Technical Memorandum 33-567

A Survey of and an Introduction to Fault Diagnosis Algorithms

F. P. Mathur
This report surveys the field of fault diagnosis and introduces the reader to some of the key algorithms and heuristics currently in use. Fault diagnosis is an important and a rapidly growing discipline. This is important to the Jet Propulsion Laboratory's research efforts in the design of self-repairable computers because the present diagnosis resolution of its fault-tolerant computer is limited to a functional unit or processor. Better resolution is necessary before failed units can become partially reusable. The approach that holds the greatest promise is that of resident microdiagnostics; however, that presupposes a microprogrammable architecture for the computer being self-diagnosed. The presentation here is tutorial and contains examples. An extensive bibliography of some 220 entries is included.
HOW TO FILL OUT THE TECHNICAL REPORT STANDARD TITLE PAGE

Make items 1, 4, 5, 9, 12, and 13 agree with the corresponding information on the report cover. Use all capital letters for title (item 4). Leave items 2, 6, and 14 blank. Complete the remaining items as follows:

3. Recipient's Catalog No. Reserved for use by report recipients.

7. Author(s). Include corresponding information from the report cover. In addition, list the affiliation of an author if it differs from that of the performing organization.

8. Performing Organization Report No. Insert if performing organization wishes to assign this number.

10. Work Unit No. Use the agency-wide code (for example, 923-50-10-06-72), which uniquely identifies the work unit under which the work was authorized. Non-NASA performing organizations will leave this blank.

11. Insert the number of the contract or grant under which the report was prepared.

15. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with... Translation of (or by)... Presented at conference of... To be published in...

16. Abstract. Include a brief (not to exceed 200 words) factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified. If the report contains a significant bibliography or literature survey, mention it here.

17. Key Words. Insert terms or short phrases selected by the author that identify the principal subjects covered in the report, and that are sufficiently specific and precise to be used for cataloging.

18. Distribution Statement. Enter one of the authorized statements used to denote releasability to the public or a limitation on dissemination for reasons other than security of defense information. Authorized statements are "Unclassified-Unlimited," "U. S. Government and Contractors only," "U. S. Government Agencies only," and "NASA and NASA Contractors only."


20. Security Classification (of this page). NOTE: Because this page may be used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, indicate separately the classification of the title and the abstract by following these items with either "(U)" for unclassified, or "(C)" or "(S)" as applicable for classified items.

21. No. of Pages. Insert the number of pages.

A Survey of and an Introduction to Fault Diagnosis Algorithms

F. P. Mathur
The work described in this report was performed by the Astrionics Division of the Jet Propulsion Laboratory.
CONTENTS

I. Introduction .................................................. 1

II. Gedanken Experiments — Sequential Machines .................. 2

III. Machine and Fault Identification Experiments ................ 3

IV. Diagnosing and Homing Experiments .......................... 3

V. Equivalent Normal Form (ENF) .................................. 6
   A. ENF Method ............................................... 6
   B. ENF Theorem ............................................. 7

VI. Boolean Difference Algorithms .................................. 8
   A. Basic Concept ............................................ 8
   B. Comments on the Boolean Difference ....................... 10

VII. D-Algorithm .................................................. 10
    A. D-Algorithm Definitions ................................. 11
    B. Algorithm to Form Primitive D-cubes From
       Singular Covers ....................................... 12
    C. Algorithm to Form Primitive D-cubes of Failure ....... 12
    D. Circuit-Labeling Convention ............................ 14
    E. D-Algorithm (DALG-II) ................................ 14

VIII. Microdiagnostics ............................................ 15

Bibliography ..................................................... 17

TABLES

1. Minimal diagnosing sequences for state pairs in M .......... 33

2. A summary of some basic properties of Boolean difference ....... 34

FIGURES

1. Classification of identification experiments ................ 35

2. Machine M transition diagram ................................ 35

3. $P_k$ tables ................................................. 36

JPL Technical Memorandum 33-567
CONTENTS (contd)

FIGURES (contd)

4. Example of ENF of a combinational circuit
   with single output .................................. 37
5. Two-level AND-OR circuit ................................ 37
6. An example of Boolean difference .......................... 37
7. Singular covers and primitive D-cubes of
   some common logic blocks .............................. 38
8. Example of D-algorithm to construct test
   for "line 7 s-a-0" .................................. 38
ABSTRACT

This report surveys the field of fault diagnosis and introduces the reader to some of the key algorithms and heuristics currently in use. Fault diagnosis is an important and a rapidly growing discipline. This is important to the Jet Propulsion Laboratory's research efforts in the design of self-repairable computers because the present diagnosis resolution of its fault-tolerant computer is limited to a functional unit or processor. Better resolution is necessary before failed units can become partially reusable. The approach that holds the greatest promise is that of resident microdiagnostics; however, that presupposes a microprogrammable architecture for the computer being self-diagnosed. The presentation here is tutorial and contains examples. An extensive bibliography of some 220 entries is included.
I. INTRODUCTION

The general field of system testing is an all-encompassing discipline ranging from fabrication processes, parts testing, and production techniques to the final system checkout. This report deals with the area of fault diagnosis as applied to digital computers and systems.

