Sequential Diagnosability of Digital Systems

ABSTRACT

F. Karonan, T. Karonan, Department of Electrical Engineering, The Technion - Israel Institute of Technology, Haifa, Israel.

We denote the following function:

\[ t = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i = 0 \\ 0 & \text{otherwise} \end{cases} \]

where \( x_i \) are the outputs of the network and \( n \) is the number of inputs.

### 2. Minimal Sequential Diagnosis

Given a set of \( X \) of possible faults and a network with \( n \) inputs, the minimal sequential diagnosis algorithm proceeds as follows:

1. Compute the number of faults \( t \) that need to be detected, where \( t \) is the number of outputs of the network.
2. For each fault \( f \), determine the sequence of outputs that would be produced by the network if that fault was present.
3. Compare the actual sequence of outputs with the expected sequence for each fault.
4. If the actual sequence matches the expected sequence for a fault, that fault is detected.
5. If all faults are detected, stop. Otherwise, remove the detected faults and repeat from step 1.

The algorithm terminates when all faults are detected, and the sequence of detected faults provides a minimal set of faults that need to be repaired.
(3.1) MINIMAL SENTENTIAL DIAGNOSIS

We estimate now the number of computations required to determine a near-optimal decision tree. Let's assume the "best" tree at this stage is the one which maximizes the weighted function of a test in a decision tree. The test which maximizes the weighted function will be selected according to a set of decision trees. Each decision tree is constructed and all its first vertices, at each vertex of the tree once out of a near-optimal solution is obtained by performing local optimization. The decision tree

\[
T_{d} = (b_{1}, \ldots, b_{m})
\]

where \( b_{i} \) are the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults.

For large networks in which the number of possible faults is greater than \( p_{i} \) and \( p_{j} \),

\[
\sum_{i=1}^{n} \frac{2^{n-i+1} \cdot b_{i}}{2^{n-i} \cdot b_{i}} \geq \frac{2^{n-i} \cdot b_{i}}{2^{n-i+1} \cdot b_{i}}
\]

The size of computer memory required for the \( K \) faults is approximated by the following expression:

\[
G = \frac{2^{n-i+1} \cdot b_{i}}{2^{n-i} \cdot b_{i}}
\]

Given these, hence the estimated number of computations is:

\[
G = \frac{2^{n-i+1} \cdot b_{i}}{2^{n-i} \cdot b_{i}}
\]

To evaluate the effectiveness of this method, we estimate the number of computations and the number of computer programs required for each test. Accordingly, the decision tree is constructed. Notice that using dynamic programming, we do not compute all possible decision trees simultaneously. Instead, we use a dynamic programming approach to find the optimal decision tree. Each \( g_{i} \) failure and two separate for which we have previously found the optimal tree, each decision tree is constructed by checking the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults. The decision tree is constructed by checking the faults which are not detected by the given faults.
which has a unique maximum at the same point. If one of the \# faults at vertex \( a \) is 0,

\[
\begin{align*}
\sum_{d-1}^{\infty} t^{d-1} & = e^t \\
\sum_{d=0}^{\infty} t^d & = \frac{1}{1-t}
\end{align*}
\]

It can therefore be written by the following function:

\[
\frac{1}{(1-d)^m} = \cdots = \frac{1}{(1-d)^1} = \frac{1}{(1-d)^0}
\]

The case \( \sum d \) is a subset of \( \sum d \) which is maximal when:

\[
\sum d = (d_1)^{d_1} \cdots (d_n)^{d_n} = 0
\]

Clearly,

\[
\sum d \frac{1}{(1-d)^0} = \frac{1}{(1-d)^n} = \frac{1}{(1-d)^1}
\]

where \( \sum d \) is the sum of probabilities of the faults included in the r-th

application of a test at vertex \( a \), and \( d \) is the number of subjects.

Uncertainty removed (or equivalently, information gained) by the vertex \( a \) is:

\[
\sum_{d-1}^{\infty} t^{d-1} = \sum_{d=0}^{\infty} t^d - \sum_{d=0}^{\infty} t^d
\]

In the above:\n
\( d \) - 1 subset \( (0, 1, \cdots, n-1) \)

contains all faults for which the vertex \( a \) is not so.

Applying the test to vertex \( a \), it divides the \# faults into at most \( \frac{1}{(1-d)^n} \) subsets.

The uncertainty at vertex \( a \) is:

\[
\sum_{d=0}^{\infty} t^d - \sum_{d=0}^{\infty} t^d
\]

are the set of faults at vertex \( a \), the vertices of the decision treeexcluding faults and faults are equivalent.

