Sequential effects in failure mode analysis

Taylor, J.R.

Publication date:
1974

Document Version
Publisher's PDF, also known as Version of record

A. E. K. Risø

Risø - M - 1740

Risø - M - 1740

Title and author(s)

Sequential Effects in Failure Mode Analysis

by

J.R. Taylor

Date
August 1974

Department or group
Electronics

Group's own registration number(s)
R-12-74

pages + tables + illustrations

Abstract

In systems which involve sequential control, standby safety systems, and safety shut down procedures, the sequence of actions is important in describing normal operation. Event sequence is also important in describing failures in such systems in many cases. Several examples can be given.

Cause-consequence analysis techniques are especially useful for studying such systems, firstly because they offer a systematic way of building a mathematical model of the failure process; and secondly, because "sequence" is treated naturally by the method. A third advantage is that many different consequences (TOP events) can be treated together, using the same analysis.

Cause-consequence analysis can be formalised, to provide a (semi) automatic method of failure mode and effects analysis. The "plant" is represented by a block diagram, with arcs representing causational links, and the blocks being described by arithmetic or logical transfer functions. A "condition" is a predicate which restricts the possible states of a system (usually by restricting the range of values of a single system variable). An "event description" is a pair of conditions, one of which is true before the event time, the other true after the event time. Event sequences can be traced through the block diagram of the system, using techniques developed for automatic theorem proving to deduce the next event at each block.

Copies to

Available on request from the Library of the Danish Atomic Energy Commission (Atomenergikommissionens Bibliotek), Risø, DK-4000 Roskilde, Denmark
Telephone: (03) 35 51 01, ext. 334, telex: 43116

ISBN 87 550 0287 0

CONTENTS

SEQUENTIAL EFFECTS IN FAILURE MODE ANALYSIS

INTRODUCTION

The kind of effects important in failure mode analysis can best be illustrated by example. The following is a simplified "adaptation" of an incident which occurred in a nuclear power plant.

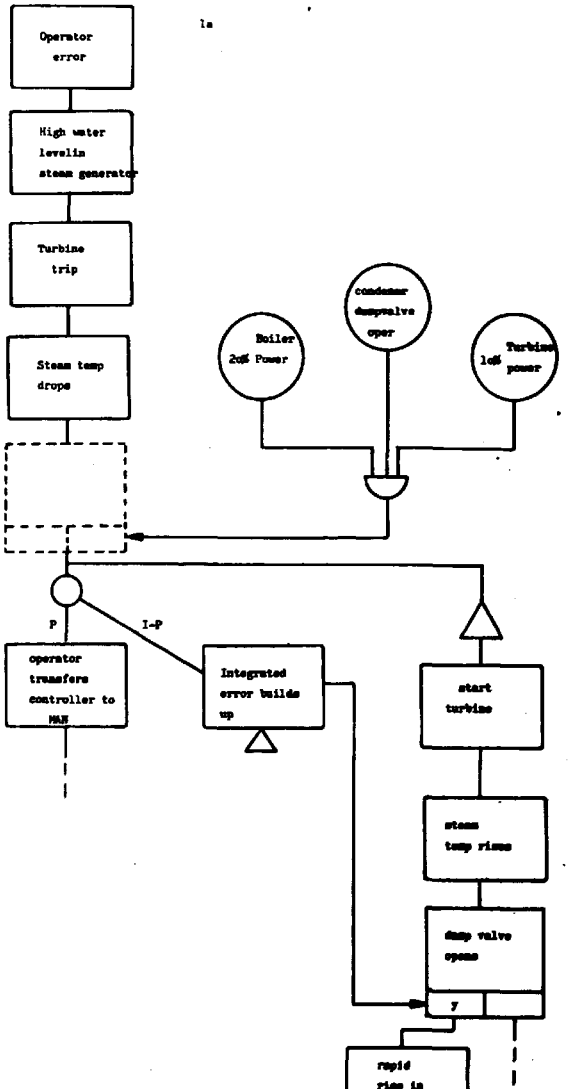
First an operator control error occurred leading to a high water level in a steam generator. The reactor was operating at 20% of full power, with the turbine operating at 10% of full power, the remaining steam being dumped directly to the condenser via the turbine by-pass valve. The high water level activated protection circuits which initiated a turbine trip, and the turbine trip in turn caused a reactor trip. Steam temperature dropped, and protection circuits, correctly, applied a signal which closed the steam dump valve.

The steam header pressure controller remained set on "AUTO" during the following period, and integrated the error between shut down steam pressure and pressure set point. Later, the steam temperature rose, and the protection circuits again permitted steam dump. The steam dump valve opened, and since the pressure controller had integrated error over a long period, the valve opened fully. Steam flow rose very rapidly, and activated many of the reset safety systems.

While no threat to safety was involved in this incident, it shows the kind of situation in which sequence and time are important.

The fact that the pressure controller was not transferred from AUTO to MANUAL would often not matter very much. But in this case, a long delay in execution of an (unusual) manual operation, plus the long delay in temperature rise, combined to make an incident worse than expected.

The plant diagram is shown in fig. 1.



CAUSE CONSEQUENCE DIAGRAMS

Cause consequence diagrams provide a method for describing the failure characteristics of a plant (D.S. Nielsen 1970, D.S. Nielsen el al. 1971). Compared with the fault tree technique used alone they have two advantages.

1) There is a systematic technique for constructing the diagrams, given just a block diagram or wiring diagram of the plant.

2) Sequence is shown explicitly. This makes the diagrams especially useful in studying start up, shut down, and sequential control problems.

The construction of cause consequence diagrams can be formalised (Taylor 1973). The advantages of formalisation are both theoretical - the meaning of the diagrams can be better defined - and practical - it becomes possible to construct the diagrams by computer and build probability models directly from the diagrams.

Events and conditions.

The first step in formalising failure mode analysis, is to describe mathematically the idea of one event causing another. And the first step in this process is to formalise the idea of a "condition".

The idea of a "condition" is "a description of a situation or state (of a process plant)". The kind of thing one wants to express is "component x is working" or "component y has not yet been repaired".

Definition:

A condition is a predicate (truth function) describing the values of (plant) variables which may be true at several consecutive instants of time.

(This definition may appear fairly restrictive and it is worthwhile checking to see if it covers the kind of ideas we want to express. Expressions such as "steam pressure is less than rupture pressure" fall naturally within the definition. An expression such as "pump is working" can be expressed by treating "working" state" as a state variable. We might also want to express [illegible] things as "condensate tank has been filling for at least 10 minutes" and this again can be expressed as a fact about a state variable. However "condensate tank has been filling for exactly 10 minutes" lies outside the definition, which is appropriate since such a statement seems to be more associated with [illegible] with "events".)

Definition:

An "event description" is a pair of conditions, one of which is "true"during an interval immediately before the "event time", the other being true during an interval immediately after the event time. The two conditions should not be consistent with each other. (Otherwise no change has taken place).

A "specific event description" could then be defined as an event description, coupled with the time at which the event takes place e.g. "at 12 o'clock, the steam pressure rises above 10 atmospheres". (Formally, this should be interpreted as for an interval immediately before 12 o'clock steam pressure is less than 10 atmospheres, and for an interval immediately after 12 o'clock, steam pressure is greater than 10 atmospheres).

Constructing cause consequence diagrams

Starting with an initial (specific) event, such as failure of a relay or the pressing of a start button, the sequence of events can be traced through a block diagram of a plant. The block diagram shows the physical components of the plant and has lines representing interactions between the components.

Fig. 2 Event chain tracing.

