A computerized fault tree construction methodology

This item was submitted to Loughborough University’s Institutional Repository by the/an author.


Additional Information:

• This article was published in the journal, Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering [© IMechE] and is also available at: http://journals.pepublishing.com/content/119780

Metadata Record: https://dspace.lboro.ac.uk/2134/3823

Publisher: © IMechE / Professional Engineering Publishing

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
A computerized fault tree construction methodology

J D Andrews and J J Henry
Department of Mathematical Sciences, Loughborough University of Technology, Leicestershire

Abstract: A new approach to automating the fault tree construction process is proposed. The methodology has features which make it applicable to binary state systems and also to process control systems. Previous attempts to model the failure generation in systems have worked well on only a small sub-class of system types and have failed to produce a generally applicable method. Thus in some previous approaches there are desirable features which are worth retaining in a new construction method. The new method is based on the flexibility of the decision table method but incorporates a way of detecting, classifying and analysing control loops, similar to that used with operators in the digraph approach. As well as using operators to deal with control loops a new operator is introduced that deals with two-state circuits. This will mean that when constructing the fault trees, the difficulty of handling repeated events will be eliminated and the size of the tree structures will be significantly reduced. The developed algorithm can produce a tree format appropriate for direct input to an analysis code. The method is demonstrated in this paper by its application to a simplified safety system from the railway industry. This system has been selected as it exhibits features that are typical of two-state circuitry but is also small enough to illustrate the new developments introduced.

Keywords: safety, reliability, fault tree analysis

NOTATION

C current
CONT contacts
DE de-energized
EN energized
F failed
L line
NC no current
RY relay
W working

1 INTRODUCTION

Many different industries are now required to conduct risk assessments of their activities. This involves the identification of potentially hazardous events and an assessment of their consequences and frequency of occurrence. Once the risk associated with any activity has been evaluated its acceptability must then be judged. One of the most popular approaches to determine the frequency of occurrence of hazardous events is provided by fault tree analysis (1). The application of this method results in a tree structured diagram representing how the component failures, human actions and software errors contribute to the cause of a specified system failure mode. By providing data for the basic events the tree can be analysed to yield the minimal cut sets and system failure probability or frequency.

Unfortunately, the development process for fault trees is a very time intensive activity. Fault trees for the WASH 1400 nuclear reactor study (2) took many man years of effort to construct. Once the trees have been developed they can be submitted to one of the many computer codes which perform the quantitative analysis. Also, since there are no rigorous rules which, if applied to a system, guarantee the generation of the correct fault tree, the quality of trees produced manually is very dependent on the experience and abilities of the engineer who has produced them. These two factors, time and quality, have been the main driving forces behind the research into ways in which fault tree construction can be implemented on a computer. To date there have been many attempts to develop such a code. The most successful have been based on digraph approaches (3–5), decision tables (6), expert systems (7) and functional equations (8–12). Despite the amount of work carried out in this area, a commercial package has yet to become available. Deficiencies are reported in the literature for these methods (13).

The majority of work on fault tree synthesis has been
applied to the control systems for chemical processing plant. The nature of such plant, and in particular the process control systems, means that it does not facilitate accurate fault tree development. The major difficulties arise due to the necessity to model continuous process variables for chemical plant and also electrical circuits which are mainly two-state.

This paper explores the possibility of using a hybrid method derived from two existing methods which would be appropriate to model two-state and continuously variable safety systems. The new approach combines the features of the decision table and digraph methods and also incorporates some developments new to this methodology.

It is not possible to describe the full analysis of a complex system in this paper, so in order to demonstrate an application of these methods a simple train detection system from the railway industry has been selected. In this way, the main features and their advantages can be illustrated. The method is, however, applicable to systems from any industry.

