Conf-841254--1

MINIMAL CUT-SET METHODOLOGY FOR ARTIFICIAL INTELLIGENCE APPLICATIONS[*]

C. R. Weisbin, G. de Saussure, J. Barhen, E. M. Oblow, J. C. White[**]

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831

CONF-841254--1

DE85 004779

ABSTRACT

This paper reviews minimal cut-set theory and illustrates its application with an example. The minimal cut-set approach uses disjunctive normal form in Boolean algebra and various Boolean operators to simplify very complicated tree structures composed of AND/OR gates. The simplification process is automated and performed off-line using existing computer codes to implement the Boolean reduction on the finite, but large tree structure. With this approach, on-line expert diagnostic systems whose response time is critical, could determine directly whether a goal is achievable by comparing the actual system state to a concisely stored set of preprocessed critical state elements.

I. INTRODUCTION

For several years now, the scientific community has employed probabilistic risk assessment (see, e.g., Ref. 1) in an attempt to quantify the likelihood of occurrence and the consequence of various plausible accident-related scenarios. One formalism adopted in the implementation of probabilistic risk assessment is that of the theory of minimal cut sets.[2] The purpose of this paper is to explore applications of the minimal cut-set approach to artificial intelligence problems; we consider expert diagnostic systems and planning systems.

Section II briefly reviews several of the expert diagnostic systems recently completed or under active development. Many of these systems depend upon tree searching for diagnosis, wherein the search is facilitated through either hierarchical partition and/or optimal path selection using algorithms such as A^*.[3] Section III discusses the motivation behind the minimal cut-set methodology and illustrates the procedure with a sample problem. Section IV considers the extension of cut sets to problems in planning and problem solving. Finally, Section V summarizes the conclusions drawn from this study.

The approach suggested in this paper is appropriate only for the important class of real world situations which can be adequately represented using complex, but finite (albeit large) logical tree structures (e.g., risk assessments, circuit diagnosis). The major savings would be for systems which must respond rapidly to requests for information or procedures (i.e., tree search must be minimized). The savings comes from the explicit off-line preprocessing of large, finite trees using existing computer codes. The preprocessed results include a concise set of critical state elements (i.e., cut sets) which are retrieved rapidly (via lookup table) on-line by comparison of system state components to cut-set elements.

II. EXPERT DIAGNOSTIC SYSTEMS AND CURRENT METHODOLOGY

There is much current activity and an emerging literature in the area of expert-diagnostic systems to facilitate trouble shooting of electronic equipment. Many of the current procedures are based upon tree search of one type or another. Vesonder et al.,[4] have reported on the ACE system being developed to provide timely and accurate selection of geographical areas whose cable lines may require rehabilitation. Some of the system design decisions follow the earlier work of McDermott.[5] Basically, ACE uses exhaustive search relying on domain knowledge for significant pruning of the space of possible conclusions. A similar problem was discussed by Bennett and Hollander[6] who describe an automated consultant that advised field service personnel on the diagnosis of faults occurring in computer installations. Search allows useful inferences to be drawn during the consultation.

Williams et al.,[7] extend the conventional search approach in their system which addresses fault isolation in a geographically distributed communications network. Available diagnostic tests impose a refinement hierarchy on the space of hypotheses, enabling exploitation of hierarchical search. Cantone et al.,[8] extend further the search procedure for their electronics troubleshooting diagnostic system by using and extending game tree search techniques. Cantone et al., use the cost of each test, the conditional probability of test outcome, and the estimated proximity to solution as parameters which direct the search and are modified

---

[*] Research sponsored by the Office of Basic Energy Sciences, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

[**] Birmingham-Southern College, Birmingham, AL.

gsw

on-line as new information is acquired. Davis[9] and Nelson[10] explore the interaction of simulation and inference, looking for differences between them as starting points for further investigation.

The one common denominator behind much of this existing work is the concept of tree search, facilitated through either hierarchical partition and/or optimal path selection. In the next section, we discuss a method which, if given a finite (albeit large) logical tree comprised of

- a top event of interest,
- logic gates (AND/OR) as a physical descriptive model, and
- independent initiating events,

attempts to preprocess this information into a much-reduced equivalent tree. This new tree eliminates the need for a substantial part of the on-line search time required to solve diagnostic problems, while greatly improving physical understanding.