Each interconnection line in the block diagram is associated with a particular plant variable and the direction of cause and effect is shown for each of the lines. The behaviour of each component is described by means of equations or transfer functions.

Given the description of an initial event in one component, the effect of this event on the next component in the causal chain can be obtained by a process of deduction. The result will be a description of the state change in the second component, and a description of the event occurring at the output of the second component (see fig. 2). When this is performed by computer, techniques of deduction similar to those used in automatic theorem proving can be used.

The true situation is in fact more complex than this, because in practice

1) a component may have several outputs,

2) a component may have several inputs,

3) the output event will generally depend on component state, and hence on earlier events,

4) a given input event may result in several alternative output events, which event occurs being selected completely at random. Human errors are often described in this way.

5) the input event may establish a "process", that is, a gradual change which may only later lead to an event (e.g. the opening of a valve leads to filling of a tank, which may later lead to overflow).

All of these effects must be built into an algorithm for tracing event sequences. Several alternative algorithms are possible.

Event tracing can proceed as long as there are no "immediate" or "analogue" feedback loops in the plant block diagram. If such a situation exists, the components involved must be separated out and treated collectively as a single component. The effects of an event on such an "analogue network" may in some cases be deduced analytically. But in most cases, deduction of the "next event" for such a network requires analogue or numeric simulation techniques.

Processes

In tracing event chains, it often happens that the next event in a sequence depends on the state of a component, or the condition which obtains at other inputs to the component. This dependency yields a fork in the event chain, which can be expressed by means of a "decision box" (Fig. 3). If the condition or state is independent of earlier events, then a description of the condition can be written in the decision box, or can be expanded by a type of "fault-tree" - a "condition tree" attached to the decision box.

In some cases, and especially those involving multiple failures, the next event in a chain will depend on conditions established by earlier events. The cause consequence diagram must express this dependency, and suitable symbols are shown in Fig. 7.

Conditions established by an event may endure for the whole period covered by an analysis and dependency on this kind of condition is relatively easy to express. In other cases, a condition may be established temporarily as a result of one event, and be terminated as a result of another event. Repair is an example of this type of phenomenon. The event "start of repair" starts the repair process, and "completion of repair" terminates the repair process.

In such cases as these, special symbols are used to express the concept of a process, as shown in Fig. 9. A process may be defined as a condition which describes continuous change in state of some part of a system.

In some cases a process will terminate by its own completion. In other cases, a process will be terminated as a result of other extraneous events.

Formal versus manual analysis

The techniques described in this paper are formal, and can therefore be applied directly by computer. In practice such direct application is undesirable. The amount of work involved would be very large because the formal analysis proceeds by considering every possible initial failure event, and the analysis results would be confusing.

The manual techniques described by D.S. Nielsen (1974) make use of heuristic rules to reduce the amount of work involved in analysis and to make the results comprehensible. Some of these "heuristic" rules are sufficiently powerful that they preserve the completeness of the formal analysis, e.g. the concept of a "critical event" is applicable to all failure sequences. Other heuristic rules described by D.S. Nielsen are presented in an earlier report (Taylor 1973).

The formal technique of consequence analysis is very similar to some of the steps used by an engineer in failure mode analysis. It is envisaged that automatic failure mode analysis will be used by an engineer on an interactive basis, to help extend the completeness and thoroughness og his analyses.

CAUSE CONSEQUENCE DIAGRAM SYMBOLS

The choice of symbols for cause consequence diagrams is an important practical consideration. At the same time, choice of symbols illustrates the type of phenomena which one meets in cause-consequence analysis.

The primary symbols used in cause consequence diagrams are event and condition boxes and decision boxes. Fig. 3 gives an example of the use of the basic symbols.

An important point in developing the set of symbols as shown here is that different kinds of probability dependency are distinguished by using different symbols. As a result there is an isomorphism between the structure of the cause consequence diagrams, and the event probability density functions (see e.g. Taylor 1974).

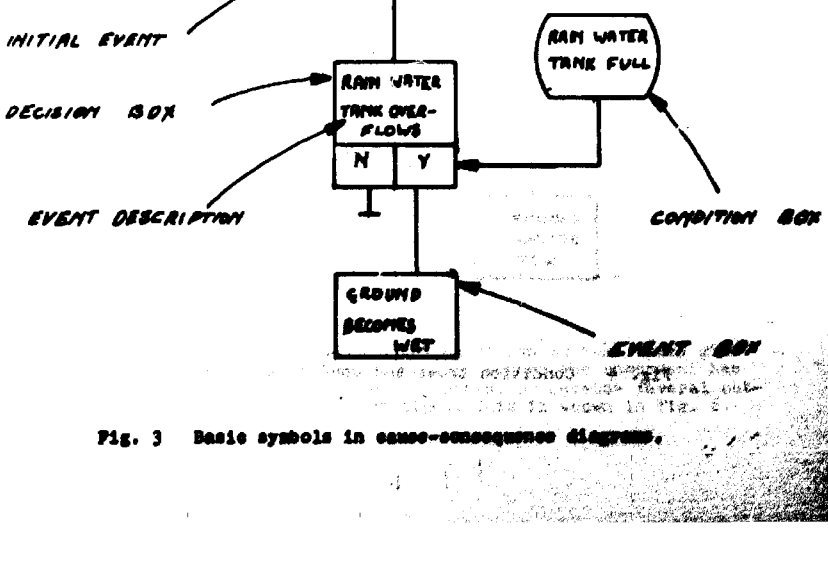


Fig. 3 Basic symbols in cause-consequence diagrams.

It is useful to be able to combine conditions, and the normal fault tree symbols, and gates, or gates, and negation indicators are used for this purpose. It is also useful to be able to introduce a condition directly into a sequence of events as in Fig. 4.

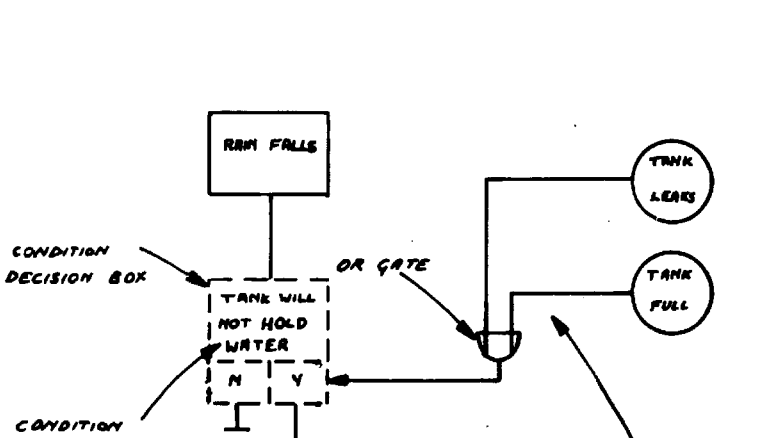


Fig. 4 Condition trees and condition boxes.

"Or" gates can be used in a similar fashion, to shorten diagrams in which several event sequences have a similar result (Fig. 5).

