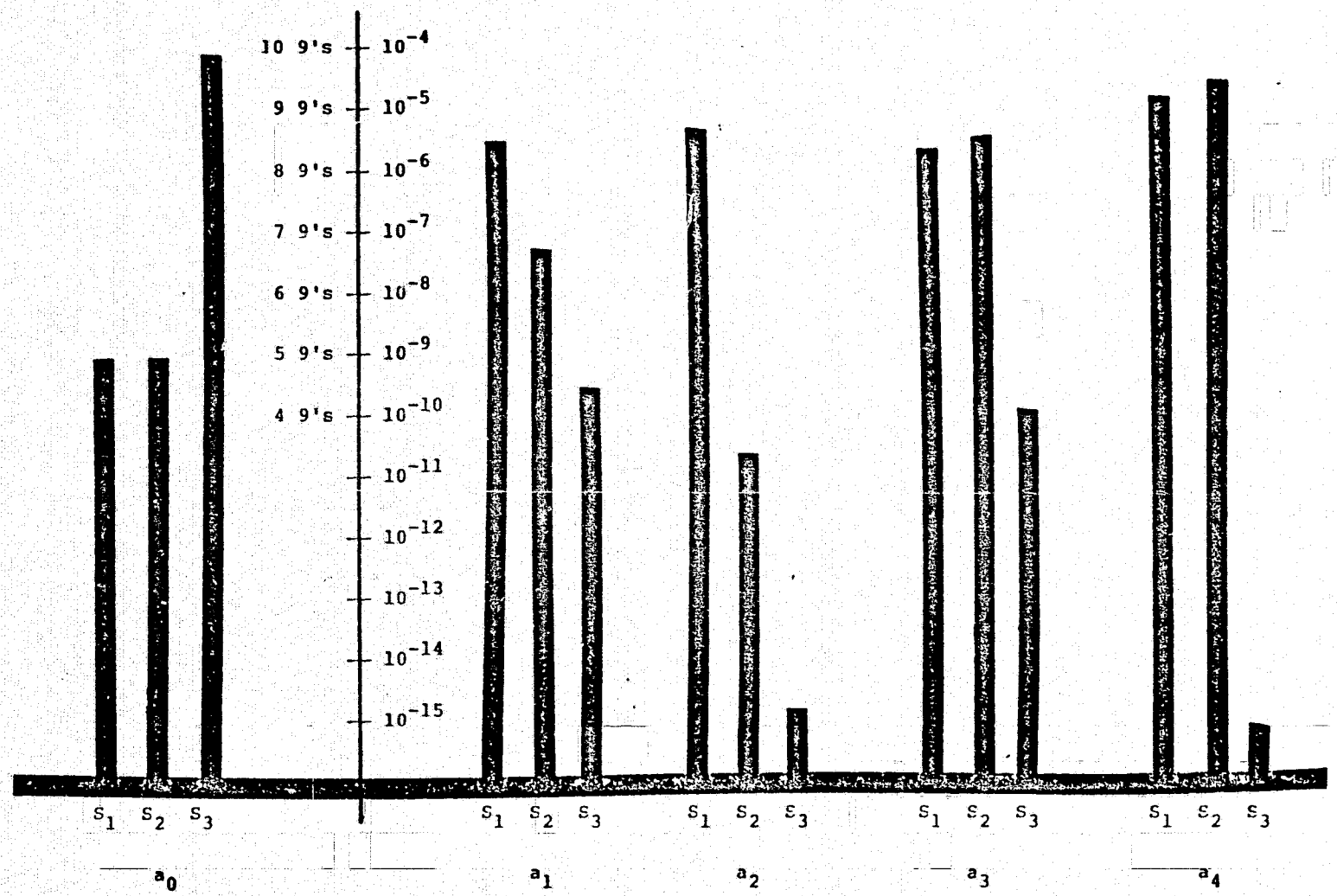


Figure 14 cont.)