The term fault diagnosis includes both fault detection and fault location, where fault detection is the process of determining whether or not the system is fault-free, whereas fault location is the process of localizing the fault to specific components, sets of components, modules, or subsystems, depending on the diagnostic resolution desired.

Early methods of fault diagnosis in digital computers consisted of diagnostic programs designed to periodically check the major functions of the computer. Often this meant, at best, the testing of the instruction set of the machine using representative sets of data and known corresponding results.

The most commonly used fault model to represent system failures is the "line stuck at 1, line stuck at 0" model. This model assumes that failures are such that any line may become permanently stuck at logical 1 or logical 0 level. This is denoted by s-a-1 or s-a-0, respectively. This model thus includes "lines open" type of failures and also the "shorted lines" type of failures that are such as not to induce feedback as a result of the short.

In general, the problem of fault diagnosis may be succinctly stated thus: By applying signals to the inputs and observing the responses at the outputs, it is determined whether or not the network is operating properly; if not, it must be determined what changes from the norm have occurred.
Exhaustive methods very quickly become unmanageable. If a network consists of \( n \) edges, where each edge may be either fault-free, s-a-0, or s-a-1, then the total number of possible "stuck-at" faults amounts to \( 3^n - 1 \) or \( 3^n \) faults, including the null fault. If only single faults are considered, then the total number of faults including the null fault is \( 2n + 1 \). As an illustration, a NAND gate realization of an EXCLUSIVE-OR circuit consists of 9 edges; thus, the total number of possible faults that may arise are \( 3^9 = 19,663 \). However, if the single fault assumption is made, the number of faults reduces to \( 2 \times 3 + 1 = 7 \).

In order to make the problem of fault diagnosis more manageable, various simplifying assumptions are made, and algorithms are developed which are applicable to specific classes of digital networks or are more efficient under specific constraints than under others. Typical subclassifications of digital networks indicate whether the network is combinatorial or sequential, and in the latter case whether it is synchronous or asynchronous; whether the network has single or multiple outputs; whether single or multiple faults are considered; whether there is fan-out or no fan-out, and if there is fan-out, whether it is reconvergent or non-reconvergent; whether or not the network is redundant; and, finally, whether the particular technique developed is heuristic or algorithmic.

This report describes Gedanken experiments of faulty machine identification, diagnosing and homing experiments, Armstrong's equivalent normal form test generation method, the Boolean Difference techniques, Roth's D-Algorithms, and the newly evolving branch of microdiagnostics. Finally, an extensive bibliography of fault diagnosis has been compiled.

II. GEDANKEN EXPERIMENTS — SEQUENTIAL MACHINES

The material presented here is based on the work of E. F. Moore entitled "Gedanken — Experiments on Sequential Machines" and also that of A. Gill. In Gedanken experiments the finite-state sequential machine is considered to be a "black box" with inputs to which stimuli can be applied and outputs on which the responses may be observed. Only in this way can the behavior, characterization, and contents of the black box be determined.
III. MACHINE AND FAULT IDENTIFICATION EXPERIMENTS

In the problem of identifying the fault of a machine, the faulty version of the machine is simply regarded as another machine. Thus, the set of all possible failed machines under the class of known faults and the fault-free machine constitutes a class of machines. The experiment is to identify the particular unknown failed machine from this set of machines. A subproblem to the machine identification problem is that of state identification. In the latter case the complete state transition table of the finite-state sequential machine is known, and the problem is to determine its initial state, or, in the case of the homing experiment, to be able to drive it into a known state. In solving problems of fault identification both the machine identification and the state identification techniques are used.

The procedure is as follows. From the known machine $M$ which may malfunction in a set of known ways, the possible faulty machines $M_1$, $M_2$, $\cdots$ are characterized by, e.g., specifying their transition tables. These faulty machines are then reduced to their minimal form and constitute an exclusive class of machines $\{M_1, M_2, \cdots, M_n\}$ such that no state in $M_i$ is equivalent to any state in $M_j$ ($j \neq i$). The disjunction machine $\Delta M$ of $M_1$, $M_2$, $\cdots$, $M_n$ is then formed. The fault identification problem in the original machine $M$ is now reduced to that of identifying the final state, i.e., conducting a homing experiment on the machine $\Delta M$.

IV. DIAGNOSING AND HOMING EXPERIMENTS

Figure 1 illustrates the theory underlying diagnosing and the allied problem of homing. The pertinent terms are ordered from the simple and obvious to the more complex and unobvious and are defined as follows:

**Experiment.** The process of applying input sequences to the input terminals of a machine, observing the resulting output sequences, and drawing conclusions based on these observations.