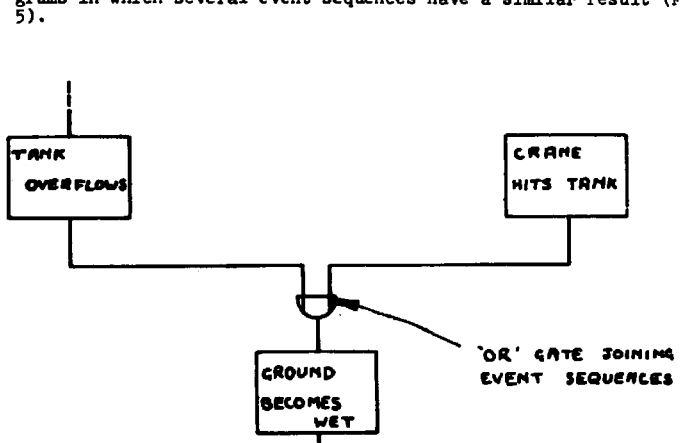
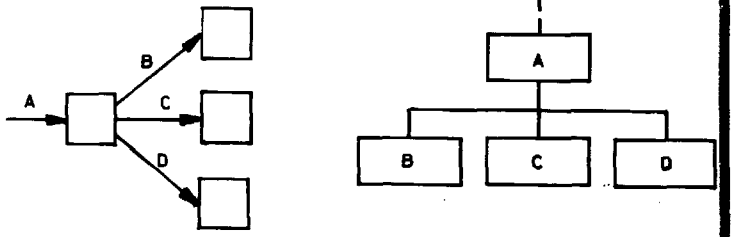


Fig. 5 Joining two event sequences.

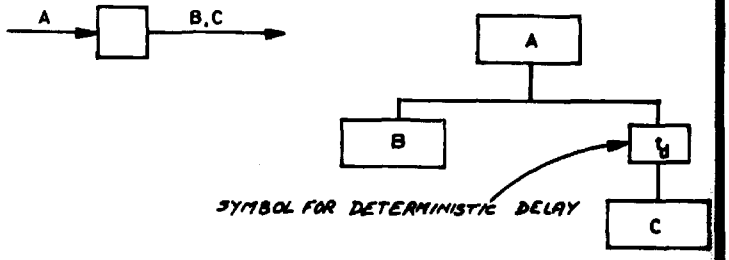
A single input event to a component can in some cases give rise to several output events, either because the component has several output lines to other components, or because several output events result at different times. This is shown in Fig. 6.

BLOCK DIAGRAM

CAUSE CONSEQUENCE DIAGRAM



One component with several outputs



Component with immediate and delayed consequences

Fig.6 Forking in Event Sequences

With the possibility that two sequences of events develop "simultaneously", the situation arises in which one event sequence can give rise to conditions in which another event sequence is unable to develop. Situations in which this kind of effect is significant occur especially where there is a variable or random delay involved. The symbols for expressing this situation are illustrated in Fig. 7.

In some cases, a single event may lead directly to one of several alternative immediate events. This is often the situation with human actions (Fig. 8).

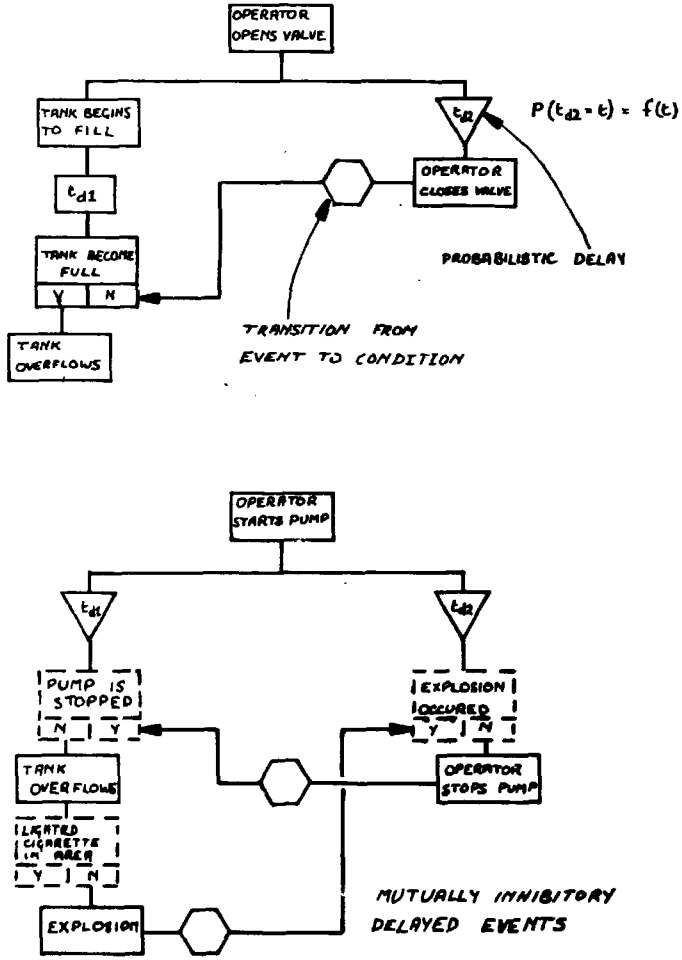


Fig .7 Events which inhibit other events

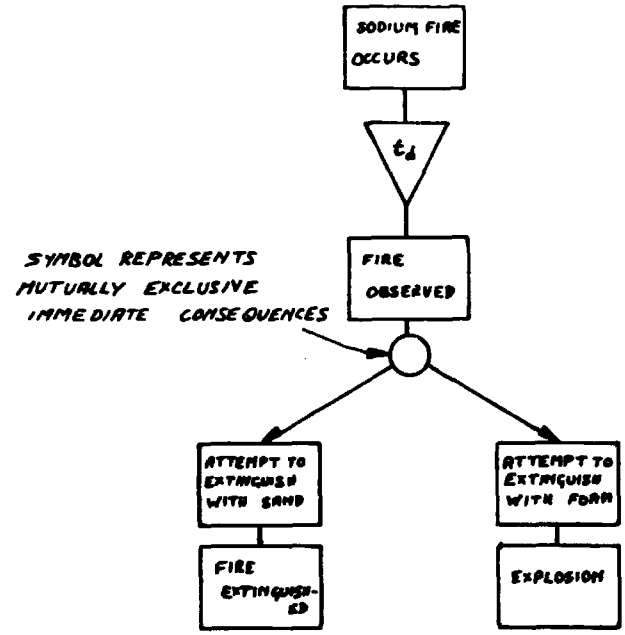


Fig 8. Alternative immediate events

Processes can be represented by a combination of the symbols for an event and a delay, as in fig. 9. It can be seen that a process description is a "shorthand" for a more complex description in terms of initiating and terminating events.

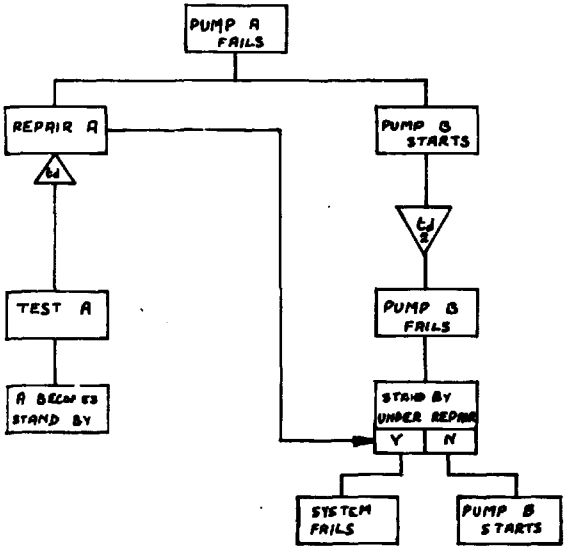
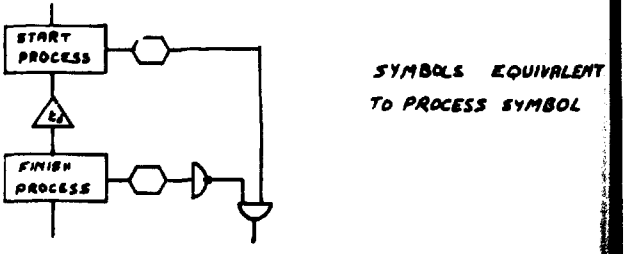


Fig.9 Use of 'process' symbol —— repair process in redundant system



Time conventions

A convention has been adopted for marking the times of events. An initial event time t_o is ascribed to the first sequence initiating event studied. Deterministic delays then lead to time indications $t_o + 1$, $t_o + 55$, etc. Delays which are not deterministic require that a new time base variable is introduced e.g. t_1, and the fact that t_1 is greater than the previous time indication in the event sequence, is recorded e.g. $t_1 > t_o + 10$. When independent events occur (such as new initiating event, or randomly delayed events in two parallel event paths) a new time base variable is introduced, but in this case, its magnitude relative to other time base variables may not necessarily be determined.