2 OVERVIEW OF THE FAULT TREE CONSTRUCTION PROGRAM

The data flow chart showing the overall approach is shown in Fig. 1. The system description is entered as a schematic diagram [such as a piping and instrumentation diagram (P & ID) for process plant] using the facilities in Auto Cad. From the entered drawing a file is created that holds the topology information for the system. This includes information defining how the components are connected together. There is a component decision table library that holds all decision tables; any of these decision tables can be amended or new decision tables added using the generic component editor. A top event, the causes of which are to be traced, is entered into the fault tree construction program. The program produces the fault tree structure which it writes to an output file that can be read by commercial fault tree analysis packages. The current version of the software is written for compatibility with a specific analysis package, but since the basic structure of information is common to most analysis codes other interfaces could be created with minor effort. Using the analysis package, qualitative and quantitative analyses can be performed giving the top event probability or frequency of occurrence. This type of approach using the computer aided design (CAD) interface would facilitate performance of reliability assessments at the design stage, a practice which is not widely performed at present.

3 SYSTEM EXAMPLE

To lessen the risk of collision between trains, the track is divided into sections and two trains are not allowed to occupy the same section of track at any one time. To insure this a train detection system is incorporated on each section, the running rails of each section are electrically isolated from the sections on either side by insulation blocks; such a section is illustrated in Fig. 2. When the track section is unoccupied electric current flows around the circuit and energizes the track relay which causes a green signal allowing a train to enter this section. If a train is in the section, the leading wheels and axle, being made of steel provide a short circuit between the tracks. Therefore the relay is de-energized, causing the signal to display red until the train has left the section. If a train attempts to enter an occupied section, the ‘train stop’ located at the signal causes the train brakes to activate. On automatic trains there is equipment to detect whether track circuit current is flowing or not. If a flow was not detected in the section, the emergency brake would be applied.

In the analysis presented here failures of the insulating blocks between sections were not taken into consideration. This restricted the analysis to enable the major features of the technique to be demonstrated on a relatively simple example. By restricting the analysis from tracing faults and failures of adjacent sections, it was anticipated that the features of the technique would be clearer. The system shown in Fig. 2 is relevant when there is no train in the section and so the signal is green.

When a train is present in the section the first axle is represented by the line labelled 3. It is considered here that no train present will represent the normal case when there will be no current path from line 2 to line 3. If there is no train and the circuit made there is current to the relay which energizes the relay coil, the contacts then make the connection to the green bulb. G1 and G2 are independent power supplies for each circuit.

The two failures of concern for this system are:
1. Failure of the red signal when a train is present.
2. Failure of the green signal when a train is not present.

4 BASIC DECISION TABLE METHOD

For information on the background of the use of decision tables in fault tree construction see reference (6). The main advantage of this method is that no restriction is placed on the number of discrete states used to represent variable deviations. This method requires the construction of a tabular model to represent the functioning/failure of any component. Each component input (connection) from other components in the system is combined with the component state (i.e. working or a failure mode) to give outputs which are passed on to other elements of the system. Therefore to perform a decision table analysis it is necessary to have a model for each component in the system and a system topology which links the component input and output ports to
show how the system is constructed. A topology diagram for the train detection system when a train is present is shown in Fig. 3.

As an example of how to construct a decision table for a component, consider the relay contacts which move position to light the appropriate signal. First the failure modes of the contact have to be decided upon. It is up to the analyst how many modes are included in the
decision table, for this simple example only one failure mode was considered, this being that the contacts fail so that no connection is made. Other failures which could have been considered are, for example, the contacts welding in either of the two circuit positions. The choice of failure modes considered is generally governed by the available failure rate data. Next the influence on the relay contacts operation is considered to identify the inputs. In this example (see CONT in Fig. 3) there are two inputs, these being the current supply from line 12 (L12) and the relay (RY ) state which fixes the contact position selected from the links to the green and red signals. For
Table 1  Decision table for relay contacts

<table>
<thead>
<tr>
<th>IN1</th>
<th>IN2</th>
<th>STATE</th>
<th>OUT1</th>
<th>OUT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>EN</td>
<td>W</td>
<td>C</td>
<td>—</td>
</tr>
<tr>
<td>C</td>
<td>DE</td>
<td>W</td>
<td>—</td>
<td>C</td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td>—</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>DE</td>
<td>NC</td>
<td>—</td>
<td>NC</td>
<td>—</td>
</tr>
<tr>
<td>EN</td>
<td>F</td>
<td>—</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>EN</td>
<td>W</td>
<td>—</td>
<td>NC</td>
<td>—</td>
</tr>
</tbody>
</table>