**Preset Experiment.** An experiment in which the applied input sequence is completely determined in advance.
**Adaptive Experiment.** An experiment in which the applied input sequence is composed of two or more subsequences, each subsequence (except the first) being determined on the basis of the previous response.

**Copy.** A machine is a copy of another machine if both have identical transition tables, and if both are at the same state before experiments commence.

**Simple Experiments.** Experiments in which only one copy of the machine is required.

**Multiple Experiments.** Experiments in which more than one copy of the machine is required.

**Length of an Experiment.** The total number of input symbols applied in the course of the experiment.

**Order of an Experiment.** The number of input sequences of which the experiment is composed.

**Multiplicity of an Experiment.** The number of copies required of the machine under investigation. (Note: a simple experiment is an experiment of multiplicity = 1; a multiple experiment is an experiment of multiplicity ≥ 2).

**Diagnosing Problem.** Machine M in one of the states \( \{s_1, \ldots, s_m\} \), transition table known, to find this state.

**Homing Problem.** Machine M in one of the states \( \{s_1, \ldots, s_m\} \), transition table known, to pass M into a known state.

**Diagnosing Experiment.** An experiment that solves the diagnosing problem.

**Homing Experiment.** An experiment that solves the homing problem. (Note: every diagnosing experiment is also a homing experiment; the converse is not true.)

**Admissible Set A(M).** Of machine M, the set of states \( \{s_1, \ldots, s_m\} \) one of which is the initial state of M.

**Admissible States.** The states in the admissible set A(M).

**m-Wise Diagnosing Problem.** The problem of identifying the initial state of M where A(M) is arbitrarily m.
Pair-Wise Diagnosing Problem. The problem of identifying the initial state of M where \( m = 2 \).

Diagnosing Sequence. For \( \{ s_{i0}, s_{j0} \} \) is an input sequence of length \( n - 1 \) or less which, when applied to M in \( s_{i0} \) and M in \( s_{j0} \), yields distinct output sequences.

Minimal Diagnosing Sequence. \( E(s_{i0}, s_{j0}) \) for the pair \( \{ s_{i0}, s_{j0} \} \) in a diagnosing sequence of length \( \ell \) where \( s_{i0} \) are \( \ell \) distinguishable and \( (\ell - 1) \) equivalent \( (1 \leq \ell \leq n - 1) \).

An Example

To find the minimal diagnosing sequence \( E(1,2) \) for the pair of states \( \{1,2\} \) of the machine M whose state diagram is as shown in Fig. 2. First construct the \( P_k \) tables. \( P_k \) is the partition of \( S \) according to the \( k \)-equivalence of states in \( S \). By observation of Fig. 2, the state transition table shown in Fig. 3a is derived. We apply the following algorithm:

Algorithm

To determine minimal diagnosing sequences for state pairs.

1. Construct \( P_k \) tables.
   Find \( \ell \) such that \( s_{i0}, s_{j0} \) are adjoint rows in \( P_{\ell - 1} \) and disjoint in \( P_\ell \). Let \( k = 1 \).

2. If \( (\ell - k) > 0 \) go to step 3.
   If \( (\ell - k) = 0 \) \( E_k \) is given the heading of any column in the \( Z_v \) subtable of M such that rows \( s_{ik-1}, s_{jk-1} \) are distinct.
   \( E_{u1}, E_{u2}, \ldots, E_{uk} \) is the minimal diagnosing sequence for \( \{ s_{i0}, s_{j0} \} \).

3. \( E_{uk} \) is the heading of any column in \( P_{\ell - k} \) such that rows \( s_{ik-1}, s_{jk-1} \) in this column have differently subscripted entries.
   These entries are \( s_{ik}, s_{jk} \).
   Increment \( k \) by \( 1 \) and return to step 2.

Now, having already constructed the \( P_k \) tables, we start from the \( P_3 \) that is the last table in which rows 1 and 2 are adjoint. Rows 1 and 2 in the \( P_3 \) table have distinct subscripts in entries 4c and 5d which appear in column \( \beta \). \( \beta \), then, is the first symbol in \( E(1,2) \). In the \( P_2 \) table, rows 4 and 5 have
distinct subscripts in entries 3b and 2a which appear in column $\alpha$. $\alpha$, then, is the second symbol in $E(1, 2)$. In the $P_1$ table, rows 3 and 2 have distinct subscripts in entries 5b and 1a, which appear in column $\alpha$. $\alpha$, then, is the third symbol in $E(1, 2)$. Alternatively, $\beta$ could have been chosen as the third symbol, since rows 3 and 2 have distinct subscripts in entries 1a and 5b, which appear in column $\beta$. In the $Z_v$ subtable, rows 1 and 5 have distinct entries (0 and 1) in column $\alpha$. $\alpha$, then, is the fourth and last symbol in $E(1, 2)$. Thus, $(1, 2)$ is either $\beta a\alpha a$ or $\beta a\beta a$. From the $P_0$ table or Fig. 2 it can be readily verified that when $\beta a\alpha a$ is applied to $M$ at state 1 and state 2 the last output symbol is 1 and 0, respectively. Consequently, if $\{1, 2\}$ is the admissible set of $M$, the distinguishing experiment may be conducted by applying $\beta a\alpha a$ and observing the last output symbol. If this symbol is 1, the initial state is 1; if this symbol is 0, the initial state is 2.