Fault tree equivalents of cause consequence diagrams

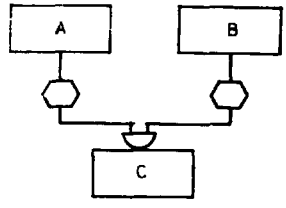
It is possible to translate descriptions from cause consequence diagram notation to fault tree notation. The interpretation of the events is somewhat different however. The "event boxes" of fault trees correspond to "the condition that an event has happened" in the cause/consequence notation. The notation
A is introduced to represent the condition
"A has occurred".

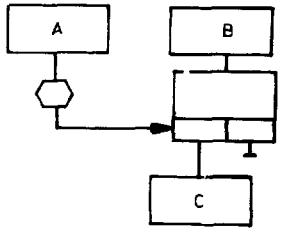
Fig. 10 shows some of the simple fault tree/cause consequence diagram equivalents. Combinations of conditions using and, or, and not gates are idential for cause consequence diagrams and for fault trees.

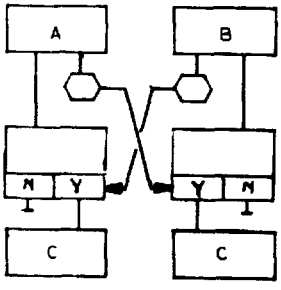
Sequences of events can be represented in fault trees using a similar technique to that used in cause consequence diagrams, of simply using a sequence of boxes, connected by lines as in fig. 11a. An alternative notation is shown in fig. 11b. The usefulness of this alternative notation is illustrated by the example "event D occurs if and only if event B occurs between events A and D".

The cases where one event directly gives rise to a second event also require new symbols, as in fig. 12 and 13.

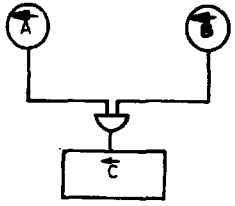
CAUSE CONSEQUENCE DIAGRAM

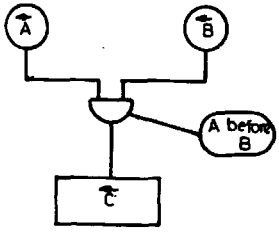


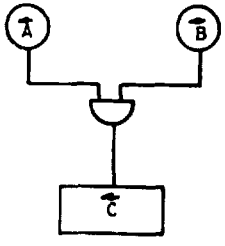




FAULT TREE



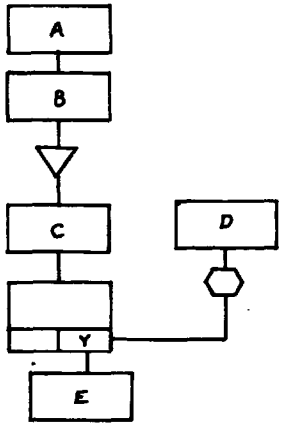




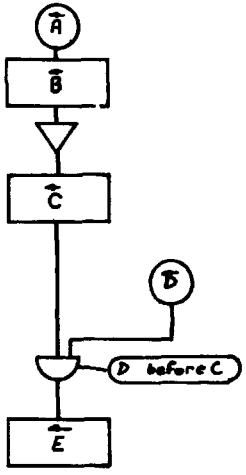
$\bar{A}$ INDICATES THE CONDITION THAT EVENT A HAS OCCURED

Fig. 10

CAUSE CONSEQUENCE DIAGRAM



SUGGESTED FAULT TREE SYMBOLS



ALTERNATIVE

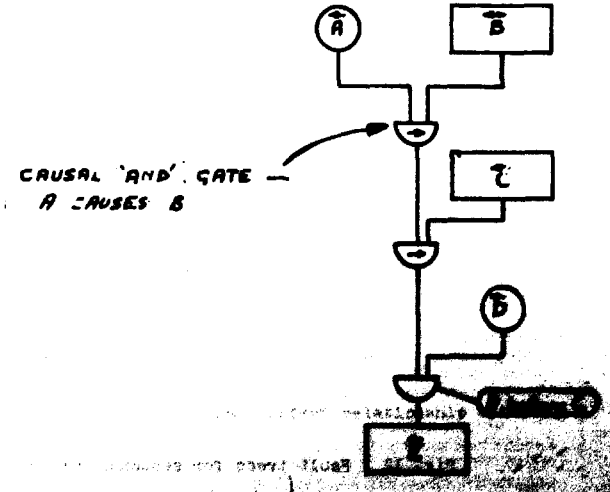
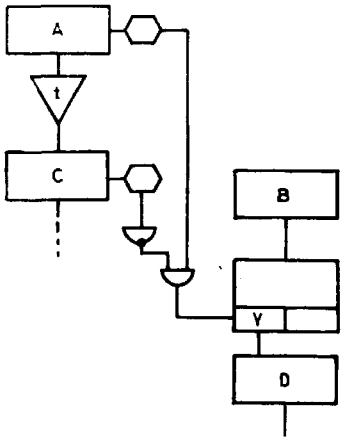


Fig. 11 Sequences of events in fault trees.

CAUSE CONSEQUENCE DIAGRAM



B occurs between A and C

SUGGESTED FAULT TREE

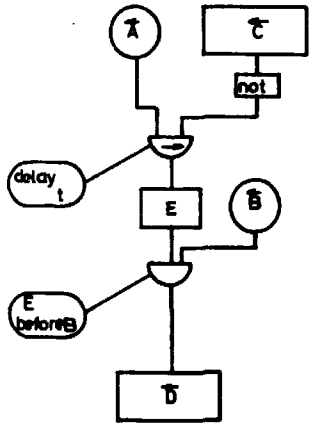
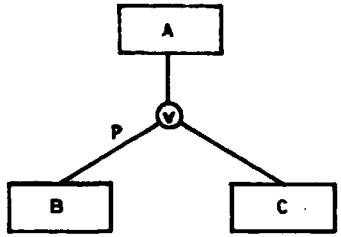
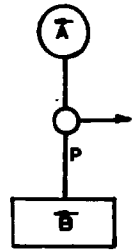


Fig. 12 Fault trees for sequence constrained events.

CAUSE CONSEQUENCE DIAGRAM



SUGGESTED FAULT TREE SYMBOLS



ALTERNATIVE FAULT TREE SYMBOLS

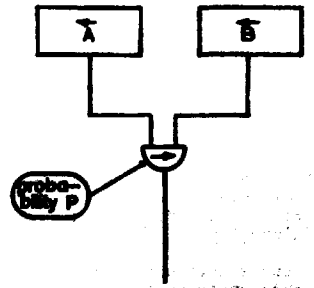


Fig. 13 Probabilistic cause effect relationship between events.

