Reliability Analysis: System Models

Jeffrey A. Barnett
(jbarnett@nrtc.northrop.com)
The Northrop Corporation
Pico Rivera, CA 90660

Abstract

Reliability analysis is a design-time activity. The goal is to anticipate problems and suggest product improvements before manufacturing and delivery. Enhanced quality and customer satisfaction along with reduced life-cycle costs are the motivation. Analysts employ models of physical systems to determine points of vulnerability and failure probabilities. Unfortunately, the models of complex systems—those with dependency cycles among components—necessarily lead to ambiguous analyses. Below, the source of these ambiguities is identified and some AI and expert systems techniques are suggested to help deal with them.

Introduction

Reliability analysis is an activity that determines, from a product design, how resistant the product is to failure. The standard input to the analysis is a structure called a fault tree. The fault tree is equivalent to a boolean expression that represents the system failure condition in terms of component failures.

The outputs of a reliability analysis are the probability of system failure, calculated from the fault tree and probabilities of component failures, and an enumeration of critical sets. A critical set is a set of components where (1) the system necessarily fails if all elements in the set fail and (2) no proper subset satisfies the first condition. Knowledge of small critical sets is important because they represent acute vulnerabilities to outside influences, e.g., battle damage, not accounted for by the component failure probabilities.

Two serious problems face this popular and ubiquitous approach to reliability analysis. The first is deriving the proper fault tree from the product design. The second is computational complexity—the number of critical sets can grow exponentially with the size of the fault tree. This paper deals with the first problem and suggests ways that AI technology may be useful for its resolution. It is a companion to Barnett and Verma [1994] where modeling and computational methods for reliability analyses are described.

The interested reader should consult the classics, Amstadter [1971] and Barlow, Fussell, and Singpurwalla [1975], for more information on the foundations of reliability analysis. Current approaches are described in the *IEEE Transactions on Reliability Analysis* and in the proceedings and tutorial notes of the yearly *Annual Reliability and Maintainability Symposium*.

Summary

The following sections, *Simple Dependencies* and *Dependency Cycles*, introduce the Functional Dependency Graph (FDG), a structure to express intra-system dependencies. They show, through examples, that analyses of systems with dependency cycles are not exact. The next sections, *Fixed Points & Models*, *Representational Adequacy*, and *Limit Cycles*, describe, in more detail, the nature of the system models used for reliability analyses and identify the sources of ambiguity. The problems encountered in this application are related to the problems encountered when control systems have multiple steady-state possibilities. It is also shown that the size of the representational structures involved preclude straightforward manual trial-and-error problem-solving strategies.

The final sections, *A Better Approach* and *Open Problems*, suggest the use of AI and expert systems techniques to construct analytic aids to improve the modeling task and help select the analysis that best reflects the actual system and the designer's intent.

Simple Dependencies

In simple system designs, dependencies can be represented by a partial order. For example, a flash light is operational if the bulb works and the bulb functions only if the battery and switch are operational. It is trivial to form a correct fault tree for these systems. However, most real-world designs display dependency

Proc. AAAI Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems, Palo Alto (1994)

cycles among system functionalities, e.g., power from a motor depends on oil pressure, but oil pressure depends on the oil pump which needs engine power to work. Deriving a fault tree when dependency cycles exist is a tricky business and a source of many subtle errors in reliability analysis.

Figure 1 is an example of an electronic system that has no dependency cycles. It is represented by a Functional Dependency Graph (FDG), a structure that expresses operability conditions and system dependencies, as described by Barnett and Verma (1994). The edges represent functionalities and the nodes represent dependencies. Both are labeled. In this example, the b_i are power, the ϕ_i are non-redundant computational capabilities, and f is the redundant computational capability that is this system's design intent.

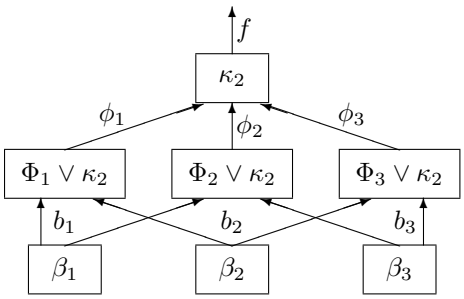

Figure 1: A redundant circuit without cycles.

