

What Do You Do When You Run Out of Computer?  
Clifton A. Ericson II  
Applied Ordnance Technology, Inc.

Abstract

Fault Tree Analysis (FTA) is a methodology for modeling and analyzing complex systems for root cause analysis and probabilistic measures of safety, reliability and risk. As might be expected, FTA of large complex systems quite often results in very large complex fault trees. Occasionally some fault tree models are too large to be successfully evaluated by a fault tree code (i.e., computer program). In this case the analyst has essentially “run out of computer”, that is his computer is unable to generate the Cut Sets for the fault tree. What does the FTA analyst do in this situation? There are various causes for this problem, and methods for resolving them, all of which are described in this paper.

Acronyms

For purposes of brevity the following acronyms and abbreviations will be used throughout this paper:

- FT - Fault Tree
- FTA - Fault Tree Analysis
- CS - Cut Set
- MinCS - Minimum CS
- DupCS - Duplicate CS
- SupCS - Super CS
- MOE - Multiple Occurring Event
- MOB - Multiple Occurring Branch

Overview

FT computer codes have a FT Solver, whose basic goal is to generate all of the MinCS's for the FT. However, in order to do this, the FT Solver often has to reduce and eliminate many non-MinCS's until only the MinCS's are left. For reasons discussed below, this often becomes too large and difficult a task for the FT Solver, and the program crashes.

This paper deals with Tree Pruning methods for solving FT's that are too large/complex for the computer to solve. The theory behind pruning is to reduce the number of CS's, thereby making the job simpler for the FT solver.

Figure 1 demonstrates the basic theory in FT pruning or reduction. Figure 1a represents a FT consisting of all OR gates that has 10 CS's and is (hypothetically) too large for the computerized FT program. By eliminating or pruning some

branches, shown by the dotted ovals, the number of CS has been reduced to 5, as shown in Figure 1b. *The objective is to judiciously prune without adversely affecting the final computational results.* In this example the FT has been manually reduced in size, and it is now solvable by the computer.

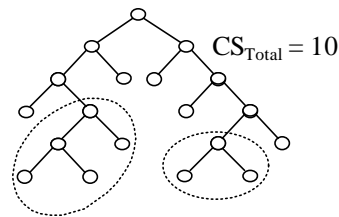


Figure 1a – Full Fault Tree

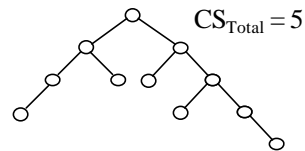


Figure 1b – Pruned Fault Tree

Definitions

The following definitions are provided to help understand some of the important concepts involved in FT pruning:

- 1) Cut Set – A unique set of events that together cause the top event to occur.
- 2) Min CS – A CS where if one element is removed, will no longer cause the top event to occur.
- 3) DupCS – A repeated CS.
- 4) SupCS – A non-minimal CS (an element can be removed and the top event will still occur).
- 5) MOE – A repeated event that occurs in more than one place in the FT.
- 6) MOB – A repeated branch that occurs in more than one place in the FT. All events within an MOB are effectively MOE's, thus an MOB is a super-MOE.

The Problem

FT computer codes are sometimes unable to solve large and complex FT's, resulting in a

computer or program crash, and an unsolved FT. There can be many reasons why a FT Solver is unable to solve a particular FT.

Some of the more common causes for failure of FT solvers include:

1. The FT is too large
2. The FT is too complex
3. Inadequate computer resources
4. Inadequate FT program/algorithm
5. Combinations of the above

Most FT analysts have experienced all of these causes at one time or another. Item 3, inadequate computer resources refers to the situation when the user's computer just does not have enough horsepower to solve the FT. That is, the computer's processor is too slow, or the computer has inadequate memory. Item 4 refers to a computer program designed to solve FT's, but the program's algorithm design is inadequate for large FT's. This can happen even with expensive commercial FT Solvers. With large complex FT's it is difficult to solve FT's strictly through Boolean algebra, and it therefore becomes necessary to resort to various reduction algorithms. Unfortunately most algorithms have a breaking point.

The fact that FT's quite easily, and quite often become very large and complex is the primary cause of failure for most FT Solvers. *What large and complex really means is that there are too Many CS's for the computer to resolve.* This is because complex FT's have a large number of non-minimal CS's, and the non-MinCS's must be crunched by the computer in order to eventually produce only minCS's. *The tree probability calculation is generated from the MinCS's only*, therefore, DupCS's and SupCS's must be removed.

FT's can become large and complex for several basic reasons. First, the system being analyzed is large and complex, and this factor directly translates into the FT. Second, CS complexity and growth generally results from Multiple Occurring Events (MOE) and Multiple Occurring Branches (MOB) in the FT. MOE's and MOB's both increase FT size, complexity and the total number of CS's. Third, multi-phase fault trees significantly drives up the total number of CS's in a FT.