ALGORITHMS FOR CAUSE CONSEQUENCE DIAGRAM CONSTRUCTION

There are several possible algorithms for developing cause consequence diagrams, starting from a block diagram of a plant, with causal directions marked, and an initial set of possible plant states.

The first algorithm given here traces the consequences (events) arising from a single initial (spontaneous) event, in time sequence. By tracing events in time sequence, problems of deciding whether a particular condition holds when developing the next event, are reduced. The algorithm works on the tree of events which it is known can occur, and outputting new additions to this tree directly.

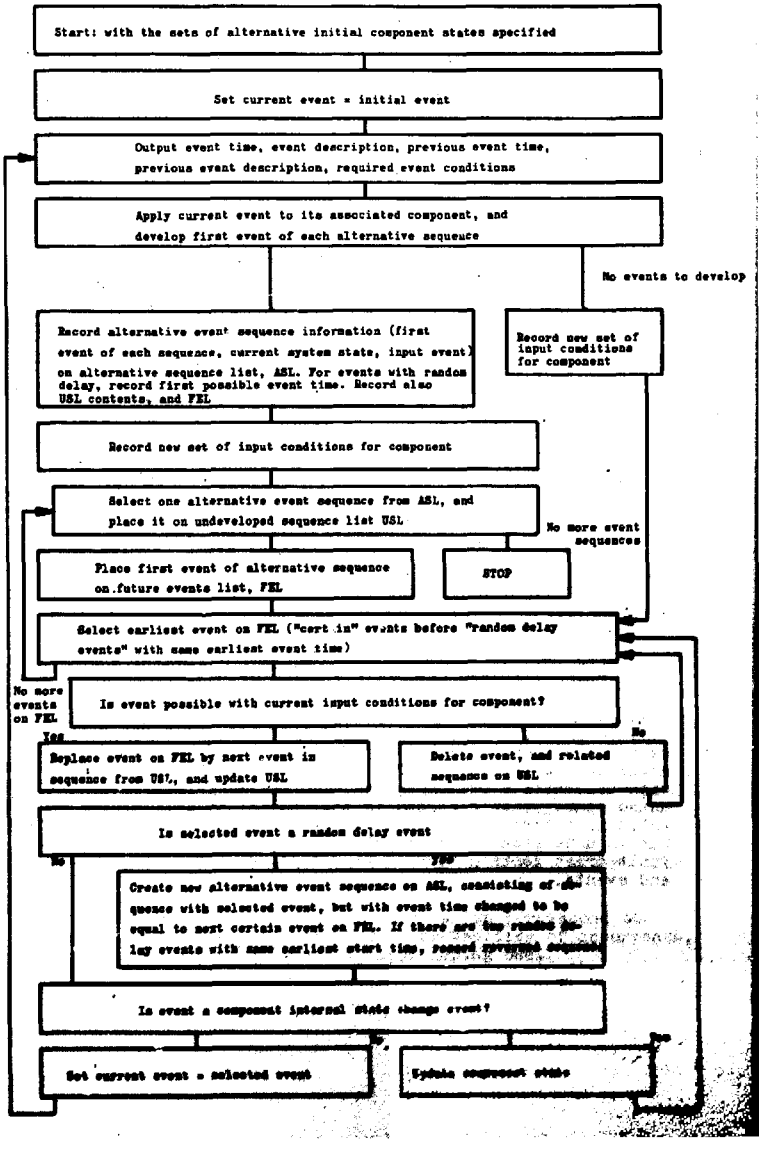
By direct consequences of an event is meant the set of events which can occur at the output of a plant component, when the event is applied at the input of a component. The "direct consequence" events are grouped naturally into alternative sequences, depending on the possible internal states and possible input signals on other input lines to the component.

At each stage in adding a new event to the tree, the description and first event of each alternative direct consequence sequence is evaluated and set on an "alternative sequence list". One such alternative sequence is selected, and is set on an "undeveloped sequence list". The first event of the sequence is then set on a "future events list". The earliest event on this future events list is then selected for further development. In this way, two mutually consistent event sequences in different parts of a block diagram can be developed concurrently.

The algorithm in fig. 14 follows these principles. The algorithm has been to some extent simplified. For some components, (for example timer/counters) it is possible for an input event to trigger an infinite sequence of output events. In this case, the algorithm will not terminate.

One solution to this problem is to store a record of each sequence under development, and as each event is added to a sequence, to check for a repetition of events and conditions. When the algorithm is performed manually, this is unnecessary. The analyst can control the development of infinite sequences using inductive reasoning. When the algorithm is performed by computer it may also be useful to allow human monitoring of event sequences, since the automatic sequence matching process reduces the effectiveness of the algorithm considerably.

When random delay events occur, the algorithm develops different sequences for each possible timing of the random event relative to other events. This is necessary because the form the sequences take may be altered by the different relative timings.



However, in the case where relative timing of a randomly delayed event is irrelevant, the resulting alternative sequences should not appear on the resulting cause consequence diagram. Again, either automatic output editing, involving matching of sequences, can be used; or manual control of the cause consequence analysis process may be preferred.

The algorithm for single failure (single initial event) consequence analysis can be applied in multiple failure analysis using an iterative process. The method of use depends on which type of multiple failures is involved.

A latent failure is one for which there are no immediate harmful consequences, but which alters the state of the system, so that some later event triggers a harmful consequence. Most multiple failure occurrences are of this kind. An immediate failure is one which leads directly to harmful consequences, even in the absence of later failures. To use the algorithm of Fig. 14 for this kind of situation, each type of failure within the system is analysed individually. The resulting system and states are recorded, and then used as possible initial states in double failure analyses (one immediate and one latent failure). The process can be iterated, to account for situations in which several latent failures are present.

In practical situations, failure frequencies are usually so low that it is unnecessary to analyse the consequences of a failure which occurs while another failure sequence is in progress. However, common mode failure mechanisms often involve several immediate failures, and this kind of analysis becomes important. In this case, multiple failures can be represented by a "failure event process", in which the initial input failure event is followed by further failure events, separated by random delays.

Fault tree algorithm

In an earlier paper, an algorithm was given for converting cause consequence diagrams to fault trees. (Taylor 1974). This algorithm is repeated in Fig. 15, and is extended to include events with alternative outcomes, and time delay. The object in editing cause consequence diagrams in this way is to remove information which is interesting from a failure mode analysis point of view, but is unnecessary for probability calculation. The resulting fault trees contain "boxes" representing conditions, reliabilities, and events.

Fig. 15 Algorithm for editing cause consequence diagrams to form fault trees

1) The algorithm of Fig. 14 produces a consequence tree for each initial event. The event nodes at the ends of branches of this tree are result nodes. One tree may have several result nodes which are identical with respect to event description, and different trees may contain result nodes which are identical in this respect.

2) Select the result (TOP event) of interest. Mark all event paths leading to this set of result nodes, and remove all other event paths from the consequence trees. If two or more paths have a common subpath, which forks via a condition box, an event decision box, or an alternative event outcome node, join the paths immediately before the result node using an "exclusive or" symbol. Fig. 15A.

3) Join all of paths established in step 2 and unjoined paths, immediately before the result nodes, using an "or" symbol. See Fig. 15B.

4) Replace all time delay symbols by time delay "causal and" symbols, as in Fig. 11.