Each node is annotated with a boolean expression that describes how the functionality labeling edges leaving the node, depends on components in the system and the functionalities labeling the edges entering the node. The β_i are batteries, the Φ_i are computational devices, and the κ_n are shorthand for expressions that evaluate *true* if n inputs are *true*: $\kappa_1 = x_1 \vee x_2$ and $\kappa_2 = x_1 x_2$ for a node with inputs x_1 and x_2, while

$$\begin{aligned} \kappa_1 &= x_1 \vee x_2 \vee x_3 \\ \kappa_2 &= x_1x_2 \vee x_1x_3 \vee x_2x_3 \\ \kappa_3 &= x_1x_2x_3 \end{aligned}$$

for a node with inputs x_1, x_2, and x_3.

Thus, a boolean equation, where *true* denotes failure, can be written for each functionality directly from the FDG. The failure conditions for the example system are expressed by

$$\begin{aligned} f &= \kappa_2(\phi_1, \phi_2, \phi_3) \\ \phi_1 &= \Phi_1 \vee \kappa_2(b_1, b_2) \\ \phi_2 &= \Phi_2 \vee \kappa_2(b_1, b_3) \\ \phi_3 &= \Phi_3 \vee \kappa_2(b_2, b_3) \\ b_1 &= \beta_1 \\ b_2 &= \beta_2 \\ b_3 &= \beta_3 \end{aligned}$$

and these equations are solved by simple substitution. Hence, the failure condition for f, expressed in terms of component failures, is

$$\begin{aligned} f = {} & (\Phi_1 \vee \beta_1\beta_2)(\Phi_2 \vee \beta_1\beta_3) \\ & \vee (\Phi_1 \vee \beta_1\beta_2)(\Phi_3 \vee \beta_2\beta_3) \\ & \vee (\Phi_2 \vee \beta_1\beta_3)(\Phi_3 \vee \beta_2\beta_3) \end{aligned}$$

and the probability of systemic failure, $p(f)$, can be expressed in terms of $p(\Phi_i)$, $p(\beta_i)$, and some conditional probabilities. The and/or expression for f is equivalent to its fault tree.

The critical sets (called minimum cutsets by some) are the terms that result when the failure condition is expressed in conjunctive normal form (CNF) and reduced by subsumption. In this case,

$$\begin{aligned} f = {} & \Phi_1\Phi_2 \vee \Phi_1\Phi_3 \vee \Phi_2\Phi_3 \vee \beta_1\beta_3\Phi_1 \\ & \vee \beta_2\beta_3\Phi_1 \vee \beta_1\beta_2\Phi_2 \vee \beta_2\beta_3\Phi_2 \\ & \vee \beta_1\beta_2\Phi_3 \vee \beta_1\beta_3\Phi_3 \vee \beta_1\beta_2\beta_3 \end{aligned}$$

is the reduced CNF representation and the ten critical sets are $\{\Phi_1, \Phi_2\}, \ldots, \{\beta_1, \beta_2, \beta_3\}$.

Dependency Cycles

The same strategy is used to analyze a system with dependency cycles: represent the system by an FDG, extract the associated equation set, solve the equations to find fault trees, then determine critical sets and failure probabilities. Consider the FDG shown in Figure 2. In this case there are dependency cycles. X, Y, and Z are generators and the associated functionalities, x, y, and z, are raw power. A and C are regulators so the functionalities a and c are smoothed power. The functionality a (respectively c) is available when the regulator A (respectively C), along with at least one input power source, is available. The functionality b is mission power. It depends on the correct functioning of combiner B and the availability of two of its three

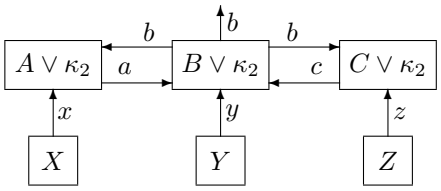

Figure 2: An FDG with cycles.

input power sources (a, c, and y), at least one of which must be smoothed.

The boolean failure conditions associated with the functionalities in this example are

$$\begin{array}{lll} a = A \vee \kappa_2(b,x) & b = B \vee \kappa_2(a,y,c) & c = C \vee \kappa_2(b,z) \\ x = X & y = Y & z = Z. \end{array}$$

and the general solution to these equations is

$$\begin{aligned} a &= A \vee BX \vee CXY \vee (C \vee Y \vee Z)X\theta \\ b &= B \vee AC \vee AY \vee CY \\ &\quad \vee (AZ \vee CX \vee XY \vee YZ \vee XZ)\theta \\ c &= C \vee BZ \vee AYZ \vee (A \vee Y \vee X)Z\theta \end{aligned}$$

where θ is a free boolean parameter, i.e., any boolean expression can be substituted for θ and the above will be a set of consistent simultaneous solutions to the original equations. Thus, the solution cannot be expressed in terms of just component failures.