The minimal diagnosing sequences for all pairs of states $\{s_{10}, s_{j0}\}$ in $M$ are listed in Table 1. The last two columns in this table indicate the last output symbols $Z_i$ and $Z_j$ observed when the minimal diagnosing sequence is applied to $s_{10}$ and $s_{j0}$, respectively.

V. EQUIVALENT NORMAL FORM (ENF)

The ENF method is that of Armstrong (1966). The equivalent normal form is an equivalent two-level circuit representation that corresponds to a multi-level circuit. It is a test generation method well suited to the selection of efficient (near minimal) test sets. It is easily programmable on a computer; however, it becomes unmanageable for large circuits. The ENF method has not been proved to be algorithmic; however, a counter example has not been discovered.

A. ENF Method

To construct ENF of a combinational circuit with single output:

1. Write the Boolean expression for the circuit. Corresponding to each gate $G_j$ of the circuit, there shall be a pair of parentheses in the expression, the parentheses being labeled with the subscript $j$. The subexpression corresponding to $G_j$ shall be en-

JPL Technical Memorandum 33-567
closed in parentheses. All literals in the Boolean expression will denote input variables.

(2) Expand to sum-of-products normal form. When a pair of parentheses is removed, its associated label is attached to each literal inside the parentheses as a subscript. No redundant terms are removed.

An example is shown in Fig. 4, where

\[
f = \left[ (A \cdot B)_1 + (B \cdot C)_2 \right]_3 \cdot D
\]

\[
= \left[ (A_1 \cdot B_1 + B_2 \cdot C_2)_3 \right]_4 \cdot D
\]

\[
= (A_1 \cdot B_{13} + B_{23} \cdot C_{23}) \cdot D
\]

\[
= (A_1 \cdot B_{13} + B_{23} \cdot C_{23}) \cdot D
\]

\[
= (A_{134} + B_{134}) \cdot (B_{234} + C_{234}) \cdot D
\]

\[
= A_1^1 \cdot B_{234} + A_{134} \cdot C_{234} + B_{134} \cdot B_{234} + B_{134} \cdot C_{234} + D_4
\]

\[
= A_{\alpha}^1 \cdot B_{\beta}^1 + A_{\alpha} \cdot C_{\beta}^1 \cdot B_{\beta}^1 + B_{\alpha} \cdot C_{\beta}^1 + D_Y
\]

which is the ENF sum-of-products form where \( \alpha, \beta, \) and \( \gamma \) are path from inputs to the circuit output. The ENF form is realized by the two-level AND-OR circuit shown in Fig. 5.

B. ENF Theorem

A test for a literal appearance in the ENF sensitizes the corresponding path in the circuit. Thus, if a set of literal appearances can be selected whose corresponding paths together contain every vertex of the circuit, and if a set of tests can be found which tests at least one appearance of each literal for the S-a-1 and S-a-0 faults, then the set of tests detects every fault of the circuit.

For example, in the above the test for a S-a-1 fault for the literal \( A_{\alpha}^1 \) is \( A_1^1 = 0, B_1^1 = 0, C_{\beta}^1 = 1 \) and \( D_4^1 = 0 \). From the theorem this input vector \((0,0,1,0)\) sensitizes the path \( \alpha \); i.e., G1G3G4 beginning at A.
VI. BOOLEAN DIFFERENCE ALGORITHMS

The Boolean difference method first applied to fault-diagnosis by Sellers et al. (1967) has been described by Akers (1959) and also by Amar and Condulmari (1967). Unlike the tabular or exhaustive search methods described earlier, Boolean difference approach is an elegant systematic equation-solving procedure. Algorithms have been implemented which can handle combinational networks with up to fifty primary input and output variables. Boolean difference techniques have also been extended to generate test patterns for asynchronous sequential circuits.

A. Basic Concept

If $f(x_1, \ldots, x_n)$ is a switching function of the $n$ variables $x_1, \ldots, x_n$ then the Boolean difference is defined as:

$$\Delta f(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, \overline{x_i}, \ldots, x_n)$$

which is also equivalent to

$$f(x_1, \ldots, 1, \ldots, x_n) \oplus f(x_1, \ldots, 0, \ldots, x_n)$$

This implies that Boolean difference $df(X)/dx_i$ is a function of $(n - 1)$ variables $x_1', \ldots, x_{i-1}', x_{i+1}', \ldots, x_n'$.

