A formalisation of failure mode analysis of control systems

Taylor, J.R.

Publication date:
1973

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

Risö-M-1654

Risö-M-1654

Danish Atomic Energy Commission

Research Establishment Risö

ELECTRONICS DEPARTMENT

A formalisation of failure mode analysis of control systems

by

J.R. Taylor

September 1973

R-9-73

Available on request from: Library of the Danish Atomic Energy Commission (Atomenergikommissionens Bibliotek), Risö, DK-4000 Roskilde, Denmark.
Telephone: (03) 355101, telex: 5072

A. E. K. Risø

Risø - M - 1654

Risø - M - 1654

Title and author(s)

A formalisation of failure mode analysis of control systems

by

J. R. Taylor

Date 1st October, 1973

Department or group

Electronics Dept.

Group's own registration number(s)

R-9-73

pages + tables + illustrations

Abstract

Failure mode analysis techniques using graphic aids to evaluate failure conditions and events have been developed over several years. Here a mathematical description of conditions and events is described, and is related to a mathematical model of a system. A systematic method for deducing event sequences is developed, and the method is applied to practical examples.

The main motivation for formalisation of failure analysis is to provide a concept of completeness or thoroughness for a failure analysis. At the same time, the possibilities for automation of failure analysis are considered.

Copies to

Abstract to

Available on request from the Library of the Danish Atomic Energy Commission (Atomenergikommissionens Bibliotek), Risø, Roskilde, Denmark.
Telephone: (03) 35 51 01, ext. 334, telex: 5072.

Fi 25-204

CONTENTS

ISBN 87 550 0224 2

Formalisation of failure mode analysis for control systems

Chapter 1

INTRODUCTION

Some simple systems work continuously until there is a failure, and then stop. The quality of such systems can be measured by reliability - the probability that the system will perform its function for a specified period.

For complex control systems, the situation is not so simple. There are many ways the system can fail, many recovery actions, and many different consequences of failure. Failure mode analysis is a way of judging the performance of such systems, by tracing the sequence of events following each failure, or each group of failures. It is useful, because it can isolate those cases where the probability of failure is high, or those where the probability is low, but the consequence is serious. During design, failure mode analysis can help to pinpoint those areas where design changes are necessary.

Various diagramatic aids are available (see e.g. Nielsen and Runge 1973), which help record the sequence of events, and the probabilities of different conditions, in a system prone to failure. Diagramatic aids are a great help, because one of the problems of failure mode analysis of a complex system, is to imagine all of the different things which could happen in a system, to decide which is most important, and to record them.

For very complex systems, such as those involving computers, it is difficult, even with the aid of diagrams, to keep track of all the information involved in a failure mode analysis. Such a system may have many thousands of components, each with a low failure rate. Some will be more critical to system performance than others. The analysis of such a system is very time consuming, and may be impossible for practical purposes.

By formalising the task of failure mode analysis, the difficulties become more clear, and some techniques for reducing the difficulties appear. Hopefully, formalisation will allow some of the tasks of failure mode analysis to be automated, as has been done for electronic logic systems (see e.g. Chang et alia 1970)

The first step in formalising analysis of failure event sequences, is to define what is meant by a system. The description should be sufficiently powerful to include all the components which are encountered in realistic systems (valves, computers, people). This task is dealt with in chapter 2.

The next step is to define what is meant by a failure event, and to explain

how the consequences of an event are calculated. This involves describing a system as a set of interconnected components; and providing a method for deducing the consequences within or at the output of a component, when an event occurs at the input of the component. The logic of cause and effect in a control system is treated in chapter 3.

In chapter 4, these ideas are applied to sequences of events, taking place in chains of system components. In chapter 5, the practical consequences of these ideas are considered.

Notation, and the mathematical background for the methods used is given in appendix 1. Appendix 2 describes the deduction methods used for event sequences. Appendix 3 describes the methods for calculating event probability distributions.

Chapter 2

SYSTEM DESCRIPTIONS

Formal descriptions of systems

Failure mode analysis involves describing what happens within a system under failure conditions. Such a description could take the form of a mental image of what happens, or a text description of the sequence of events in the system, or a diagram showing the sequence of events and progress of continuous changes. These descriptions are informal, where they are derived from a mental picture of the system.

A formal description of a system is a description written according to certain precise rules, and with corresponding precise rules for manipulation and use. For purposes of failure mode analysis, the two kinds of model are complementary. The formal description may be more detailed, or less detailed, than an informal description. But the consequences of the formal description can be evaluated systematically, and the degree of completeness of the evaluation can be measured.

Control systems involve many complex components such as computers, switching circuits, amplifiers, and devices such as motors and turbines, which are not usually described by one common theory. General systems theory (Windeknecht 1971) does provide a tool for describing such varied components, and has been used here. Appendix 1 provides an abbreviated version of the theory.

General systems theory enables a system to be described as a collection

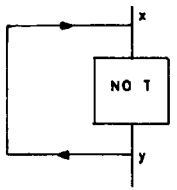
of interconnected 'black boxes'. The interconnections are called inputs and outputs, each input and each output is a (possibly vector) function of time. Different time bases (time sets) can be used to describe different systems. This is important, because some systems, such as switching networks, are best described in terms of a discrete time base, others, such as motors, in terms of continuous time.

A system description provides information about the relationship between input and output functions of time. Any system can be described by specifying the set of possible input functions of time, the set of possible output functions of time, and a mapping between them. In general, there will be an additional (vector) parameter, the state of the system at some particular time, before the mapping becomes completely specified. This is a requirement; for a particular input function of time and a particular system state, there should be only one output function of time.