This situation is unfortunate because, without fixing the value of θ, neither critical-set enumerations nor system failure probabilities are available. Rewrite these equations more abstractly as $v = \xi_v \vee \mu_v\theta$, where v is either a, b, or c, and it follows that

$$\begin{aligned} p(v) &= p(\xi_v \vee \mu_v\theta) \\ &= p(\xi_v) + p(\overline{\xi}_v\mu_v|\theta) \cdot p(\theta). \end{aligned}$$

Clearly, $p(v)$ cannot be calculated without $p(\theta)$. Further, unless θ and the literals are marginally independent, some conditional probabilities will be needed as well. All that can be inferred at this point is that

$$p(\xi_v) \leq p(v) \leq p(\xi_v \vee \mu_v),$$

a range that may be too large to be informative. The left and right end points of this interval correspond, respectively, to $\theta = \mathit{false}$ and $\theta = \mathit{true}$.

Computational techniques to solve simultaneous sets of functional equations, based on methods similar to those developed by Dionne, Mays, and Oles [1992], are described in Barnett and Verma [1994].

Fixed Points & Models

The simultaneous solutions to a set of boolean equations are called its fixed points. A fixed point is a value for each variable such that the result of substitution regurgitates the original forms. Consider a simple case with two equations, $x = \alpha + \beta y$ and $y = \gamma + \delta x$. The fixed points are all of the form $x = \alpha + \beta\gamma + \beta\delta\pi$ and $y = \gamma + \alpha\delta + \beta\delta\pi$, where π is arbitrary, because

$$\begin{array}{ll} x = \alpha + \beta y & y = \gamma + \delta x \\ \quad = \alpha + \beta(\gamma + \alpha\delta + \beta\delta\pi) & \quad = \gamma + \delta(\alpha + \beta\gamma + \beta\delta\pi) \\ \quad = \alpha + \beta\gamma + \beta\delta\pi & \quad = \gamma + \alpha\delta + \beta\delta\pi \end{array}$$

In the example of the previous section, the fixed points are the solution triples parameterized by θ. In general, the fixed points of simultaneous boolean equations are a parameterized family with one or more independent free boolean parameters.

Each fixed point represents a correct model of the system in some possible world that is consistent with its FDG. Figure 3 shows the most primitive non-trivial

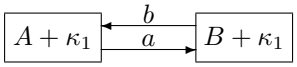

Figure 3: Engine with two cylinders.

FDG that has a dependency cycle.[1] Its fixed points are

$$a = A + B + \theta \qquad b = A + B + \theta.$$

Thus, three interesting cases occur: $\theta = \mathit{true}$, $\theta = \mathit{false}$, and θ is something else. Each case represents a valid system model. For the sake of this discussion, let us assume that A and B are cylinders in a combustion engine and that each depends on the other's proper performance to run smoothly.

When $\theta = \mathit{true}$, $a = b = \mathit{true}$; in other words, the system is in a failed state 100% of the time. This is a valid model because it is the state of the system when it is stopped—without a mechanism such as a starter motor there is no way to change this fact. On the other hand, when $\theta = \mathit{false}$, $a = b = A + B$ and each cylinder functions smoothly as long as both are operational. This is a valid model when the motor is initially operating. The system will continue to operate as long as neither cylinder experiences a break down.

The third possibility is that θ be neither *true* or *false*, e.g., $\theta = S$. In this case, $a = b = A + B + S$ and the system is equivalent to one with the augmented FDG, shown in Figure 4, where the fixed points are

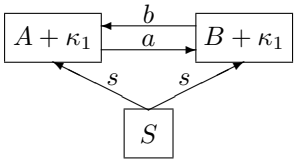

Figure 4: Augmented FDG.

$a = b = A + B + S + \pi$, and $\pi = \mathit{false}$ is chosen. The augmented system corresponds to one with an additional mechanism, say, a starter motor. Now the

[1] The most primitive example is a functionality that depends only on itself in a system with no components. In this case, the equation is $f = f$ and the fixed points are $f = \theta$ for an arbitrary θ.

system can operate correctly from either an initial idle or running state since it has an explicit mechanism to affect state transitions.