From the definition of EXCLUSIVE-OR function it follows that when $df(X)/dx_i = 1$

$$f(x_1, \ldots, 1, \ldots, x_n) \neq f(x_1, \ldots, 0, \ldots, x_n)$$

and when $df(X)/dx_i = 0$

$$f(x_1, \ldots, 1, \ldots, x_n) = f(x_1, \ldots, 0, \ldots, x_n)$$

Thus $df(X)/dx_i = 1$ is a condition for $f(X)$ being dependent on the value of $x_i$. Which implies that a change at $x_i$ always causes a change at $f(X)$
when \( \frac{df(X)}{dx_i} = 1 \), and \( x_i \) does not cause a change at \( f(X) \) when \( \frac{df(X)}{dx_i} = 0 \). Hence \( \frac{df(X)}{dx_i} = 1 \) is a sufficient condition for the path from \( x_i \) to \( F \) to be sensitized.

In general, \( \frac{df(X)}{dx_i} \) is neither 1 nor 0 but some Boolean function not containing \( x_i \); hence, input test patterns may be selected in order to satisfy the condition \( \frac{df(X)}{dx_i} = 1 \). Now, since \( \frac{df(X)}{dx_i} \) is independent of \( x_i \) in general, two, i.e., a pair of test patterns, can be selected corresponding to in one case \( x_i = 1 \) and in the other \( x_i = 0 \).

The Boolean difference approach may be used to detect inversion-type faults (1 \( \rightarrow \) 0, 0 \( \rightarrow \) 1) in addition to the conventional "stuck-at" faults.

An example of Boolean difference is shown in Fig. 6. We seek to find the conditions under which an error in \( x_1 \) causes an error at the output.

**Solution**

\[
\frac{df(X)}{dx_1} = \frac{d(x_1x_2 + x_3 + x_4)}{dx_1}
\]

using the property

\[
\frac{d(A + B)}{dx_1} = \overline{A} \cdot \frac{dB}{dx_1} \oplus B \cdot \frac{dA}{dx_1} \oplus \frac{dA}{dx_1} \cdot \frac{dB}{dx_1}
\]

\[
\frac{df(X)}{dx_1} = (x_3 + x_4) \cdot \frac{d(x_1x_2)}{dx_1}
\]

Now, using the property

\[
\frac{d(A \cdot B)}{dx_1} = A \cdot \frac{dB}{dx_1} \oplus B \cdot \frac{dA}{dx_1} \oplus \frac{dA}{dx_1} \cdot \frac{dB}{dx_1}
\]

\[
\frac{df(X)}{dx_1} = (x_3 + x_4) \cdot x_2 \cdot \frac{dx_1}{dx_1}
\]

which, from the property \( \frac{dA}{dx_1} = 1 \) if \( A \) depends only on \( x_1 \)

\[
\frac{df(X)}{dx_1} = (x_3 + x_4) \cdot x_2
\]

\[
= \overline{x_3} \cdot \overline{x_4} \cdot x_2
\]

\[
\overline{x_3} \cdot \overline{x_4} \cdot x_2 = 1 \text{ when } x_2 = 1, x_3 = 0 \text{ and } x_4 = 0.
\]
This is the condition when an error in $x_1$ causes an erroneous output.

The input test patterns for detecting failure at $x_1$ are $(x_1, x_2, x_3, x_4) = (0, 1, 0, 0)$ and $(1, 1, 0, 0)$.

B. Comments on the Boolean Difference

The astute reader will have observed that the Boolean difference concerns itself with only input variables, and the reader may well ask how does one generate test patterns for "internal" failures? The solution is to break the circuit at that point and consider it as an extra input to the circuit. Having obtained test patterns for this extra input one works backward to specify the test patterns, using only the original circuit inputs. Also, it is possible to form a two-level equivalent circuit of any multi-level circuit (see Subsection V, Equivalent Normal Form). Also, partial Boolean difference, Boolean difference chains, and Boolean difference of multiple variables have been defined. Application of Boolean difference to sequential circuit uses a similar approach to that of the D-algorithm of first modeling the circuit using the Moore model and cutting the feedback lines to reduce the model to an equivalent combinational circuit during delta time intervals. A summary of some basic properties of Boolean difference is found in Table 2.

References on the Boolean difference are found in Bibliography entries 2, 3, 10, 11, 44, 65, 91, 92, 93, 94, 134, 135, 152, 190, 191, and 192.

VII. D-ALGORITHM

The D-algorithm is the first method for generating tests for nonredundant combinational circuits that has been proved to be algorithmic; i.e., if a test exists for detecting a failure, then by the application of the D-algorithm one can find this test. The D-algorithm was formulated by J. P. Roth (see Bibliography entries 169, 170, 178, 180) and is precisely and elegantly expressed in terms of the calculus of cubical complexes. The D-algorithm is a systematic application and extension on the basic concept of path sensitization and is a logical culmination of Eldred's pioneering work. Besides algorithms for generating tests for specific faults Roth's program TESTDETECT solves the related problem of enumerating the set of all faults.
that a particular test can detect. Finally the D-algorithm has been extended to apply to asynchronous sequential logic by transforming the problem of finding a single failure in the sequential circuit to that of finding multiple failures in an iterative combinational circuit constructed from the sequential model by the cutting of all its feedback lines. The D-algorithms DALG and DALG II, as well as their extensions: the sequential circuit heuristics, known as iterative test generator (ITG) and macroblock test generator (MTG) have been programmed in the APL language.