5) Replace condition boxes and event decision boxes by "and" gates with specified event sequence, as in Fig. 10. If both outcomes of a decision box or condition box lie within the marked consequence paths, duplicate the paths leading to the box, and use two sequential and gates, one with the positive condition marked, the other with the negative condition, each leading to its appropriate consequent event sequence.

6) Replace alternative event outcome symbols by probability "causal and" symbols as in Fig. 13. If both alternative outcomes lie within the remaining event paths, replace the alternative event outcome symbol by two probability "causal and" symbols, and duplicate the event paths leading to the alternative event outcome symbol.

7) Replace event sequence forks (parallel event sequence symbol) by duplicating the sequence preceding the fork and removing the fork symbol. Fig. 15c.

8) Remove all event boxes in the consequence trees, apart from initial event boxes.

9) Remove all condition and event boxes which correspond to normal operation, and which have a probability of occurrence, during the period of interest, approaching one.

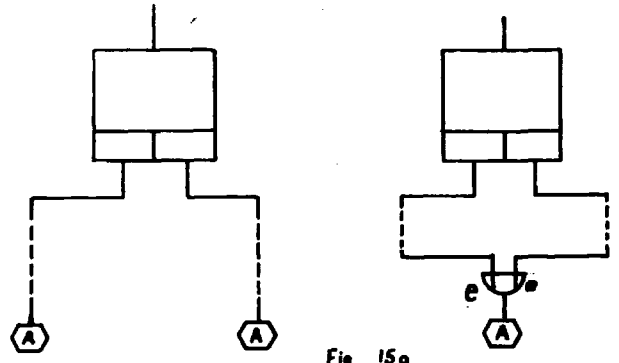


Fig 15a

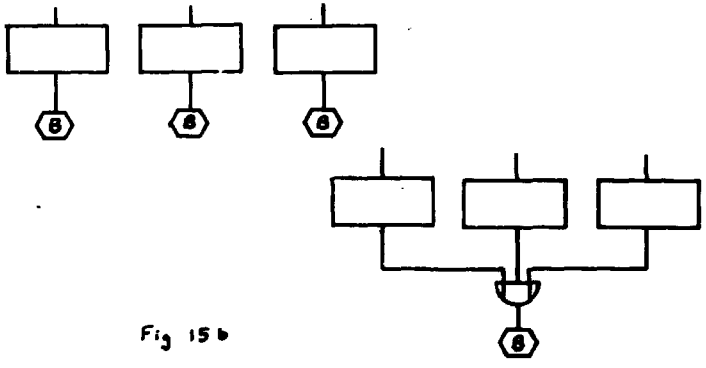


Fig 15b

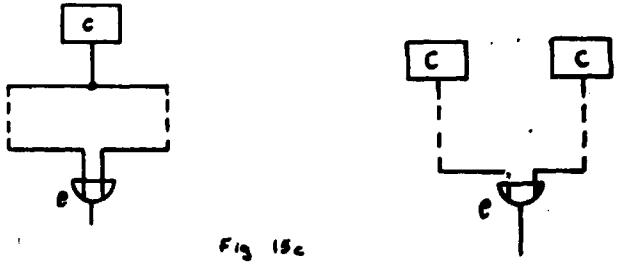


Fig 15c

CAUSE CONSEQUENCE ANALYSIS GENERATES MINIMAL CUT SETS

It has been pointed out by D.S. Nielsen (1974) that cause consequence diagrams produced by hand involve only minimal cut sets (c.f. Fussel 1973). While this is not strictly true of the construction algorithm of Fig. 14, and the editing algorithm of Fig. 15, the exceptions are unlikely to occur in practice. It is possible to extend the algorithm of Fig. 14 so that the cause consequence diagram, when edited by the algorithm of Fig. 15, contains only minimal cut sets. This is useful, because the probability computation rules for the resulting fault trees are much simplified.

A proof that the extended algorithm generates minimal cut sets is presented here, and the extensions to the algorithm are presented simultaneously. In this way, it is easier to see why the extensions are required.

Definition:

In cause consequence diagram terms, a minimal cut set is defined as a set of initial events and primary conditions, which together can lead to a particular unwanted result, and which does not contain a subset of initial events and primary conditions which can lead to the same unwanted event.

Theorem:

The extended cause consequence diagram construction algorithm generates minimal cut sets.

1) To simplify the proof, the term "partial cut set" is introduced, to represent the set of initial events and primary conditions which must hold in order to reach a particular point within an event sequence.

2) Assume the converse of the theorem. Then there are two cut sets for a given result (TOP) event, one of which is a subset of the other.

3) Since each partial cut set arising in a cause-consequence diagram must contain an initial event, and each initial event occurs in just one consequence event [illegible], the [illegible] minimal partial cut set must occur in the same consequence event [illegible] minimal partial cut set.

4) It is assumed that condition [illegible] boxes to an event sequence path, [illegible] normal form, with only minimal [illegible] in so far as they occur, are [illegible] conditions arising directly from [illegible]

This assumption is satisfied naturally by the event deduction algorithms which could reasonably be used in cause consequence analysis

5) In order for two separate partial cut sets of any form to arise within an event consequence tree, there must be an "or" symbol within the tree. Such an "or" symbol can arise in either of two ways -

a) because of an "or" symbol within a condition tree.

b) because of forking (due to parallel event sequences, or alternative event sequences) in the event tree, where both branches of the fork lead to the same event.

6) Case a

Given an event sequence with a decision box as its last node; the partial cut set prior to the decision box can be represented as

$$B_1 \vee B_2 \vee \ldots \vee B_n$$

Where B_1 is a partial cut , that is, a conjunction of conditions and events. The condition associated with the decision box can, according to 4), be represented as

$$C_1 \vee C_2 \vee \ldots \vee C_m$$

where C_1 is a partial cut . The partial cut set immediately following the decision box will be

$$\bigvee_{\substack{i=1 \ldots n \\ j=1 \ldots m}} B_i \& C_j$$

This new partial cut set will contain non minimal cuts only if there are p, q, r, s such that

$$B_p \& C_q \leq B_r \& C_s$$

Assuming that each Bi is a minimal cut (induction hypothesis)

$$B_p \not\leq B_r$$

And from assumption 4

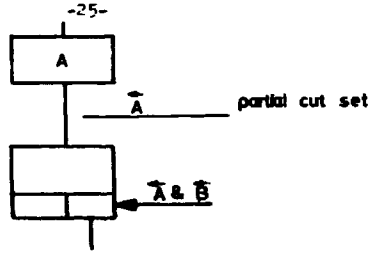
$$C_q \not\leq C_s$$

Therefore we must have

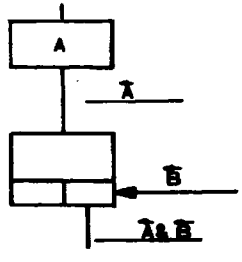
$$B_p - B_r \leq C_s$$
or $$C_q - C_s \leq B_r$$

7) The situation described in step 6 can occur in principle, in the development of a cause consequence diagram. A simple example

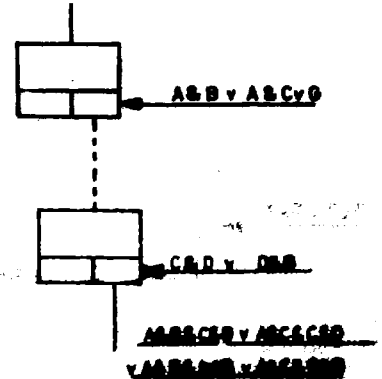
is



This corresponds to the case in which a condition for further development of an event sequence is automatically fulfilled, as a condition of its earlier development. By maintaining a record of the partial cut set of each event sequence, such conditions can be removed, to give