All three of the choices for θ are logically consistent with the original FDG: the augmented system is the original with $\theta = S + \pi$. The interesting difference is that the minimum fixed point of the augmented system can be used. Clearly, a system dependency, not present in the original, has been uncovered and, just as clearly, the modeling task would be better served if the derived FDG were used in place of the original assuming that it represents the actual intent of the system designers.

The set of fixed points that satisfy a set of boolean equations form a lattice. Let $f_i = f_i(\theta_1, \ldots, \theta_m)$ be the functionalities in a system and the θ_j be the free parameters that occur in their fixed points. Then

$$\begin{aligned} f_i(\theta_1, \ldots, \theta_m) &\subset f_i(\theta'_1, \ldots, \theta'_m) \\ p(f_i(\theta_1, \ldots, \theta_m)) &\leq p(f_i(\theta'_1, \ldots, \theta'_m)) \end{aligned}$$

if $\theta_j \subset \theta'_j$. (The relation "$\subset$" can be thought of as referring to area in a Venn diagram.) Clearly, $\theta_j = \mathit{true}$ and $\theta_j = \mathit{false}$ lead, respectively, to upper and lower bounds[2] on the probabilities and area of the functionalities. Thus, these assignments are called, respectively, the maximum and minimum fixed points.

It should be noted that the critical sets derived for the minimum fixed point are failure sets[3] for *every* model consistent with the FDG. Thus, the failure condition represented by the minimum fixed point is the intersection of the failure conditions of all models consistent with the FDG. Another way to say this is that the minimum failure condition is the one structurally implied by the FDG. Similarly, the failure condition associated with the maximum fixed point is the union of the failure condition of all models consistent with the FDG.

Representational Adequacy

An important goal of reliability analysts is that their analyses reflect all significant dependencies. Therefore, the FDG derived from the product design must include all necessary components and properly express interdependencies. If an FDG does this, then the values substituted into the fixed points for the free parameters are necessarily functionality- and component-free. In other words, the only reasonable substitutions for the θ_j are the boolean constants *true* and *false*.

[2]This is so because functionality fixed points do not contain any negated θ_j terms.

[3]A failure set is a set of components whose simultaneous failure entails system failure. A minimum failure set is a critical set.

In theory, this means that the designer/analyst can examine the maximum and minimum solutions and the critical-set enumeration they yield to see if either agrees with his intuition about the system. If neither is satisfying, the FDG is not an adequate model and needs to be amended.

In practice, this is not a reasonable approach. The table in Figure 5 summarizes some facts about four typical systems that have been analyzed by the author. As is apparent, the number of boolean operators appearing in the fixed point expression of the main system functionality is of the order of magnitude of several hundred to several thousand. In addition, the number of critical sets can be quite large. Thus, these structure are seen to be too large to be well-understood by manual inspection.

Case	Components	Boolean Operators	Critical Sets
MCB	23	724	131
FCB	28	1992	479
HP	42	235	16,293
AB	58	259	548,754

Figure 5: Typical test-case complexity.

Limit Cycles

The control theory concept of limit cycles is a useful metaphor to understand the fact that FDGs do not have unique solutions. A limit cycle is a steady state that a system enters from a given set of initial conditions. It is called a "cycle" since the steady state may be a regular tour of quasi-stable states. As an example, consider a set of coupled vibrators. The steady states of the system are various harmonics of some base frequency. The steady state entered depends on the initial state of the system. For some sets of initial conditions, there are multiple high-probability possibilities.

Consider the two-cylinder engine example introduced above. There are two steady states: (1) idle and (2) rotating smoothly. Given initial conditions expressed in "rpms" and smoothness, the engine will either eventually halt or achieve smooth rotation. Clearly, we cannot know which state is achieved unless the initial conditions and the transition function are known. Since the FDG specifies neither, multiple solutions necessarily appear.

The fixed points of the functionality equations correspond to the possible limit cycles of the analyzed system. Hence, without some additional information, it is no more possible to recognize the proper system failure condition than it is to predict its limit cycle.

There is more to the metaphor of fixed points and limit cycles than the fact that both admit multiple solutions: multiplicity in both is a result of dependency cycles (aka coupling) and both may require several independent parameters to properly specify. The cause of multiplicity is that components in a loop must achieve a joint steady state (fixed point) that satisfies mutual constraints and several choices are consistent with these constraints.

The number of independent parameters necessary to describe the possible steady states is a function of the number of loops in the system description and the strength of coupling among the loops. For example, the system shown in Figure 2 has two loops, $A \leftrightarrow B$ and $B \leftrightarrow C$, however, only one parameter is necessary.