It should be noted that since the D-algorithm guarantees obtaining tests for non-redundant combinational networks then by implication the D-algorithm may also be used to detect redundancy of design in combinational circuits for which complete tests are not generatable.

A. D-Algorithm Definitions

Singular cover is a rearranged compact form representation of a truth-table (an x is used to denote that the corresponding variable may be a 1 or 0).

Singular cube rows of singular covers are termed singular cubes.

D-cubes represent the input-output behavior of the failing as well as the good circuit.

Primitive D-cubes of a logic block (element) is a subset of D-cubes defined as those D-cubes having a D or D on the output coordinate.

Primitive D-cube of failure is the particular primitive D-cube of a failing block. These are determined by both the logic function and the assumed failure of the block.

D-drive is the operation of constructing a D-path or set of D-paths (sensitized path) from the failed block to a primary output.

D-frontier consists of all the blocks in the circuit having D's or D's on some of their inputs and X's on their outputs.

Consistency operation is the justification of all 0, 1 values imposed during D-drive in terms of primary-input values.
**D-signals** are 5-valued logic \(0, 1, X, D, \overline{D}\) used to describe the behavior of a circuit with failures. The symbol \(D (\overline{D})\) denotes a logical value which is 1 (0) in the good circuit and 0 (1) in the circuit with failure.

B. **Algorithm to Form Primitive D-cubes From Singular Covers**

Step 1. Intersect cubes (rows) of the singular cover, these cubes must have different output values.

Step 2. The intersection rules of the input values are:

\[
\begin{align*}
1 \cap 0 &= \overline{D} \\
0 \cap 1 &= D \\
X \cap X &= X \\
0 \cap 0 &= 0 & X = X \cap 0 &= 0 \\
1 \cap 1 &= 1 & X = X \cap 1 &= 1
\end{align*}
\]

C. **Algorithm to Form Primitive D-cubes of Failure**

Step 1. Use cubes of the singular cover.

Step 2. Intersect pairs of cubes.

Step 3. Use same intersection rules as the algorithm to form primitive D-cubes.

Step 4. Select one cube of each pair from the singular cover, the other member is the corresponding cube from the singular cover of the failed gate.

Step 5. Assign bars by the following convention: A vertex of the intersection have value \(\overline{D}\) if the vertex has value 0 in the cube of the failure-free block, and value 1 in the cube of the failed block. All other cases are assigned values D.
Example

Given the truth table for the INHIBIT-NOT logical block:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(1) To determine its singular cover:

<table>
<thead>
<tr>
<th>Singular cover</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

(2) To determine the primitive D-cubes:

<table>
<thead>
<tr>
<th>Primitive D-cube</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>its dual</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>1</td>
<td>D</td>
<td></td>
<td>D</td>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>D</td>
<td>D</td>
<td></td>
<td>0</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>D</td>
<td>D</td>
<td></td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

This is determined from the 0-cube 0 1 0 in the singular cover, which is used as the generator. Generators have the property that a change in a single input must force a change in the output. A change in either input or block inputs causes a change in the output.

A change in input 1:

\[
\begin{array}{c}
1 \\
\oplus \\
1 \\
\cdot \\
1 \\
X \\
1 \\
= \\
1 \\
\downarrow \\
D \\
1 \\
D
\end{array}
\]
A change in input 2:

\[
\begin{array}{c}
0 \ 0 \ 1 \ \wedge \ X \ 0 \ 1 = 0 \ 0 \ 1 \\
0 \ \ D \ \ \bar{D}
\end{array}
\]

A change in inputs 1 and 2:

\[
\begin{array}{c}
1 \ 0 \ 1 \ \wedge \ X \ 0 \ 1 = 1 \ 0 \ 1 \\
D \ \ \bar{D} \ \ D
\end{array}
\]

D. Circuit-Labeling Convention

1. Assign an integer label to each vertex of the circuit.

2. Designate gates (blocks) by the label of the vertex corresponding to its output. Note: each vertex corresponds to either a primary input signal or a gate output signal.

3. Assign vertex number by using the "leveling rule" which is that the integer associated with a block shall be greater than the integers of all vertices which feed it.

E. D-Algorithm (DALG-II)

Start by applying a primitive D-cube of failure at the failing block (logic element), thus obtaining a D or \( \bar{D} \) at its output.

(The goal is to construct a D-path (sensitized path) or set of D-paths, from the failed block to a primary output, an operation called D-drive.)