III. MINIMAL CUT-SET APPROACH

A. Motivation

As will be illustrated in more detail below, the minimal cut-set approach[2] is intended to simplify a complicated tree structure with numerous "AND"- and "OR"- gates through the use of Boolean algebra. The motivation is to achieve improved clarity for complex real-world problems (logic tree structures for a nuclear reactor accident description could be tens to hundreds of pages), to rank essential contributors (or sequences) to system failure, and to quantitatively assess the likelihood of the top event occurrence.

The minimal cut-set approach has close relationship to branch and bound techniques and the dynamic programming principle which seek to discard redundant paths to improve efficiency.[11] However, the latter techniques were intended to apply during tree search. We envision the logic (fault) trees being drawn by scientists/engineers (perhaps teams of these independently constructing subsystems) with expertise in the physical domain of interest, and not necessarily by numerical analysts whose concern is in efficient tree representation. The resulting logic tree may be immense (but finite), contain much redundancy, and obscure through complexity much of the interesting information. Rather than continued on-line search through such a large tree, off-line *preprocessed* automated construction of a simplified representation would be helpful to reduce the time required to identify whether interesting states of the world (unusual faults, desired goals) can be achieved from other (previous) world states, and what are the key state elements which govern this decision.

B. Procedure

A fault tree is a logic diagram depicting certain events (initiating events) and the consequences of their occurrence. Such a tree represents graphically a set of Boolean equations which can be analyzed using Boolean algebra. We use these algebraic rules to deduce minimal cut sets which are the smallest combination of component failures which cause the top event to occur. These cut sets are entirely equivalent to representing the original logic tree in normal disjunctive form. The collection of single-component, minimal cut sets represent those single failures which will cause the top event to occur. Two-component minimal cut sets represent the double failures which together will cause the top event to occur. For an n-component minimal cut set, all n components in the cut set must fail in order for the top event to occur.

We describe the process of cut-set determination through an illustrative example. Consider the fault tree in Fig. 1. T_e represents the top event of interest, A through D represent potential initiating events, and rectangles R_1 through R_5 represent intermediate states. The AND/OR logic is labelled with being AND, being OR. Such trees are constructed by domain engineers following the physics (e.g., wiring diagrams) of the given problem. The equivalent Boolean equations for this simple example are:

$$T_e = R_1 \cdot R_4 \quad (1)$$
$$R_1 = A + R_2 \quad (2)$$
$$R_2 = B + R_3 \quad (3)$$
$$R_3 = C \cdot A \quad (4)$$
$$R_4 = C \cdot R_5 \quad (5)$$
$$R_5 = D \cdot B \quad (6)$$

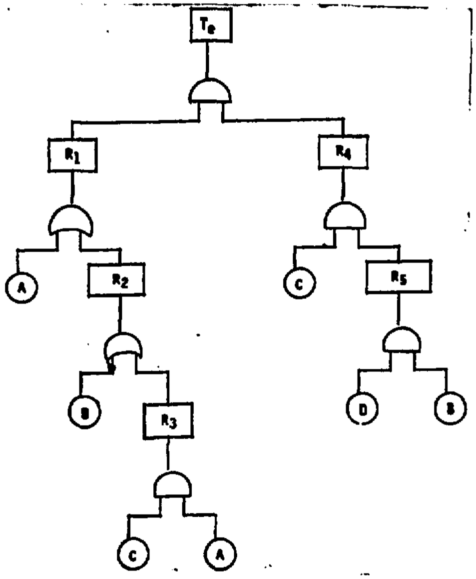

Fig. 1. An Illustrative Fault Tree Simulating the Domain Physics

An OR gate is equivalent to the Boolean symbol "+", (i.e., for $R_1 = A + R_2$ either of the events A or R_2 or both must occur in order for R_1 to occur). The AND gate is equivalent to the Boolean symbol "•", (i.e., for $T_e = R_1 \cdot R_4$ both R_1 and R_4 must occur if T_e is to occur).

Performing top down substitution using the laws of absorption $\{(X + X \cdot Y = X),$ $(X \cdot (X + Y) = X)\}$ and the idempotent laws $\{(X \cdot X = X),\ (X + X = X)\}$ we find:

$$T_e = B \cdot C \cdot D \tag{7}$$

This can be represented in the much simpler diagram of Fig. 2.