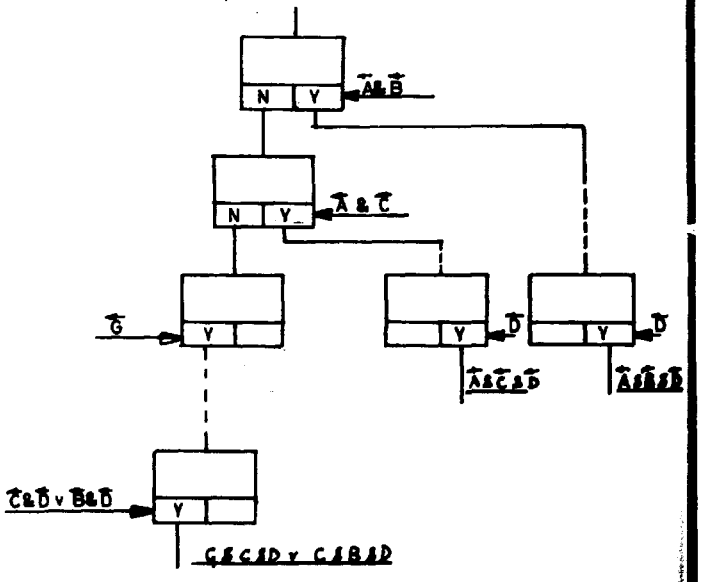


8) More complex examples than that of 7) can occur e.g.



In this case, however, conditions may not simply be deleted, since the result would be incorrect.

This situation can be detected, if a record of partial cut sets is kept for each event sequence, and a check is made for cases where Bp⊆Cq ⊈ Br⊆Cs. The problem can then be removed, by duplicating the previous event sequence, once for each problematic cut set, and then deleting redundant conditions, e.g.



9) By induction, if the construction algorithm is extended in the way described in steps 7 and 8, non minimal partial cut sets cannot be introduced via a condition tree.

10) case b

Forking in the consequence event tree can be introduced via

c) condition box or alternative event outcome nodes.

d) parallel event sequences.

11) case c

A condition box fork, or alternative event outcome fork, require that an additional condition is added to the cut sets. A positive condition is added to the cut sets on one side of the fork, a negative condition is added to the cut sets on the other side of the fork. Therefore, when the two branches are later joined via an 'or' gate, cut sets from one branch cannot be subsets of cut sets from the other branch.

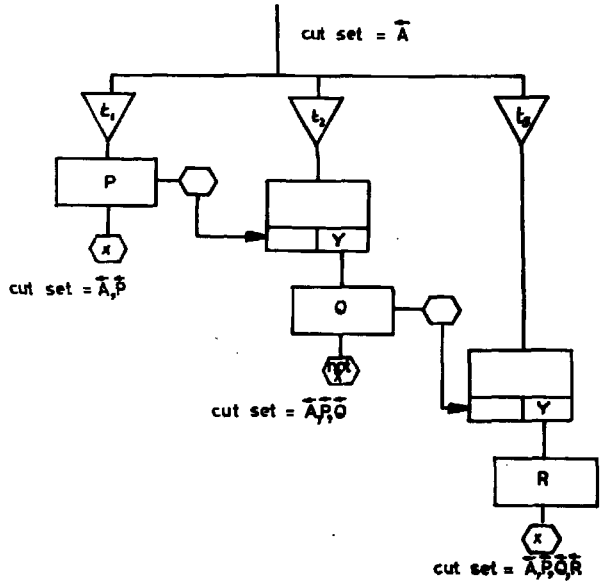
12) case d

This case involves two parallel event paths within the block diagram, both of which lead to the same component, and cause the same result event. Then either the time delays are identical for the two paths (case e); or a condition established as a result of an event on one path, is later negated (as a result of a delayed event within the component, or via an event sequence along a third path) and then still later is reestablished by the second event sequence (case f).

13) In the case e, a trivial extension to the cause consequence diagram construction algorithm can ensure that two events never occur simultaneously (by adding a small increment to one event time so that two events can never occur precisely simultaneously.)

14) case f

This case, where an event happens, its effect is removed by a later event, and the event again happens, can be illustrated by the following example.



If the required result of fault tree analysis is to determine event probability density functions, then both cut sets should be included in the eventual fault tree. If what is required is the probability that at some time t, event x has happened (at least once), then development of parallel sequences should be stopped at the first time at which the desired result event occurs, and processing should proceed to other alternative sequences.

15) Since non minimal cut sets cannot be introduced into a cause consequence graph via condition trees, nor via forking of the event sequence paths, and since these are the only ways new cut sets can be introduced, the extended cause consequence diagram algorithm produces diagrams involving just minimal cut sets.

USING THE ALGORITHMS

Event deduction vs. failure transfer functions

Fussel (1973) has described a technique for constructing fault trees automatically. In this, he used the concept of failure transfer functions. These are sets of schemas for parts of fault trees, associated with particular plant components. The partial fault trees describe both the ways in which the individual component can fail, and also the ways in which the component can transmit the effect of a "failure" event at its input.

It is clear that such failure transfer functions could be extended, to apply to components with delay involved. They already take account of event sequence. Therefore a technique using failure transfer functions could be used to replace the event deduction process, and to eliminate the need for equations for each component. On finding an output event for a component, this would be matched against the set of input events for the failure transfer functions of the next component. See Fig. 16.

This approach has many advantages. The event deduction process can in some cases be inefficient. And the fault tree notation used to express failure transfer functions is readily understood by analysts.

The major advantage of using event deduction is that of space - time trade off. A comparatively short equation can correspond to a large number of transfer function trees. In these cases the equation/deduction form saves space. Clearly, availability of both methods of representation would be best.

A further advantage of the equation form is that it is not committed to direction of cause and effect. So a single set of equations can be used to describe components such as direct current motor/generators, in which causation can be in either direction.

Reversal of consequence analysis

The cause consequence analysis procedures so far described start by listing all the possible component failure events, and working forwards to evaluate all possible consequences of these events. This corresponds to a formalisation of failure mode analysis.

It is also possible to operate the technique in reverse, so that starting from a given unwanted result, all of the event sequences leading to that result are obtained. The procedure is illustrated by Fig. 16, where Fussels "failure transfer functions trees" are used to describe components. The procedure for cause consequence diagrams differs from the one given in Section

for constructing fault trees (Pussel 1973) in the kind of output produced, and in the fact that timing of events must be recorded and used in the algorithm. The determination of relative timings of events becomes a responsibility of the construction algorithm, as does the determination of compatibility between the possibly mutually exclusive conditions established during an event sequence.

Forward cause-consequence analysis is able to provide a description of consequences which is complete with respect to a particular set of component failure models, and up to a certain degree of multiple (immediate or latent) failure. For example, all the consequences of a single immediate failures occurring simultaneously with three latent failures might be evaluated.

It would be desirable to retain this kind of completeness for the reverse analysis, so that all unwanted consequences were studied, while gaining the advantage of ignoring inrelevant consequences. Rasmussen has suggested (1973) a useful heuristic for limiting the number of end consequences to be considered, by assuming that an unwanted result must arise from a release of energy, and that this requires either a large energy flow or large energy accumulation, in conjurction with a breach of the energy containment boundary. In this way, attention can at first be limited to events occurring in energy constraining components. The principle could be extended to components containing occumulations of poison.

For such incidents as explosion, breach of containment is not necessarily a trigger. Powers (1974) has described a technique for discovering failure potential, by searching for sets of factors (conditions) which lead to accidents if simultaneously present.