Initially the D-frontier (defined to consist of all the blocks in the circuit having D's or \( \bar{D} \)'s on some of their inputs and X's on their outputs) consist of the successors of the failed block. The D-frontier is then moved toward the primary outputs by applying suitable primitive D-cubes to the blocks in the D-frontier thus advancing the D-frontier itself, until a D or \( \bar{D} \) has been imposed on a primary output. At this point, a connected D-path has been established from the failed block to the primary output, and the D-drive is terminated.
Now, all 0,1 values imposed during D-drive have to be justified in terms of the primary-input values. This is accomplished by the consistency operation; i.e., iteratively applying suitable singular cubes to all blocks having output values not justified by their inputs, if this is possible, until all 0,1 signals have been driven back to primary inputs. An example is shown in Fig. 8.

To construct a test for "line 7 s-a-0," first the primitive D-cube of failure 1 1 D is used to impose values 1 1 D on lines 1, 2 and 7. The primitive D-cube D 0 D is then applied, thus driving a D through block 9. As a consequence, a 0 must be imposed on line 3. The D on line 9 is now driven to the primary-output 12 by applying to lines 9, 10 and 12 the primitive D-cube D 0 D. This completes the D-drive. The consistency operation now amounts to justifying the 0 value on line 10. This is done by imposing a 0 on line 4. The obtained test for "7 s-a-0" is 1100xx.

References on D-algorithms are found in Bibliography entries 18, 19, 20, 39, 43, 65, 67, 154, 161, 169, 170, 171, 173, 174, 175, 176, 177, 178, 180, and 186.

VIII. MICRODIAGNOSTICS

Microprogram routines stored in read-only memories, in contrast to the traditional control unit (CPU) of the classical computer, act as the master control governing at the gate level the data flow of the machine.

The traditional machine language diagnostics consisting of function tests, measurements tests, utility programs, and diagnostic monitors apply at the level of machine instructions and data words. However, microdiagnostics or diagnostic microprograms have a much finer resolution in that the data paths are controlled at the gate level thus enabling the application of diagnostic tests at the circuit level. Another advantage is that the amount of hard core (or unaccessible circuitry) is drastically reduced.

An example of the application of microdiagnostics is the IBM System 360 model 85. Its diagnostic routines are written in a microinstruction language and are executed out of a read-and-write form of control storage.
They claim that 85% of all failures in the model 30 CPU are fault-locatable to four or less SLT cards.

In the author's view, the approach that holds the greatest promise for the diagnosis of closed (inaccessible to human intervention) computer system is that of resident microdiagnostic; however, that presupposes a microprogrammable architecture of the computer being self-diagnosed.

References on microdiagnostics are found in Bibliography entries 8, 31, 79, and 96.
BIBLIOGRAPHY


BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


JPL Technical Memorandum 33-567


BIBLIOGRAPHY (contd)


JPL Technical Memorandum 33-567


JPL Technical Memorandum 33-567
BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


JPL Technical Memorandum 33-567
BIBLIOGRAPHY (contd)


BIBLIOGRAPHY (contd)


Table 1. Minimal diagnosing sequences for state pairs in M

<table>
<thead>
<tr>
<th>(s_{i0})</th>
<th>(s_{j0})</th>
<th>(E(s_{i0}, s_{j0}))</th>
<th>(Z_i)</th>
<th>(Z_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>(\beta\alpha\alpha\alpha)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(\alpha\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>(\alpha\alpha\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>(\alpha)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>(\alpha\alpha\alpha)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2. A summary of some basic properties of Boolean difference

\[
\begin{align*}
\frac{df(X)}{dx_i} & \triangleq f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \overline{x_i}, \ldots, x_n) \\
\frac{df(X)}{dx_i} & = f(x_1, \ldots, 1, \ldots, x_n) \oplus f(x_1, \ldots, 0, \ldots, x_n) \\
\frac{df(X)}{dx_i} & = 0 \text{ if } f(X) \text{ is independent of } x_i \\
\frac{df(X)}{dx_i} & = 1 \text{ if } f(X) \text{ depends only on } x_i \\
\frac{df(\overline{X})}{dx_i} & = \frac{df(X)}{dx_i} \\
\frac{df(X)}{dx_i} & = \frac{df(X)}{d\overline{x_i}} \\
\frac{d}{dx_i} \left( \frac{df(X)}{dx_j} \right) & = \frac{d}{dx_j} \left( \frac{df(X)}{dx_i} \right) \\
\frac{df(f(X) \cdot G(X))}{dx_i} & = f(X) \frac{dG(X)}{dx_i} \oplus G(X) \frac{df(X)}{dx_i} \oplus \frac{df(X)}{dx_i} \cdot \frac{dG(X)}{dx_i} \\
\frac{df(f(X) + G(X))}{dx_i} & = \overline{f(X)} \frac{dG(X)}{dx_i} \oplus G(X) \frac{df(X)}{dx_i} \oplus \frac{df(X)}{dx_i} \cdot \frac{dG(X)}{dx_i} \\
\frac{df(f(X) \oplus G(X))}{dx_i} & = \frac{df(X)}{dx_i} \oplus \frac{dG(X)}{dx_i} \\
\frac{df(f(X) \cdot G(X))}{dx_i} & = f(X) \frac{dG(X)}{dx_i} \text{ if } f(X) \text{ is independent of } x_i \\
\frac{df(f(X) + G(X))}{dx_i} & = \overline{f(X)} \frac{dG(X)}{dx_i} \text{ if } f(X) \text{ is independent of } x_i
\end{align*}
\]
Fig. 1. Classification of identification experiments