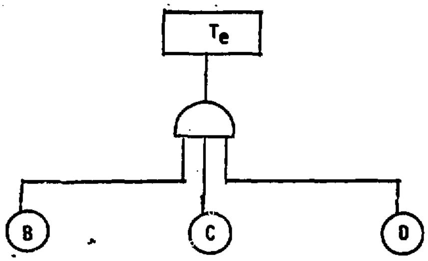

Fig. 2. Boolean Equivalent of Fig. 1

Note that event A does not appear in Fig. 2, so it's success or failure is irrelevant to the occurrence of the top event T_e. In this simplified problem, we find a single three-component cut set $B \cdot C \cdot D$, all of which must occur for T_e to occur. In other problems we may find several cut sets about which we can focus our attention or direct further tests. (See Section IV.) Overall however, a significant reduction in complexity is seen even further when one considers more realistic cases of highly complicated and redundant trees.

Although the solution to this simplified problem may have been seen without this method, system failure models for more complicated structures will be difficult to arrive at by inspection and quite time consuming to determine by search. The minimal cut-set computation provides an analyst with a thorough and systematic method for identifying the basic combinations of component failures which can cause undesired events.[2] For small fault trees, the determination of minimal cut sets can be done by hand. For larger trees, various computer algorithms and codes for fault-tree evaluation are available.[12-14]

With a set of minimal cut sets, and their associated probabilities for initiation obtained in an automated off-line procedure, it is clear that for rapid on-line analysis:

(1) The description of the fault causes can be clearly presented.
(2) The probability of the top event can be easily computed, given the probability of the initiating events.
(3) The contributor sequences to the top-event fault can be quantified and ranked.

IV. THE CONCEPT OF MINIMAL CUT SETS APPLIED TO PLANNING

A. General Method

In this section we illustrate, with a simple example, how the concept of minimal cut sets might be used by a planner, or a problem solver.

We assume that a World Model (WM) can be represented as a set of N clauses which can assume the value T (true or 1) or F (false or 0). These clauses are the assertions of facts that are known to the planner to be true or false. We define a state of the WM to be a particular instantiation of the values of each of the N clauses. Hence, the state space of the WM is the set of 2^N points on the unit hypercube in N-dimensional binary space. All 2^N points in the WM may not be physically or logically realizable states of the WM; restrictions may be introduced of the form $C_i \cdot C_j = 0$ (clauses C_i and C_j are incompatible, or have an empty intersection) or $\sum_{i\epsilon I} C_i = 1$ (one and only one of the clauses C_i with $i \epsilon I$ must be true). Each of the above relations between clauses defines a subspace on the unit hypercube which is "forbidden" to physically meaningful states of the WM.

In addition to the World Model the Planner employs a set of production rules. These production rules are mappings from the state space onto itself; the rule operates on one state of the WM to produce a new state. As is often done, we assume the production rules to comprise three lists; a precondition list defines the domain of the rule, an add list specifies the clauses that are made true by the application of the rule, a delete list specifies the clauses that are made false.[15,16] If the precondition list of a rule specifies the truth value of m clauses, this defines a subspace of dimension N-m as the domain of the rule.

For a given set of goal states, a "fully developed AND/OR search tree" may be constructed by the algorithmic procedure described on Fig. 3, for the simple case where the delete lists are empty. By construction, the tree specifies all the combinations of true clauses from which the goal states may be reached. Such a tree is analogous to a fault tree specifying all the combinations of initiating events that can cause the

---

(1) Label the root of the tree T_e and place the clause representing the goal states under it.

(2) If the following 3 conditions are satisfied by a clause C_i:

  (a) C_i is at the end of a branch of the tree
  (b) C_i is in the "Add-list" of a rule R_j
  (c) There is no node labelled R_j between T_e and C_i

Then replace the "leaf" C_i by a two-branch, OR-node labelled R_j with one branch terminating with the leaf C_i and the other branch pointing to an AND-node, labelled S_j with branches pointing to all the clauses in the "precondition list" of the rule R_j. Then go to (2).

Else terminate.