Performability for the Example of Section 3.5.

c) Processor Module Failure Rate = 0.00001 failures/hour
Washington, D. C. to New York Mission

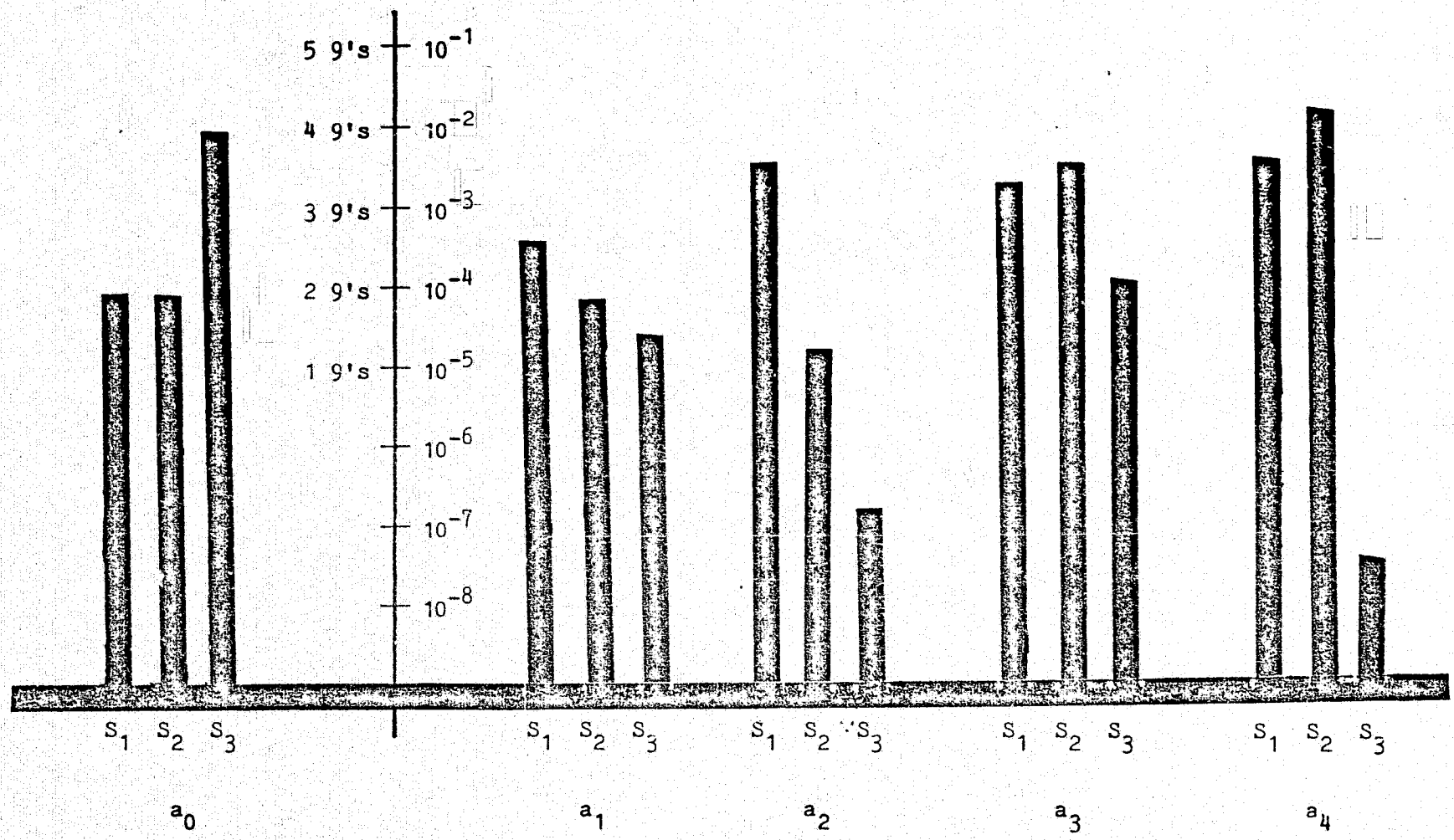


Figure 14 cont.)