In general, FT's grow in size and complexity from the following causes:

- OR gates - increases quantity of CS's
- AND gates - increases order of CS's
- MOEs - increases quantity of CS's and generate DupCS's
- MOB's - increases quantity of CS's and generate DupCS's and SupCS's

#### The Solution

There are several possible solutions for reducing a large quantity of CS's to only MinCS's:

- CS Truncation (pruning)
- FT Reduction (pruning)
- Increase computer capacity
- Improve FT code
- Use simulation rather than analytical approach

Each of these possible solutions has merit, except that *FT pruning is generally the only option within direct control of the analyst.* He is usually provided with a computer and a fault tree program. If they are not able to solve his FT, he must then resort to FT pruning.

There are two basic methods of FT pruning:

1. Truncation by probability or order
2. Reduction by functionality

Truncation refers to eliminating CS's based on CS probability and/or CS order. Very small probability CS's can be removed because they do not add significantly to the total tree probability. High order CS's (e.g., six input AND gate) can be removed, because in theory they would result in a small probability. Of course care must be taken to ensure the probabilities are relatively small, and they would in total not add up to a significant value. *This method of truncation of course does have an impact on the total tree probability.*

FT Reduction refers to eliminating tree nodes and branches based on tree structure functionality. Tree structures with MOE's and MOB's generate DupCS's and SupCS's. Certain MOE/MOB patterns in the tree structure allow for pruning of some MOE's and MOB's prior to computation. This in effect eliminates the DupCS's and SupCS's for the computer. The non-MinCS's would have to be removed anyway. *This method of truncation does not have an impact on the total tree probability.*

Removing select MOE/MOB's from the FT calculation can provide a great return on effort.

MOB's are responsible for creating a significant quantity of DupCS's and Sup's, which would have to be removed by the FT Solver. The larger the MOB subtree, the greater the proliferation of DupCS's and Sup's. Therefore, manual preprocessing the removal of these nodes from the FT calculation makes the job easier for the FT Solver.

In order to perform FT Reduction it is necessary to understand how non-minimal CS's are created. Figure 2 demonstrates how DupCS's and SupCS are derived from a FT. In this example, a simple MOB consisting of two events generates two additional non-MinCS's. These non-MinCS's must be removed to obtain the necessary MinCS's. If the number of failure nodes below MOB X1 were larger, there would also be many more non-MinCS's.

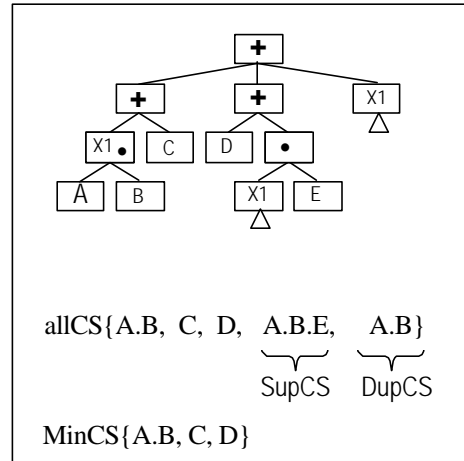


Figure 2 – FT With DupCS and SupCS

The following laws of Boolean Algebra are the basis for reducing and eliminating non-MinCS's:

- 1)  $A + A = A$
- 2)  $A \bullet A = A$
- 3)  $A + A \bullet B = A$

FT MOE/MOB Reduction Patterns

The following six cases help develop some generalized rules that can be used in pruning MOE's and MOB's from a FT.

Case 1	Case 2	Case 3
<p>All OR gates to Top on both sides.</p> <p>CS={A, B, C, D, E, A, F}                      = {A, B, C, D, E, F}</p> <p><math>A + A = A</math></p>	<p>OR gates to top on both sides, except AND gate immediately above the right side MOB.</p> <p>CS={A, B, C, D, E, A.F}                      = {A, B, C, D, E}</p> <p><math>A + A \bullet F = A</math></p>	<p>OR gates to top on left side, right side is OR gates, except AND gate higher above Right MOB.</p> <p>CS={A, B, C, D, E.A, E.F}                      = {A, B, C, D, E, F}</p> <p><math>A + A \bullet E = A</math></p>
<p>Rule: Delete either one of the MOB's.</p>	<p>Rule: Delete subtree from AND gate down on <u>right</u> side.</p>	<p>Rule: Delete MOB below the AND gate on <u>right</u> side.</p>

Case 4	Case 5	Case 6
<p>OR to top on both sides, AND directly at the top.</p> <p>CS={A.D, A.E, A.A, A.F, B.D, B.E, B.A, B.F, C.D, C.E, C.A, C.F}</p> <p>= {A, B.D, B.E, B.F, C.D, C.E, C.F}</p> <p><math>A \bullet A = A</math>  <math>A + A \bullet D = A</math></p>	<p>All AND on right side, OR gates on left side, AND gate at top.</p> <p>CS={A.D.E.A.F, B.D.E.A.F, C.D.E.A.F}</p> <p>CS={A.D.E.F}</p> <p><math>A \bullet D \bullet E \bullet A \bullet F = A \bullet D \bullet E \bullet F</math>  <math>A \bullet D \bullet E \bullet F + B \bullet D \bullet E \bullet A \bullet F = A \bullet D \bullet E \bullet F</math></p>	<p>AND gates on all right MOB branch, except top.</p> <p>CS={A, B, C, D.E.A.F}</p> <p>CS={A, B, C}</p> <p><math>A + D \bullet E \bullet A \bullet F = A</math></p>
<p>Rule: Unable to delete either MOB. Resulting CS's would not produce CS A.</p>	<p>Rule: Can delete entire <u>left</u> side of top AND gate.</p>	<p>Rule: Can delete entire <u>right</u> side of top OR gate.</p>

Each of these cases provides a proof on why and how certain MOE/MOB patterns allow for the removal of the MOE/MOB from the tree prior to CS resolution.