Fig. 2. Machine M transition diagram
**Fig. 3.** $P_k$ tables

### Table $P_0$

| $S_v$ | $\alpha$ | $\beta$ | $S_{v+1}$ | $\alpha$ | $\beta$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

$P_0$ table determined from Figure 2

### Table $P_1$

| $\Sigma_1$ | $S_v$ | $S_{v+1}$ | $\alpha$ | $\beta$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>1a</td>
<td>4b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1a</td>
<td>3b</td>
<td>1a</td>
</tr>
<tr>
<td>$b$</td>
<td>4</td>
<td>3a</td>
<td>4b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2a</td>
<td>5b</td>
<td></td>
</tr>
</tbody>
</table>

$P_1$ table (determined from the $P_0$ table). The response subtable is not shown again. $\Sigma_1$, the 1-equivalence classes are formed by adjoining those states in the transition table having identical rows in the response subtable: if, states are adjoin in $P_1$ if and only if, for every input symbol, they yield identical output symbols.

### Table $P_2$

| $\Sigma_2$ | $S_v$ | $X_v$ | $S_{v+1}$ | $\alpha$ | $\beta$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>1a</td>
<td>4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1a</td>
<td>5c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>3</td>
<td>5c</td>
<td>1c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>4</td>
<td>3b</td>
<td>4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>5</td>
<td>2a</td>
<td>5c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$P_2$ table (determined from the $P_1$ table). $\Sigma_2$, the two-equivalence classes are formed by adjoining those adjoin states in the $P_1$ table that have identical subscript rows. Disjoint rows in $P_1$ remain disjoint in $P_2$.

### Table $P_3$

| $\Sigma_3$ | $S_v$ | $X_v$ | $S_{v+1}$ | $\alpha$ | $\beta$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>1a</td>
<td>4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1a</td>
<td>4d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>3</td>
<td>5d</td>
<td>1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>4</td>
<td>3b</td>
<td>4c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>5</td>
<td>2a</td>
<td>5d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table $P_4$

| $\Sigma_4$ | $S_v$ | $X_v$ | $S_{v+1}$ | $\alpha$ | $\beta$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>1a</td>
<td>4d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>2</td>
<td>1a</td>
<td>5e</td>
<td>1a</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>3</td>
<td>5e</td>
<td>1a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>4</td>
<td>3c</td>
<td>4d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e$</td>
<td>5</td>
<td>2b</td>
<td>5e</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$P_4$ table cannot be refined further, since the equivalence classes have no adjoin rows with distinguishable subscript rows. The most refined partition $P = P_4$.
Fig. 4. Example of ENF of a combinational circuit with single output

Fig. 5. Two-level AND-OR circuit

Fig. 6. An example of Boolean difference

\[ f(x) = x_1 x_2 + x_3 + x_4 \]
<table>
<thead>
<tr>
<th>Block</th>
<th>Truth table</th>
<th>Singular cover</th>
<th>Primitive D-cubes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>D 1 D</td>
</tr>
<tr>
<td>AND</td>
<td>1 0 0</td>
<td>0 X 0</td>
<td>1 D D</td>
</tr>
<tr>
<td>0 1 0</td>
<td>X 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td></td>
<td>1 1 0</td>
<td>1 1 0</td>
<td>D 1 D</td>
</tr>
<tr>
<td>NAND</td>
<td>1 0 1</td>
<td>0 X 1</td>
<td>1 D D</td>
</tr>
<tr>
<td>0 1 1</td>
<td>X 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td></td>
<td>1 1 0</td>
<td>0 0 1</td>
<td>D 0 D</td>
</tr>
<tr>
<td>NOR</td>
<td>1 0 0</td>
<td>1 X 0</td>
<td>0 D D</td>
</tr>
<tr>
<td>0 1 0</td>
<td>X 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1 2 3</td>
<td>1 2 3</td>
<td>1 2 3</td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td>0 0 0</td>
<td>D 0 D</td>
</tr>
<tr>
<td>OR</td>
<td>1 0 1</td>
<td>1 X 1</td>
<td>0 D D</td>
</tr>
<tr>
<td>0 1 1</td>
<td>X 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 2</td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>0 1</td>
<td>X X</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 7. Singular covers and primitive D-cubes of some common logic blocks

![Diagram](image)

Fig. 8. Example of D-algorithm to construct test for "line 7 s-a-0"