These techniques are relevant for accident analysis. For studies of production continuity and availability, one should presumably start from the event "cease production of end product" and work backwards.

It has been observed in practice that it is relatively difficult for an analyst to "work backwards", in a direction opposite to that of causality. Instead there is a tendency to guess an earlier event which could lead to a given end result, and then confirm the guess by forward analysis. The programmed algorithms may be able to support manual analysis in this respect.

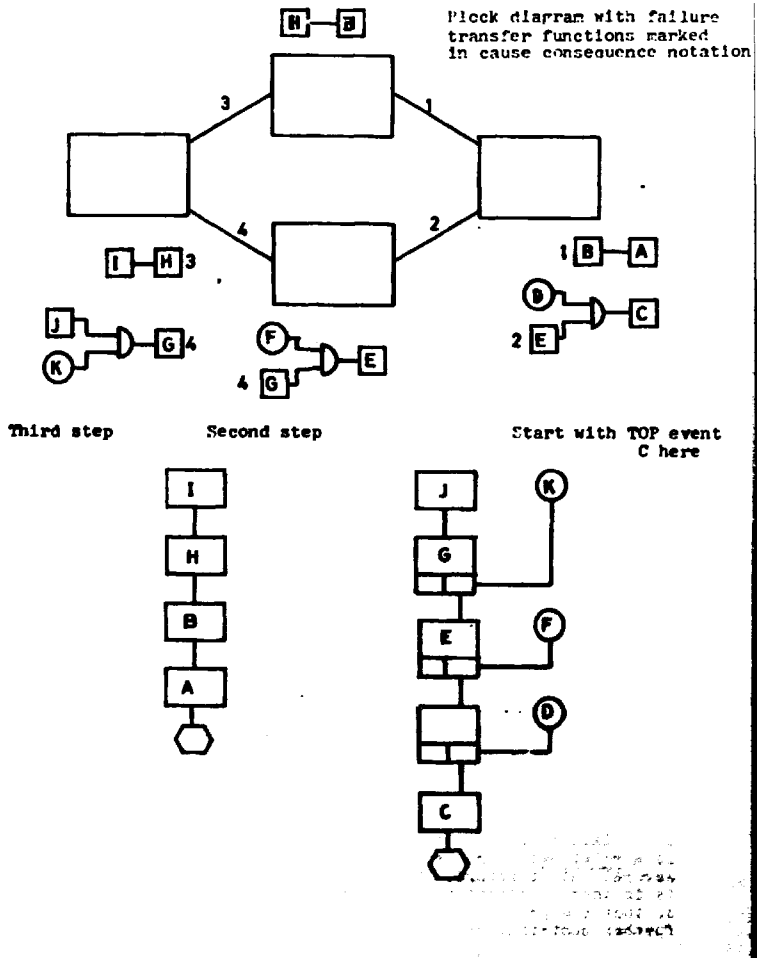


Fig. 16 Example of cause [illegible] direction.

USE OF CAUSE CONSEQUENCE TECHNIQUES - MULTIPLE LATENT FAILURES

Cause consequence analysis is a tool intended primarily for the study of problems in which sequence and timing are important. Nuclear reactor shut down systems are just of this type.

Experience shows that when a failure event of some importance occurs ("abnormal occurrences", as reported in (USAEC 1974) for example), it often involves a single initiating failure, plus several further failures. In an earlier study (Taylor 1974) of abnormal occurrences reported for two nuclear reactors during one year, the frequency of six fold failures is not significantly different from the frequency of three fold failures. While this result is derived from a fairly small sample, the results are also supported by wider less carefully controlled studies. The high frequency of incidents involving several failures is initially very surprising, but becomes less so when it is realized that several simultaneous failures are not involved. What is involved in the simultaneous activation of several latent failures. However, the results depart so much from the usual model derived via fault tree analysis, as to require explanation.

A histogram of the distribution of multiple failure occurrences is shown in Fig. 17. For the purpose of deriving this histogram, incidents were analysed to discover the original cause of failure conditions until a category from the group design, fabrication, installation, operation, or random mechanical/electrical error could be assigned. Common mode failures (those in which an unanticipated causal link excists between components, such as environmental effects on components of similar type) were regarded as single failures for the purposes of the study.

The surprising fact about the form of Fig. 17, is not the high frequency of multiple failures - the frequency of such failures is not regarded by the author as especially high in any case. What is surprising is that for multiple failures from two to ten fold, the frequencies are roughly the same.

Three effects appear to the author to be relevant in explaining these facts.

1) Latent failures are involved. On the basis of the data there is a relatively high probability that shut down systems contain several latent errors. This is clear, on the basis of the data, but is in sharp contrast to the fact that test frequencies are chosen so that the probability of multiple failures should be low. Some further contributing explanation must be sought.

2) Some initial events move the system into a new or unusual operating state, which differ from those assumed during design and the planning of tests. In this new state, several latent failures are activated. This second explanation, taken together with the first, seems satisfactory. For each "unusual state" there is a single initiating failure which triggers it. The frequency of such failures should be somewhat less than one third the frequency of failures which in themselves lead to significant consequences. The number of latent failures activated in such unusual states is then between one and eight (roughly), with the frequency of the different numbers failures more or less evenly distributed.

3) The consequences of multiple failures are often more significant. Therefore in gatherering data on "abnormal occurrences" the number of multiple failure incidents would be unusually high.

This is, in fact, the case, and means that a simple failure which would not merit the "abnormal occurrence" classification, in itself, can under certain circumstances lead to very significant chains of consequences. This effect needs to be taken into account, when estimating the frequency of trigger events, as described under point 1.

It is clear that in arriving at quantitative explanations of such facts, the concept of latent and immediate failures are important. Sequence is also significant, in that the stage at which a latent failure is activated, is often important. For such systems, cause consequence analysis appears to us to be an important tool.

Fig. 17 Relative frequency of multiple failure incidents against number of failures per incident

REFERENCES

J.B. Fussel, 1973 — A Formal methodology for Fault tree construction, Nuclear Science and engineering, 52 421-432.

D.S. Nielsen, 1970 — The Cause Consequence diagram method as a basis for quantitative accident analysis, Report Risø-M-1374. *)

D.S. Nielsen, 1971 — Cause Consequence Diagrams NARS Publication 2 Nordic working group on reactor safety.

D.S. Nielsen, 1974 — Use of cause consequence charts in practical systems analysis, Tobe presented at the conference on Reliability and Fault Tree Analysis, Operations Research Centre, Berkeley, California, September 3-7, 1974.

G. Powers, 1974 — Fault tree synthesis for chemical processes, AICLE Journal Vol. 20 No. 2, March 1974.

J. Rasmussen, 1974 — Om ulykkers anatomi og bekæmpelse (Anatomy of accidents) (in Danish), Report Risø-M-1696. *)

J.R. Taylor, 1973 — A formalisation of failure mode analysis of control systems, Report Risø-M-1654. *)

J.R. Taylor, 1974 — A semiautomatic method for qualitative failure mode analysis, Report Risø-M-1707. *)

USAEC, 1974 — Office of operations evaluation, Summary of Abnormal occurrences reported to the Atomic Energy Commission during 1973, May 1974, OOE-OS-001.

J.R. Taylor, 1974 — Design errors in nuclear power plant, Report Risø-M-1742. *)

*) Available on request from: Library of the Danish Atomic Energy Commission (Atomenergikommissionens Bibliotek) Risø, DK 4000 Roskilde, Denmark. Tel.:(03) 35 51 01. Telex: 43116.