Each of the output connections, two variable states, namely, current (C) and no current (NC) are of concern. A table (Table 1) is constructed which considers all possible combinations of inputs from line 12 in a column labelled IN1, inputs from the relay in a column labelled IN2, the system state and the effects that these events will have on the outputs to the green and red signals (columns labelled OUT1 and OUT2 respectively). Entries to the table are determined by considering the output events. There is only one situation that can cause current in output 1 (the green signal connection), which is current in line 12 (C IN1), the relay energized (EN IN2) and the contacts working (W). Similarly there is only one cause of current in output 2, which is current in line 12 (C IN1), relay de-energized (DE IN2), and contacts working (W). These conditions are incorporated in the first two lines of the contacts decision table illustrated in Table 1. Turning to causes for no current, NC, in output 1, there are three ways in which this result can occur:

1. NC in IN1 will cause NC in OUT1 and OUT2 regardless of the relay state (IN2) or the state of the component.
2. The relay (IN2) being de-energized will cause NC in OUT1 regardless of IN1 or the component state.
3. If the contacts fail there will be NC in OUT1 or OUT2 regardless of the input states.

The situations are shown in rows 3–5 of the contacts decision table. There are also three ways which produce NC in output 2, two of these have already been described in cases 1 and 3 above. The remaining case is when the relay is energized and the contacts work which gives OUT2 NC regardless of the status of IN1. Incorporating this combination gives the final decision table for the contacts shown in Table 1.

The ‘—’ in the input and state columns indicates the ‘don’t care’ condition, which means that the specified input states will result in the specified output event regardless of the value of the variable. The final fault tree needs only to represent the necessary and sufficient causes of an event. When a decision table features more than a single output event (as in Table 1) then in some circumstances the combination of input events for a table entry will be necessary and sufficient for one output event but not another. To avoid the input conditions being developed in these cases an ‘unspecified’ state ‘—’ is placed in the relevant output column. Therefore the ‘—’ occurs because the output state is undetermined by the input events or will produce a non-minimal set of causes. The decision tables for the other components have been constructed using the same technique.

Table 2  Decision table for signal/bulb

<table>
<thead>
<tr>
<th>IN1</th>
<th>STATE</th>
<th>OUT1</th>
<th>OUT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>W</td>
<td>ON</td>
<td>C</td>
</tr>
<tr>
<td>NC</td>
<td>F</td>
<td>OFF</td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>OFF</td>
<td>NC</td>
<td></td>
</tr>
</tbody>
</table>

5 FAULT TREE CONSTRUCTION

The method of fault tree construction using decision tables will be illustrated for the top event green lamp off (GLOFF) for the train detection system when there is no train present. First the decision table for the component whose output contains the top event variable is found. Then the output column of this table is searched for occurrences of the top event variable in the specified state, in this case OFF. Table 2 presents the signal/bulb decision table, which contains the top event variable. For the bulb, the entry OFF appears twice in OUT1, these are in rows 2 and 3. Since there is more than one cause of this event an OR gate is used at the top event structure in the fault tree to combine these two possibilities as direct causes of the top event. As row 2 only contains the failure state of the component, this is added to the OR gate as a basic event and the next row is dealt with. In row 3 there is only one entry in the input and state columns (NC IN1), if there were two or more entries then these would be entered in the fault tree as inputs to an AND gate. Each of these events are then expanded in turn. By examining the system topology diagram (Fig. 3) it can be seen that IN1 for the green bulb (GB) connects to the contacts (CONT) OUT1. Therefore, the variable and its value (NC IN1) for the bulb is traced directly to its cause (NC OUT1) for the contacts. In the contacts decision table entries for NC in OUT1 are sought. Three matches are found (rows 3–5, Table 1). Each of these events is then added to the fault tree giving the top event structure shown in Fig. 4. Undeveloped events are each expanded in turn until the fault tree is completed. Any working states incorporated on the fault tree at this stage are then transformed in terms of failure events, i.e. NOT gates containing the component failures.

Prior to adding each event from the decision tables to form the input to a gate, an identifier is attached to the gate as in Fig. 4. The identifier is constructed from the component name and, depending on how many matches are found in the decision table for the event, either the row number or output column number is included.
Fig. 4 Fault tree for green bulb off constructed using decision table method

there is more than one match in the decision table then the output column number will be used. This identifier is used to check for inconsistent and repeated events. A repeated event occurs in the fault tree when the system being tackled contains circuits. Inconsistent events are those which cannot cause the events appearing at higher levels in the tree structure.