Fig. 3. Rule for the Construction of a Fully Developed Search Tree, Special Case where Delete Lists are Empty.

top event. The Boolean equations represented by the tree may be reduced to a set of minimal cut sets using top-down substitution and elimination through the rules of Boolean algebra. (If there are compatibility relations among the clauses of the WM specifying forbidden regions in state space, the minimal cut sets may be further reduced.)

As an illustration, Table 1 specifies six production rules and two compatibility relations for a World Model consisting of the conjunction of 10 clauses, C_i, $i = 0, \ldots 9$. For purposes of illustration we consider here the simpler case where the rule delete list is empty. For the set of goal states specified by the clause C_0 being true, Fig. 4 shows the "fully developed AND/OR search tree" and Fig. 5 presents the corresponding reduced tree and shows the seven minimal cut sets. [Note that in conformity with standard useage, we have used slightly different notation for fault trees (Figs. 1 and 2) and search trees (Figs. 4 and 5).]

Table 1. Production Rules and Compatibility Relations Corresponding to the Tree of Fig. 4

Rule Name	Preconditions	Add List
R_1*	$C_6 \cdot C_8 \cdot C_9$	C_0
R_2	$C_4 \cdot C_7$	C_6
R_3	$C_3 \cdot C_5 \cdot C_7$	C_8
R_4	C_4	C_5
R_5	C_2	C_4
R_6	C_1	C_3

Compatibility relations: $C_3 \cdot C_4 = 0$ $C_6 + C_7 = 1$
Goal states: States for which C_0 is true.

*Read if clauses C_6, C_8 and C_9 are true then rule R_1 can be applied and clause C_0 becomes true.

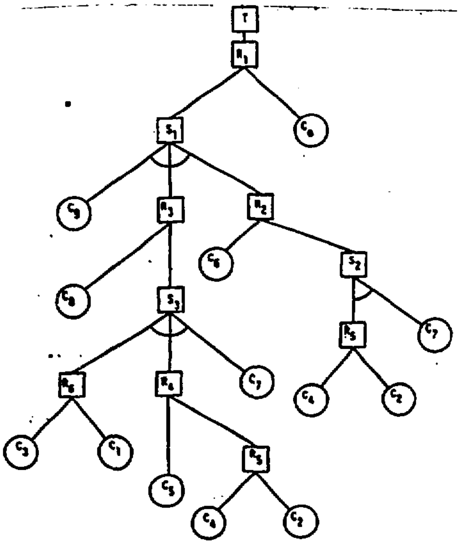

Fig. 4. Fully developed search tree for goal states C_0.

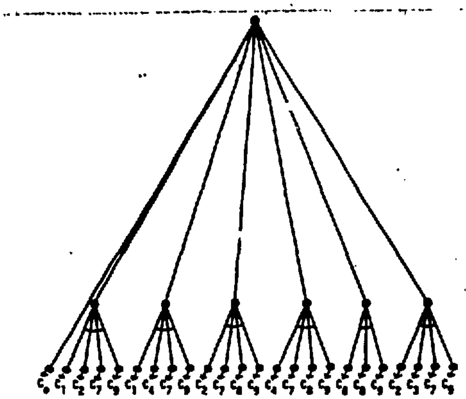

Fig. 5. AND/OR tree equivalent to the fully developed search tree of Fig. 4.

Geometrically, a minimal cut set consisting of m clauses defines an N-m subspace, in the N dimensional space of the WM, from which some goal state can be reached by the production rules. The union of all minimal cut sets defines the regions of state-space from which a goal state can be reached.

B. Possible Application

A World Model consisting of a large number of clauses and with many production rules can be "preprocessed" to obtain the minimal cut sets corresponding to several goal states of interest, these minimal cut sets can then be stored in tabular form. A planner can then immediately determine if a given goal state can be reached from the present state of the WM. Note that preserving the cut sets corresponding to a given goal is not the same as explicitly preserving all possible states of the world. For a complex model of a real-world application described using a finite tree, the cut-set formulation is a concise representation of storing only independent sets of critical state components essential to achieve the goal of interest. For storage economy the entire search tree has not been preserved, but by inspection of the minimal cut sets the planner can determine at once if the WM is in a state from which perserverance in the search will be rewarded, or, if the WM is in a state from which perserverance will only lead to frustration.