We present now the algorithm used to calculate a nearly minimal decision tree using the

second weighting function. This function incorporates the number of faults of faults distinguished by the vertex in the decision tree.

The first weighting function is the distribution function. The second weighting function is the distribution function.

Uncertainty for a nearly minimal solution is significantly smaller than that required for a

solution for a nearly minimal solution. This is because the size of the computer memory

is limited and each test increases the number of faults distinguished by the vertex.

These different weighting functions are used in the literature for generating nearly minimal

and nearly minimal solutions.
A. MODULE LEVEL DIAGNOSTIC PROCEDURE

Example Network

![Diagram of a network with logic gates]

Let \( P_i \) be the a priori probability that a fault will occur in the module \( M_i \) for completeness, we define the fault-free module \( M_0 \) containing the fault \( F \) only. Let \( P_{M_j} \) be the probability that a fault will occur in the module \( M_j \) of both modules \( M_i \) and \( M_j \).

For completeness, we define the fault-free module \( M_0 \) containing the fault \( F \) only. Let \( P_{M_2} \) be the probability that a fault will occur in the module \( M_2 \) of both modules \( M_i \) and \( M_j \) only.

In this network, the fault-free module \( M_0 \) is defined to be the module \( M_0 \) containing the fault \( F \) only.

We denote the fault-free module \( M_0 \) containing the fault \( F \) only.

In this case, we denote a new module \( M_0 \) that will represent a new module \( M_0 \) containing the fault \( F \) only.

Suppose there are such faults, for example, we assume that there are no faults in this network. In most cases a smaller number of tests because only faults at different modules allows to be distinguished.

Module level diagnoses requires in most cases a smaller number of tests because only faults are considered.

The specific fault within the module is sufficient. A module can be an integrated circuit, a network, or even a subsystem. In this network, the fault-free module \( M_0 \) containing the fault \( F \) only.

The network's diagnosis is performed function we construct the decision tree shown in Fig. 2. For example, the decision tree is a graphical representation of the relationship between the possible outcomes and the evidence. The decision tree is a flowchart-like diagram used to determine the outcome of a series of related decisions or events. Each branch of the tree represents a possible decision or event, and the tree is used to help analyze the potential outcomes of each decision or event.
### Fault Table for the Network in Fig. 1

#### Table 1

<table>
<thead>
<tr>
<th>Faults</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
<th>$x_{10}$</th>
<th>$x_{11}$</th>
<th>$x_{12}$</th>
<th>$x_{13}$</th>
<th>$x_{14}$</th>
<th>$x_{15}$</th>
<th>$x_{16}$</th>
<th>$x_{17}$</th>
<th>$x_{18}$</th>
<th>$x_{19}$</th>
<th>$x_{20}$</th>
<th>$x_{21}$</th>
<th>$x_{22}$</th>
<th>$x_{23}$</th>
<th>$x_{24}$</th>
<th>$x_{25}$</th>
<th>$x_{26}$</th>
<th>$x_{27}$</th>
<th>$x_{28}$</th>
<th>$x_{29}$</th>
<th>$x_{30}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{3.1}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: The table represents the fault conditions for various faults in the network.
PICT. 2 - Decision tree for the network in FIG. 1.
not divided into subparts and the test applied distinguishes among modules only.
For each i, hence, is the maximum when faults in the same module are
either 0 or Wi. For each i, hence, the maximum is a

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

The maximum value is a

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

which becomes:

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

Hence the average information gain due to test i:

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

Simple manipulations yield:

\[ I_i = (2, ..., 2, 1_2) \]

is the probability that the rest will produce the vector

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

where

The average uncertainty remaining after application of test i:

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

which satisfies:

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

probability is divided into the following probabilities:

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

We wish to distinguish among faults in different modules; hence the initial uncertainty

\[ \frac{f_i W_i}{(1)^d} \log_{10} \frac{T_{i,0} t_i}{(1)^d} \]

Clearly
The decision tree constructed for this network is shown in Fig. 3. The average number of tests required to locate a faulty module in this decision tree is \( C = 3.94 \).

The decision tree constructed for this network is shown in Fig. 3. The average number of tests required to locate a faulty module in this decision tree is \( C = 3.94 \).

For the pseudo-module containing the equivalent faults of \( M_1 \) and \( M_2 \), let \( W = M_1' \) and \( W' = M_2' \).

Example: The network in Fig. 1 is constructed out of two modules, \( M_1 \) and \( M_2 \).
REFERENCES


ACKNOWLEDGMENT

This research was carried out as a part of a M.S. dissertation under the guidance of Professor Z.A. Kornat, at the Faculty of Electrical Engineering, Technion - Israel Institute of Technology.