5.1 The identification of repeated and inconsistent events

Before a gate is added to a branch of the tree, the new identifier is compared character by character with all the other identifiers above it on that branch. If there is a complete match then this is a repeated event, no variables need to be tested. If there is a match apart from
Table 3: Decision table for the relay

<table>
<thead>
<tr>
<th>IN1</th>
<th>STATE</th>
<th>OUT1</th>
<th>OUT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>W</td>
<td>C</td>
<td>EN</td>
</tr>
<tr>
<td>NC</td>
<td>F</td>
<td>NC</td>
<td>DE</td>
</tr>
</tbody>
</table>

the last number, which indicates the row of the relevant decision table, then there may be a repeated or an inconsistent event depending on the variable inputs in each of the rows. For this case the method used for identification will be described with reference to an example. The identifiers that include the output column number are not checked for matches as repeated or inconsistent events will be detected when this is expressed in terms of its rows later in the tree.

Consider the two identifiers RELAY ROW 1 and RELAY ROW 2. The decision table for the relay is shown in Table 3. The relay has two outputs in the system, the first is to line 6 (L6) and the second to the contacts (CONT). In the first instance consider output for both identifiers coming from line 6, i.e. the first output column, then the entries in output column 1 are current (C) and no current (NC) for each of the rows respectively. As no one branch of a fault tree can contain the same component twice with mutually exclusive variable states, in this case C and NC, these two events are inconsistent events. On the other hand, if the output for the second identifier came from the contacts, then the output variables cannot be compared as they are to different components. In this case the input variables to the rows need to be compared as these will be the next variables that are expanded in the tree. Only input columns that do not contain any ‘don’t care’ states are compared and the state column is ignored. So in this case the only variables to be compared are from input column 1, namely current (C) and no current (NC); therefore, as before, this event would be an inconsistent event. If the variables had matched then there would have been a repeated event.

5.2 Rules for dealing with repeated and inconsistent events

1. For a repeated event the branch of the tree where the second occurrence of the event appears is terminated and the next input event of the gate is expanded. The gate whose input event development was terminated is not detected even if it now contains only one input.

2. For an inconsistent event which is an input to an AND gate, the branch of the tree is traversed upwards until an OR gate or the identifier which made the match is encountered (since any AND gate output event cannot now occur). Then the branch is terminated at this point and the next input event of the gate where the termination has taken place is expanded, again the gate is not deleted.

3. An inconsistent event which is input to an OR gate is dealt with in exactly the same manner as that for a repeated event.

The tree structures produced by this method are very large and contain too many unnecessary branches. A modification to the method that yields more concise tree structures and hence lessens the amount of time spent checking for repeated and inconsistent events would be of real benefit. This efficiency can be gained by utilizing a structure similar to the type of operator found in the digraph technique.

6 MODIFIED DECISION TABLE APPROACH

The main weakness in the decision table method in its original form is its inability to detect, classify and analyse control loops and circuits. Therefore modifications have been made to the basic method to consider these situations. The decision tables are modified to incorporate a gain which indicates how deviations in the input variable affect the output state. This gain is rather like that used with the directed edges in the digraph method. The gain takes a sign value which is + or − depending on the relationship between the input and output variables. If, for example, an increase in an input variable causes an increase in the output variable then the associated gain would be positive since there is a direct relationship between these variables. On the other hand, if an increase in an input variable causes a reduction in an output variable then the gain would be negative, i.e. an indirect relationship. The other main change to the format of the decision tables is the inclusion of an EXCLUSIVE state row. This row is incorporated in the decision table so that mutually exclusive component states are identified and hence inconsistent events can be reduced from the tree using Boolean algebra. This row is added at the bottom of the decision table to show all the states in which the component can reside. The first entry in the EXCLUSIVE row is the working state of the component, all subsequent entries are the possible failure states.

This information is used in fault trees which contain basic events representing the component in its working state. The fault tree should contain only failure events and so the component working state is replaced by NOT any of its failed states. The keyword EXCLUSIVE appears in the table to identify the occurrence of this row, see Table 4. In the new decision tables an identifier is also included that indicates if the component is a power supply. This takes the form of a P after the 3 numbers in row 3 of the decision tables. For example if the decision table in Table 4 was a power supply then the format of row 3 would become 2, 2, 1, P.