When planning in a hierarchy of abstraction spaces,[16] the minimal cut sets on a high level of abstraction can be used to assign criticality values to the important clauses for further exploration in lower hierarchical spaces.

Finally, the total amount of initial processing has not been reduced; the large tree is processed in the determination of the cut sets. The essential feature is that the bulk processing is done off-line, reducing greatly the on-line computation by simplifying complex trees to single-level combination of events in disjunctive normal form.

V. CONCLUSION

The ideas outlined in this paper support the possibility of utilizing a methodology developed for the analysis of fault trees in risk analysis, to other areas of interest in AI. After a presentation of the concept of minimal cut sets, the paper sketches possible application in the fields of Expert Systems, Problem Solving and Automatic Planning. Some simple illustrations are given.

A complete and rigorous review of all the possible applications of the concept of minimal cut sets to the many areas of interest to AI workers is beyond the scope of the present discussion, but it is felt by the authors that further exploration of the ideas sketched here may be of significant value.

ACKNOWLEDGEMENTS

The CESAR program is supported by the Department of Energy's Office of Basic Energy Sciences. We sincerely appreciate the encouragement and support of Dr. A. Zucker and Dr. F. C. Maienschein of the Oak Ridge National Laboratory, and Dr. O. Manley (DOE/BES) in making this effort a reality. We are grateful for the technical review by Dr. C. Jorgensen and Dr. M. Skorska. The manuscript was expertly prepared by R. D. Lawson and A. L. Houston, and reviewed and edited by E. S. Howe.

REFERENCES

1. "PRA Procedures Guide, A Guide to the Performance of Probabilistic Risk Assessments for Nuclear Power Plants," NUREG/CR-2300, Vols. 1 and 2 (January 1983).
2. W. E. Vesely, et al., "Fault Tree Handbook," NUREG-0492 (January 1981).
3. P. E. Hart et al., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on SSC: SSC4, 100-107 (1968).
4. G. Vesonder et al., "ACE: An Expert System for Telephone Cable Maintenance," Eighth Joint Conference on Artificial Intelligence, Aug. 8-12, 1983, Karlsruhe, Germany, p. 116.
5. D. V. McDermott, "R1: The Formative Years," AI Magazine, 2, 21-29 (1981).
6. J. Bennett and C. Hollander, "DART: An Expert System for Computer Fault Diagnosis," Seventh International Joint Conference on Artificial Intelligence, Aug. 24-18, 1981, p. 843.
7. T. Williams et al., "Diagnosis of Multiple Faults in a Nationwide Communications Network," Eighth Joint Conference on Artificial Intelligence, Aug. 8-12, 1983, Karlsruhe, Germany, p. 179.
8. R. Cantone et al., "Model-Based Probabilistic Reasoning for Electronics Troubleshooting," Eighth Joint Conference on Artificial Intelligence, Aug. 8-12, 1983, Karlsruhe, Germany, p. 207.
9. R. Davis, "Diagnosis Via Causal Reasoning: Paths of Interaction and the Locality Principle," National Conference on Artificial Intelligence, Aug. 22-26, 1983, p. 88.
10. W. Nelson, "Reactor: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents," National Conference on Artificial Intelligence, Aug. 18-20, 1982, p. 296.
11. P. Winston, "Artificial Intelligence," Second Edition, Chapter 4, Addison-Wesley Publishing Co. (1984).
12. W. E. Vesely and R. Narum, "Prep and Kitt Computer Codes for the Automatic Evaluation of a Fault Tree," Idaho Nuclear Corporation, Idaho Falls, Idaho, IN-1349 (1970).
13. W. J. Van Slyke and D. E. Griffin, "ALLCUTS, A Fast, Comprehensive Fault Tree Analysis Code," Atlantic Richfield Hanford Co., Richland, WA, ARH-ST-112 (July 1975).
14. R. C. Erdmann et al., "WAMCUT, A Computer Code for Fault Tree Evaluation," Science Applications, Inc., EPRI-NP-803 (June 1978).
15. R. E. Fikes et al., "Learning and Executing Generalized Robot Plans," AI3, 251 (1972).
16. E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," AI5, 115 (1974).

DISCLAIMER