Performability for the Example of Section 3.5.

d) Processor Module Failure Rate = 0.001 failures/hour
Washington, D. C. to Los Angles Mission

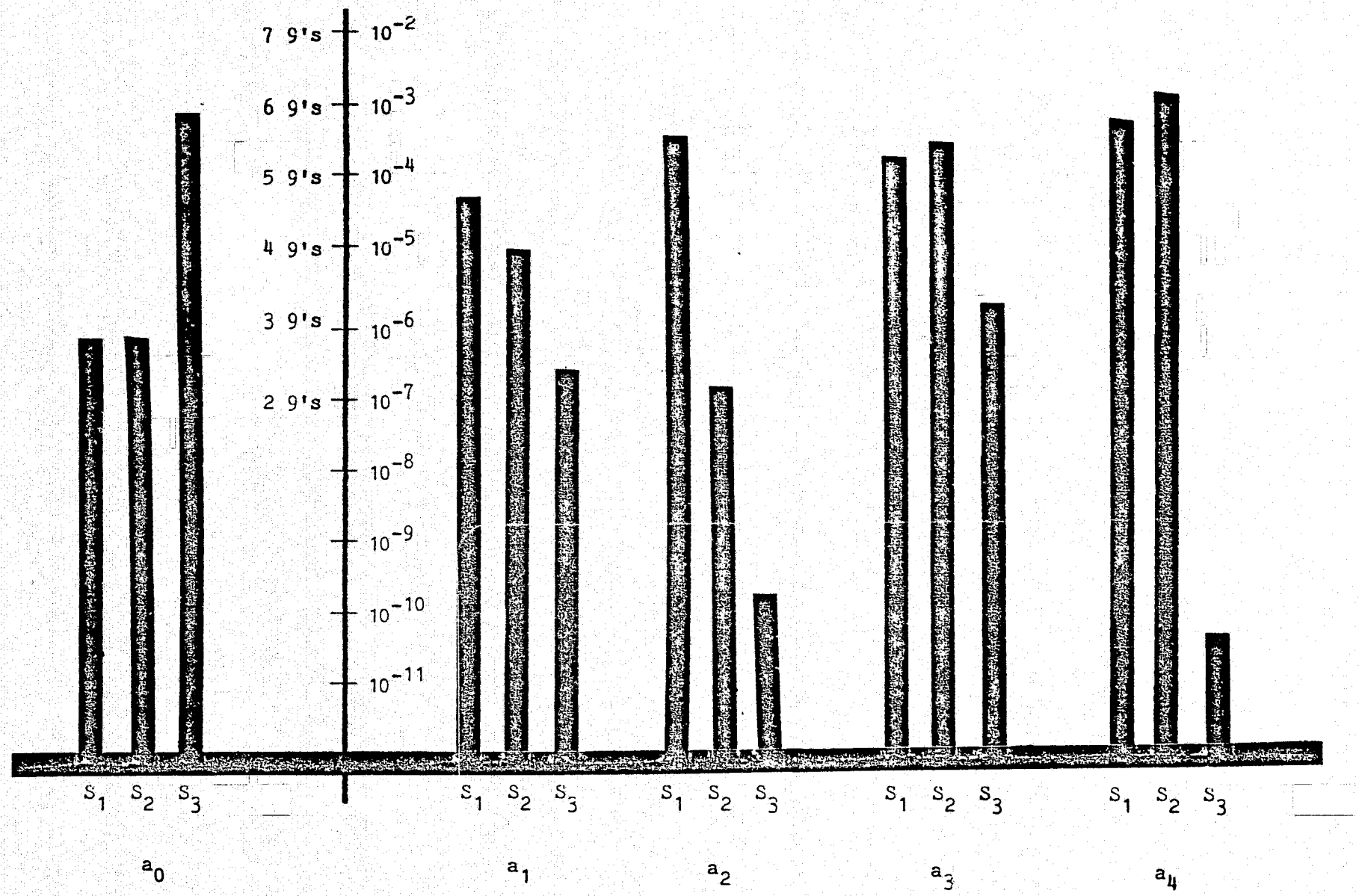


Figure 14 cont.)

Performability for the Example of Section 3.5.

e) Processor Module Failure Rate = 0.0001 failures/hour
Washington, D. C. to Los Angles Mission