The new decision table format for the contacts is
shown in Table 4. The row containing NORMAL is used to provide a component identifier for situations when there is more than one component of the same type. For example there might be two different types of valve in the system then a unique identifier for each of the valves would be entered in this row. In the third row of Table 4 the three numbers correspond to the number of inputs in the table, the number of outputs in the table and the number of tables. There may be more than one table for the different variables, e.g. in a valve there may be one to model temperature deviations and one to model flowrate deviations. For the contacts there are 2 inputs, 2 outputs and only one table which traces the variable current. The fourth row is the gain row where the relationship between each of the input and output variables is stored. The order of the stored gains is such that each input is dealt with in turn and its relationship with each of the outputs is added to the row of gains.

6.1 Loop/circuit identification

Before the tree is constructed using the modified decision table method all of the loops and circuits in the system need to be identified, these are identified by using the topology of the system. This will be demonstrated again using the train detection system example.

The first step in the construction of the fault tree for the train detection system for the top event ‘red signal off when there is a train in the section’ is to build up the topology graph of the system. This is achieved by using the decision tables together with the topology diagram (Fig. 3) to link each of the system components as the variable states are traced from the top event back through the system. A topology graph is a directed graph consisting of nodes and edges. The nodes represent specific values of the output variables for the component identified. The edges link the nodes such that those nodes directly below any other nodes contribute to their cause. The topology graph unlike a fault tree does not, however, indicate how these events need to combine. A node on a topology graph represents an input event from a row of a decision table. As each new node is added to the graph, the component identifier, output column number, input column number and variable state being traced are also included. Construction starts with the top event component so the initial component to consider is the red lamp/bulb. From its decision table (Table 2), it can be seen that the top event variable occurs in output column 1. From the table the causes of this output OUT1 = OFF) coded as 01 = NL in Fig. 5 are given by two rows, the bulb fails (row 3, Table 2) or there is no current in input column 1 (row 2, Table 2); the state column of the decision tables is of no interest in the graph construction since it does not trace faults back through the system and so the first of these events is ignored and only the second is traced. The red bulb connects to output column 2 of the contacts (system shown in Table 4. The row containing NORMAL is used to provide a component identifier for situations when there is more than one component of the same type. For example there might be two different types of valve in the system then a unique identifier for each of the valves would be entered in this row. In the third row of Table 4 the three numbers correspond to the number of inputs in the table, the number of outputs in the table and the number of tables. There may be more than one table for the different variables, e.g. in a valve there may be one to model temperature deviations and one to model flowrate deviations. For the contacts there are 2 inputs, 2 outputs and only one table which traces the variable current. The fourth row is the gain row where the relationship between each of the input and output variables is stored. The order of the stored gains is such that each input is dealt with in turn and its relationship with each of the outputs is added to the row of gains.

6.2 Negative feedback loop/circuit detection

The characteristic of the NFB loop is that it has the ability to correct moderate disturbances in the sensed variable. To identify a NFB loop from the topology graph a path must exist that starts and ends with the same variable with a resulting negative product of the gains. If there is a path starting and ending at the same variable with a positive overall gain it identifies either a positive feedback loop (which does not need special treatment since it will be detected on a system start-up)
Fig. 5  Topology graph for red bulb off
or a circuit. A circuit requires other conditions such as power supply to be contained on the loop. For the identification process each of the branches of the topology tree is dealt with, one at a time. Each node on the branch is compared with every other node above it on the branch for a match. To obtain a match three criteria must be met:

(a) the component type and numbers must match;
(b) the output column number must match; and
(c) the output variable must match.

In this example, it can be seen from Fig. 5 that the first match appears on branch 1 with the components generator 2, output column 1 and variable NC. So the path of the first potential NFB loop or circuit is GEN2→CONT→GB→T4→L12→GEN2 (i.e. generator 2→contacts→green bulb→T-connector→line 12→generator 2). Once all the branches have been traversed in this way the resulting loops need to be compared against each other for duplicates. If two loops contain the same components but in one loop current is being traced and in the other no current is traced, then these loops are treated as completely separate loops and are both retained in the list. If any duplicates are found these repetitions can be discarded. In this example there are a possible six NFB loops or circuits.