The following generalized Rules for FT pruning by MOE/MOB reduction have been established:

- 1) If all OR gates between MOB's and top, then delete either MOB.
- 2) If the top is an AND gate, with OR gates directly below on both sides, then an MOB cannot be deleted without restructuring the tree.
- 3) If one MOB branch is all AND gates except for the top, then completely delete that entire branch.
- 4) If one MOB branch is all AND gates, including the top, then delete the entire opposite branch into the AND gate.
- 5) If one MOB has an AND gate immediately above it, then an OR gate, then the entire AND gate branch can be deleted.
- 6) If one MOB has an AND gate higher above it, then delete that MOB.

The six cases above used a single node as proof for an entire MOB. Is this valid even with nodes and subtrees below the MOB? Figure 3 demonstrates that this is acceptable by re-analyzing Case 1 with events below the MOB.

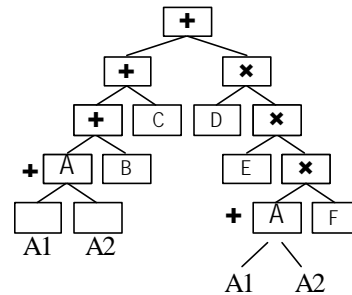


Figure 3 – Reduction with MOB Events

CS={A1, A2, B, C, D.E.A1.F, D.E.A2.F}

CS={A1, A2, B, C}

$A1 + D \bullet E \bullet A1 \bullet F = A1$

### Conclusion

FT Pruning is a viable solution for solving FT's that are too large and complex for a computer program to solve. Pruning techniques can be incorporated into the computer program, or they can be applied manually prior to computer resolution of the FT. So, when you run out of computer, you may have to resort to FT pruning.

Removing nodes and branches from the FT (for CS computation only) is a very serious business. Some pruning methods can be performed without any impact on the final CS solution and the final quantitative number. However, other pruning methods do have an impact. They actually eliminate low probability CS's and thereby affect the final probability calculation. The trick is to do this judiciously so that the quantitative impact is numerically insignificant.

### References

- [1] C. A. Ericson II, FTAB -- A New Generation Computer Code For Fault Tree Analysis, 15<sup>th</sup> International System Safety Conference, 1997.
- [2] C. A. Ericson II, Fault Tree Analysis By Design, 16<sup>th</sup> International System Safety Conference, 1998.
- [3] C. A. Ericson II, Fault Tree Analysis – A History, 17<sup>th</sup> International System Safety Conference, 1999.
- [4] C. A. Ericson II, Accident Investigation Using EEFTA, 18<sup>th</sup> International System Safety Conference, 2000.
- [5] C. A. Ericson II and John D. Andrews, Fault Tree and Markov Analysis Applied to Various Design Complexities, 18<sup>th</sup> International System Safety Conference, 2000.
- [6] N. H. Roberts, W. E. Vesely, D. F. Haasl & F. F. Goldberg, Fault Tree Handbook, NUREG-0492, 1981.

### Biography

Clifton A. Ericson II  
Applied Ordnance Technology, Inc.  
6406 Medallion Drive  
Fredericksburg, VA 22407 USA

phone 540-663-3104  
fax 540-663-3273  
email: cliftonericson@cs.com  
ericson@aot.com

Mr. Ericson currently works for Applied Ordnance Technology, Inc. as a system safety project manager. Prior to joining AOT, he worked in System Safety Engineering and Software Engineering for 35 years at the Boeing Company, in Seattle, WA. He has been involved in all aspects of system safety engineering including hazard analysis, fault tree analysis, software safety, safety certification, safety documentation, research, new business proposals, safety audits and safety training.

He has performed Fault Tree Analysis and Hazard Analysis on most major Boeing products, including Minuteman, SRAM, ALCM, Apollo TIE, Morgantown Personal Rapid Transit, B-1, AWACS, Space Station, Hydrofoil and 737/757/767 systems. He is the developer of the MPTREE, SAF and FTAB fault tree computer programs. He also worked as a programmer/analyst for Boeing Computer Services, and performed avionics software research for Boeing Commercial Aircraft.

He has been involved in all aspects of fault tree development since 1965, including analysis, computation, multi-phase simulation, plotting, documentation, training and programming. He has written papers on software safety and taught software safety at the University of Washington. Mr. Ericson holds a BSEE from the University of Washington and an MBA from Seattle University. He is currently President of the System Safety Society, and is on the technical review committee for the Journal of System Safety and the Journal of Process Mechanical Engineering (UK).