This author has been unable to find a computationally efficient way to predict the minimum number of independent free parameters necessary to represent the fixed point of an FDG. The solution techniques introduced in Barnett & Verma [1994] allow us to find the valid fixed points, albeit, with more parameters than necessary. However, while these parameters are independent (the choice of boolean expression to substitute for each can be arbitrary), they are not free (the choice for some parameters can mask the effect of the choices made for others). Resolution of such problems belongs more to the fields of boolean algebra and physical system modeling than it does to reliability analysis.

A Better Approach

What is needed is technology that helps the designer/analyst to

1 Form an FDG from the system design, and

2 Select the fixed point that provides the best model of the intended system.

These are co-problems because, fixed points other than the minimum or maximum can occur only if there are either inter- or intra-system dependencies not reflected in the FDG. Further, several fixed points can be valid descriptions of the same system in different situations. This is no different than a control system designer relying on different limit cycles in different situations. In order to properly model multi-modal systems, it may be necessary to use multiple FDGs. When this is desired, another item must be added to our wish list: one that helps the designer/analyst

3 Modify an existing FDG and/or select a different fixed point when the system is used in a related but different mode.

These aids are ideal applications for AI and expert systems techniques: models of the interactions of intricate physical mechanism are necessary along with heuristic knowledge to tame inherent computational complexity and select acceptable approximations.

Further, this problem domain displays enough regularity to suggest many reasonable rules of thumb; for example, the pervasive use of redundancy to promote reliability may suggest ways to factor FDGs into smaller, more manageable units that are easier to understand and evaluate, i.e., identify groups of subsystems that perform the same or similar function.

Another clue that suggests interesting factorizations and reuse of heuristics, is knowing which components are identical to one another. For example, the regulators, A and C in Figure 2, are two instances of the same component as are X and Y. Further, the *usage* of A and C by B are identical so that we should expect symmetry in the analysis between A coupled with X and C coupled with Y. This is not obvious from the FDG but might be inferred from the original design or knowledge about the basic components included in the design. A smart tool would need to draw these types of inferences or at least be clever enough to inquire about them when basic design symmetries are detected.

We saw above that the source of ambiguity in reliability analyses is the presence of dependency cycles. A good heuristic is to identify loops in the FDG and interact with the designer/analyst to determine their intended steady-state behavior. If necessary, additional mechanisms—such as the starter motor appearing in Figure 4 but not Figure 3—may need to be added to model the intended operation of the loop elements.

In some instances, the mechanisms added to the system description will not be physical components. Rather, they might be symbols to represent design assumptions. For example, instead of representing a starter motor, S might represent assumptions about the initial state of the system. If it does, $p(S)$ would represent the strength of belief that the system fails to achieve an operational steady state because of the initial state. This interpretation of the augmented system depicted in Figure 4 is better than that of a starter motor unless, the derivation of the FDG actually overlooked an existing system component.

Open Problems

The approach outlined in the previous section is only half-baked. Research results from several areas are necessary to pursue the agenda that it entails. Symbolic modeling of physical systems (qualitative physics) is the current focus of a substantial research community. Clearly, that work will provide valuable insights to our endeavor. Many of the relevant results are chronicled in the AI Journal and the Proceedings of the AAAI

and IJCAI conferences of the last several years.

Another source of valuable results that address efficiency and computability will be provided by research in graph theory. In particular, the algorithms developed for compilers (e.g., loop detection and common subexpression detection) and work with commodity flow and other economical models will be useful.

Finally, the vast body of knowledge developed for the analyses of control and feedback systems will be useful. This may seem surprising since their mathematics is very different from ours; they model continuous worlds with differential equations while we model discrete worlds with boolean equations. However, the deep metaphor between multiple fixed points and multiple limit cycles suggests much commonality. After all, both communities describe and model the same sorts of systems.

References

Amstadter, B. L., 1971. *Reliability Mathematics*. McGraw-Hill.

Barlow, R. E., Fussell, J. B., and Singpurwalla, N. D. (eds). 1975. *Reliability and Fault Tree Analysis*. SIAM.

Barnett, J. A. and Verma, T., 1994. Intelligent reliability analysis. To appear in *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*.

Dionne, R., Mays, E., and Oles, F. J., 1992. A Non-well-founded approach to terminological cycles. *Proceedings of the Tenth National Conference on Artificial Intelligence*. pp. 761–766.