1 cont(02)→bulbR(02)→t4(01)→line12(01)→gen2(01)
2 gen2(01)→cont(01)→bulbG(01)→t4(01)→line12(01)
3,4 line4(01)→relay(01)→line6(01)→line7(01)→axle(02)→line8(01)→line9(01)→gen1(01)→line1(01)→line2(01)→axle(01)
5,6 line2(01)→axle(02)→line5(01)→line9(01)→gen1(01)→line1(01)

The state of the variable that is traced round each circuit is as follows:

(1) No current (2) No current (3) Current (4) No current (5) Current (6) No current

Once all of the possible NFB loops have been detected these loops have to be classified as either NFB or circuits. This is where the gain contained in the modified decision tables is used. If loop 1 above is considered, the relationships between the appropriate inputs and outputs for all the components on the loop are extracted from the appropriate decision tables. In this case they are all positive. So when the product of the gains is taken, the resultant gain is also positive. This, together with the fact that the loop contains a power source (gen2), indicates that the path between the common components is a circuit.

6.3 NFF loop detection

Whereas feedback loops correct disturbances which currently exist in the system, a NFF (negative feedforward) loop can, in theory, prevent the disturbance progressing through the system. This is achieved by sensing an upstream variable and manipulating a downstream variable. A NFF control loop features two paths in the system structure, a causative path and a corrective path. These features are identified on a topology graph by paths which progress upward from a common node which occurs on two or more branches and the upward paths converge at a second node. In other words there need to be two or more paths between two components in the system.

The rules for the method of detection are:

1. Only deal with branches that have a common end node.
2. For two branches with a common end node traverse up the topology graph to locate the second common node where the branches meet.
3. The resultant gains on each path of the branches are then evaluated. If the resultant gains are of a different sign then the loop is a NFF loop, otherwise the loop can be discarded.

7 FAULT TREE CONSTRUCTION

When control loops or circuits are encountered during fault tree construction ‘operators’ are utilized which develop causes of the failure events being traced by considering the failure of the loop or circuit as a single entity.

The fault tree is constructed in the usual manner, but as each new component is encountered in the tree production a check is made as described in Section 5 to see whether the component lies on a circuit, NFF loop or NFB loop. If the component lies on a NFF loop or a NFB loop, the operators developed by Lambert (14) are applied. If the component lies on a circuit, the new circuit operator is applied.

7.1 Circuit operators

There are at two circuit operators. The first is applicable when the no current situation is traced around the circuit. This first operator is illustrated in Fig. 6 and can be summarized as follows:

1. A failure of any component in the circuit will cause the open circuit condition with no current. Therefore, deal with all components in turn adding their relevant failure states to an OR gate (right-hand branch).
2. When each component is being dealt with, the decision tables are checked to see if there are any external inputs to the circuit at that component. Any entries of NC in the input that traces the path around the circuit can be ignored as this just represents the circuit continuity. If there are, then these external inputs must be examined. If the external input is any
state of the variable current then this external is inconsistent and can be ignored. Otherwise the external input is expanded using the decision tables as with any other variable state in the fault tree (left-hand branch).

When current flowing in a circuit is traced then for this condition all the components in the circuit must be working, so an AND gate appears in the operator, as shown in Fig. 7.

To demonstrate the use of the circuit operator it is necessary to consider the construction of the fault tree for the train detection system with top event RBOFF.