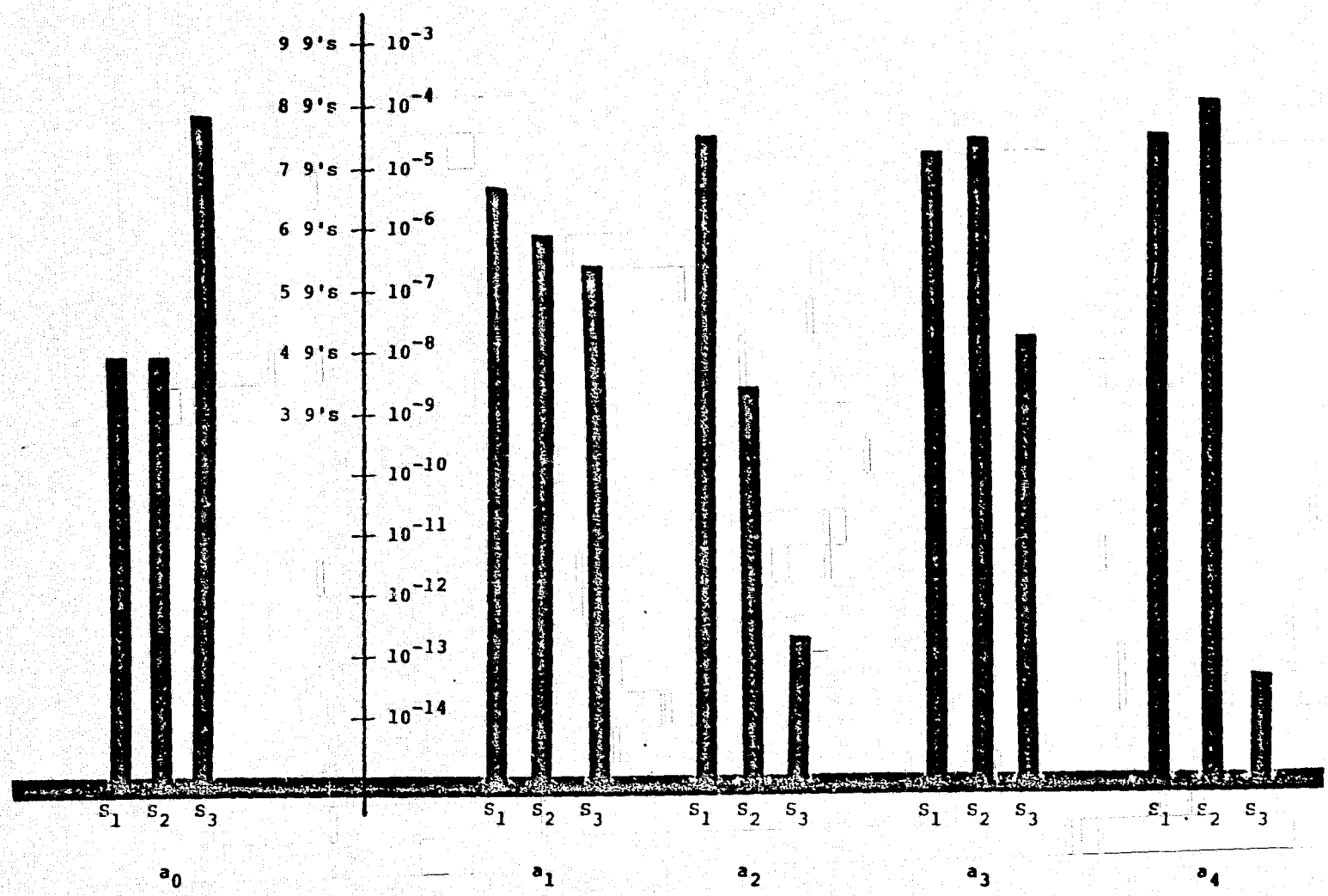


Figure 14 cont.)

Performability for the Example of Section 3.5.

f) Processor Module Failure Rate = 0.00001 failures/hour
Washington, D. C. to Los Angles Mission

S_2, the probability of the high fuel consumption achievement, a_1, is about an order of magnitude lower than for S_1; similarly, the probability of the diversion achievement is usually several orders of magnitude lower. Thus, the reconfiguration and diversion policies associated with S_2 have decreased the chances of either diverting or having poor fuel consumption, but have done so at the expense of increasing the probability of crashing. If a suitable worth function W_S(see Section 3.1.1) were provided, the worth of each system in each environment could be calculated.