From the topology graph in Fig. 5 and the decision tables for the components there are six circuits. Initially the tree is developed in the normal manner as shown in Fig. 8 until the identifier BULBrow3 is added. The only input to row3 is NC, the topology graph shows that this input is connected to OUT2 of the contacts. So the next component to be dealt with is Cont(02) NC. The six circuits are searched for this combination and a match is found in circuit 1. As this is the first circuit the only check that needs to be made is if the circuit contains a power source, which it has in component gen2. A note is made of the entry component to the circuit for consistency checks later, in this case it is the contacts.
circuit operator for NC is now applied, so an OR gate G2 is added, then each component on the circuit is dealt with in turn. The decision table for the contacts, Table 4, is viewed for causes of NC in output 2, there are three matches in the output column and these occur in rows 3, 5 and 6: (a) NC, -,-, -; (b) -, -, F; or (c) -, EN, -. The inputs from any of the components on the circuit can be deleted from the three matched rows as these just trace the path of the circuit. In this case the first input column is from L12 and the second from the relay, therefore since L12 is part of the circuit all inputs in column 1 can be ignored (left-hand branch of operator), which just leaves rows 3 and 6 to consider. As there are still two rows, an OR gate, G3, is added to G2 of the fault tree and these rows are expanded in turn. Row 5 of the contacts decision table contains only the component in its failure mode, therefore the failure (CSF) is added to G3, see Fig. 8. Any inputs that are not from the circuit are described as external inputs, in this case the input in row 6 is from the relay which is not on the circuit. When an external input is encountered the component name, output column and variable are checked against all the inputs in the circuits, to see if it lies on one of the other circuits. In this case relay 02 EN does not lie on any of the six circuits, therefore this event will be further expanded by the normal fault tree techniques later. The next component on the circuit is BulbR(02), the component is dealt with in the same manner as for the contacts. From the decision table for the bulb there are two matches in OUT2 for the variable NC, the matches occur in rows 2 and 3. As the bulb only has one input, this is the input that traces the path round the circuit and, as before, can be ignored, leaving only the failure of the red bulb (RBF) to be added to G2. All of the other components on the circuit are dealt with similarly, the only difference occurring with the T-connector, for which a match is found in circuit 2 for T4(OUT1)NC. The decision table for the T-connector is shown in Table 5.

For consistency, the second circuit on which the T-connector is located has to contain the input component on the loop being traced with the traced condition matching the correct output column and also a power supply. Without these conditions it cannot affect the fault conditions being traced. It can be seen from circuit 2 that it does contain a power supply, gen2, but does not contain the component with the correct output column, i.e. Cont(02)NC. Therefore, no entries are incorporated on the fault tree for this component.

When the application of the circuit operator has been completed the stored external input to G3 is expanded. In this case the variable in input 2 of the contacts is EN. This is connected to the relay output column 2. As this variable and output column do not appear on any of the circuits the fault tree is constructed in the normal manner. Output column 2 of the decision table for the relay is searched for the variable state EN, only one match is found which is in row 1 of the table, the inputs to row 1 are C (current) and W (working). As there are two inputs, an AND gate (G4), is added to the OR gate G3 containing the input CSF. A NOT gate is then developed from G4 to deal with the working state of the relay. The working state is converted to NOT failed states since

---

**Table 5** Decision table for T-connector

<table>
<thead>
<tr>
<th>IN1</th>
<th>IN2</th>
<th>STATE</th>
<th>OUT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td>W</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>W</td>
<td>C</td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td></td>
<td>NC</td>
</tr>
</tbody>
</table>
failed states are only desired in the fault tree to ensure the correct Boolean reduction during the analysis. The relay input 1 is connected to line 4 output 1 and the variable now being traced is C (current). All of the circuits are searched for line 4 01 C and a match is found in circuit 3. Now the circuit operator for current is applied to give the complete fault tree shown in Fig. 8.

Boolean reduction on the fault tree in Fig. 8 gives the following causes of the top event:

1. RBF
2. G2F
3. L12F
4. CSF
5. RYF L4F AC L2F L1F G1F L9F L8F L7F L6F

Removing the success states in failure combinations gives the correct cut sets for the top event RBOFF.

8 CONCLUSION

The newly proposed method for automating the fault tree construction process has been successfully demonstrated by application to the train detection system. Although this is only a small example it has features typical of larger binary state circuitry. Utilizing the circuit operator was shown to produce a more efficient algorithm, reducing time and effort testing the logical consistency of repeated events encountered each time a new gate was added to the tree. It also has the added advantage of significantly reducing the size of the trees. When the circuit operator is not applied the resulting fault trees contain many obsolete branches. The method described has been incorporated into a computer software package. This package will take its input from a P&ID type system definition and has a library of component decision tables to formulate the system being modelled. In addition to the binary state systems this method, by its use of operators, also has the capability to model fault development in continuous process plant control systems.

REFERENCES