Within the example given in this section we can plainly see the advantages of performability analysis over traditional reliability analysis. Reliability results indicate the probability of "success" or "failure" with respect to some set of success criteria, but do not as succinctly reflect the performance of the system. For example, a traditional reliability analysis of the mission in this section might have determined the probability that 2 of the 4 processors were still working at the end of the mission. Using more sophisticated methods such as the phasing techniques of Esary and Ziehms [8] would have improved the reliability analysis. However, the performability analysis demonstrated in this section can accomodate even more general relations between state behavior and performance than those treated by traditional phasing techniques. In particular, the analysis is able to treat levels of system performance which cannot be formulated in terms of per-phase structure functions or per-phase sets of "success states."

For instance, consider S_3 and suppose phase 3 has state 4 and the weather is not Category III. Then from Table 12, the

relation between the states of phases 1 and 2 and the associated accomplishment levels can be determined:

q_1	q_2	accomplishment level
{0,1}	{4,3,2}	a_1
{4,3,2}	{0,1}	a_1
{4,3,2}	{4,3,2}	a_0
{0,1}	{0,1}	a_4

Thus, if $q_2 \in \{4,3,2\}$ then the "success states" for a_1 cannot include $q_1 \in \{0,1\}$. Similarly, if $q_1 \in \{4,3,2\}$ then for a_1, $q_2 \in \{0,1\}$. In other words, the "success states" for phases 1 and 2 are influenced by one another. In fact, they are R-dependent where R is $\gamma^{-1}(a_1)$. (See Section 3.3.2.)

During the next reporting period, we plan to study further examples of performability analysis.

4. REFERENCES

[1] "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Semi-Annual Status Report No. 1, NASA Grant NSG 1306, November, 1976.

[2] "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Semi-Annual Status Report No. 2, NASA Grant NSG 1306, July 1977.

[3] "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Proposal for extension of NASA Grant NSG 1306, submitted to NASA Langley Research Center and subsequently approved, May 1977.

[4] M. Loève, Probability Theory, Third Edition, Princeton, NJ: Van Nostrand, 1963.

[5] W. G. Bouricius, W. C. Carter and P. R. Schneider, "Reliability Modeling Techniques for Self-Repairing Computer Systems," Proc. 24th ACM National Conference, pp. 295-305, August, 1969.

[6] R. E. Barlow and F. Proschan, Statistical Theory of Reliability and Life Testing Probability Methods, New York: Holt, Rhinehart and Winston, Inc., 1975.

[7] J. F. Meyer, "Computation-Based Reliability Analysis," IEEE Trans. Computers, Vol. C-25, pp. 578-584, June 1976.

[8] J. D. Esary and H. Ziehms, "Reliability of Phased Missions," Reliability and Fault Free Analysis, SIAM, Philadelphia, PA, pp. 213-236, 1975.

[9] B. P. Zeigler, "Towards a Formal Theory of Modeling and Simulation: Structure Preserving Morphisms," JACM, 19 (4), October, 1972, pp. 742-764.

[10] C. J. Masreliez and B. E. Bjurman, "Fault-Tolerant System Reliability Modeling/Analysis," presented at the AIAA Guidance and Control Conference, San Diego, CA, August, 1976.

[11] A. W. Naylor, "On a Concept of Generalized Dependence for System Decomposition," Proc. of the 1977 Joint Automatic Control Conference, Vol. 2, San Francisco, CA, 1977, pp. 521-526.

[12] T. B. Smith, A. L. Hopkins, et al., "A Fault-Tolerant Multiprocessor Architecture for Aircraft," Vol. 1, Technical Report, NASA Contract NAS1-13782, The Charles Stark Draper Lab., Inc., Cambridge, MA, July, 1976.

[13] J.-C. Laprie, "Reliability and Availability of Repairable Structures," Digest 1975 International Symposium on Fault-Tolerant Computing, Paris, France, June, 1975, pp. 87-92.

[14] J. G. Kemeny and J. L. Snell, Finite Markov Chains, Princeton, NJ: Van Nostrand, 1960.

[15] R. E. Miller, Switching Theory. Volume I: Combinational Circuits, New York: John Wiley and Sons, Inc., 1965.

[16] S. Pakin, APL/360 Reference Manual, Second Edition, Chicago, IL: Science Research Associates, 1972.

[17] B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan, and J. E. Templeman, Airborne Advanced Reconfigurable Computer System (ARCS), Final Report, NASA Contract NAS1-13654, Boeing Commercial Aircraft Company, Seattle, Washington, August, 1976.