A system description is 'complete', if it allows the output function of a system to be determined uniquely, given an input function of time. For many purposes, it is sufficient to have a partial description of a system, for example one which specifies the output for those inputs normally met in practice.

A system description is 'consistent' if for each particular state of the system at each point in time, every subsequent input function results in only one output function. All system descriptions should be consistent.

If a system description is inconsistent, it may mean that a mistake has been made in writing it down. Alternatively, it may mean that the description is an inadequate model of reality. For example the 'not gate' shown in fig. 1 is connected from output to input. The description alongside the 'not gate' is adequate for many purposes, but not in the configuration shown. The description leads to a contradiction because the switching delay of the 'not gate' has not been described.



Not gate description

Input = x, x(t) = 0 or x(t) = 1
Output = y, y(t) = 0 or y(t) = 1

x(t) = 0 implies y(t) = 1
x(t) = 1 implies y(t) = 0

Inter connection description

x(t) = y(t)

x(t) = 0 implies y(t) = 1
x(t) = y(t)
x(t) = 0 implies x(t) = 1

Fig. 1 Inconsistent system description

Such a trivial fault as this would be detected easily in most design procedures. For more complex examples, checking for consistency provides a way of detecting some types of design errors, and modelling errors.

An example

An example is given here to show how formal notation can be used to describe simple systems. The system is chosen to be as simple as possible, while still providing an interesting failure mode analysis problem. Just the system itself is described here, analysis will be treated later. Fig. 2 shows a system for supplying compressed air for pneumatic operation of machinery. The demand is fluctuating and intermittent, but requires a reasonably constant pressure. Hence a compressor with limited capacity is provided, and a reservoir tank is provided to smooth pressure fluctuations, and accomadate peak demand. For safety reasons, both the peak pressure and the minimum pressure in the system must be limited, and sensors are provided for safety reasons.

The example is typical of many installations, except that greater emphasis has been placed on safety than is usual.

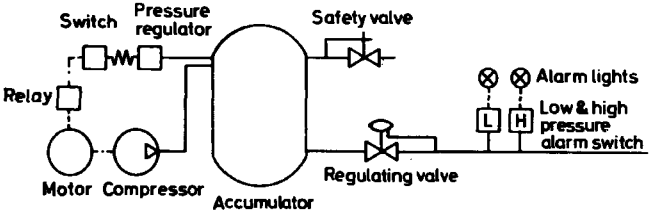


Fig.1. Schematic diagram of compressed air supply system.

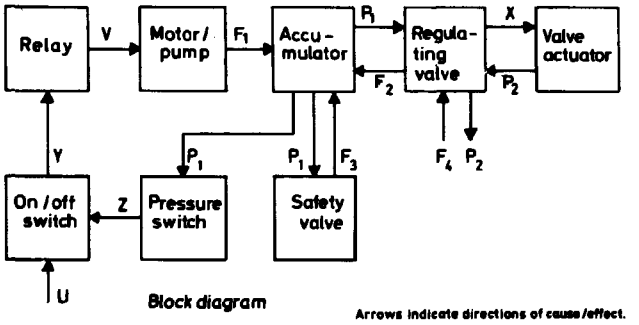


Fig. 2 A simplified compressed air supply system

The input and output functions of the various components in the block diagram are as follows.

F_1	air flow from compressor into accumulator	Kg m/s
F_2	air flow from accumulator to regulating valve	Kg m/s
F_3	air flow from safety valve	Kg m/s
F_4	air flow from regulating valve to supply lines	Kg m/s
P_1	air pressure in accumulator	Kg/cm^2
P_2	air pressure in supply lines	Kg/cm^2
V	motor power supply voltage	Volt
x	regulating valve position	cm
z	pressure regulator switch signal	off/on
u	on-off switch position	off/on
y	on-off switch output	off/on
K_1-K_{16}	constants	

The various components, when working properly, can be described by

motor/compressor

$$F_1(t) = K_1\ V(t) \qquad 1$$

relay

$$y(t) = 0 \ \underline{\text{implies}}\ V(t) = 0$$
$$y(t) = 1 \ \underline{\text{implies}}\ V(t) = K_{14} \qquad 2$$

pressure switch

$$\underline{\text{if}}\ P_1(t) < K_2 \ \underline{\text{then}}\ z(t) = 1$$
$$\underline{\text{else}}\ z(t) = 0 \qquad 3$$

on off switch

$$\underline{\text{if}}\ u(t) = \text{'on'} \ \underline{\text{then}}\ y(t) = z(t)$$
$$\underline{\text{else}}\ y(t) = 0 \qquad 4$$

accumulator

$$P_1(t) = K_3 \int_{t_o}^{t} (F_1(\tau) - F_2(\tau) - F_3(\tau))d\tau + P_1(t_o) \qquad 5$$

safety valve

valve closed $\underline{\text{if}}\ P_1(t) < K_4 \ \underline{\text{then}}\ F_3(t) = 0$

valve open $\underline{\text{else}}\ F_3(t) = K_5 \sqrt{P_1(t)}$

regulating valve

$$\underline{\text{if}}\ x(t) > 0 \ \underline{\text{then}}\ P_2(t) = P_1(t) - K_6 \frac{F_2(t)^2}{x(t)}$$

$$\underline{\text{and}}\ F_2(t) = F_4(t)$$
$$\underline{\text{if}}\ x(t) = 0 \ \underline{\text{then}}\ F_2(t) = 0 \qquad 7$$

valve actuator

valve closed: $\underline{\text{if}}\ P_2(t) \geq K_7/K_8 \ \underline{\text{then}}\ x(t) = 0$

normal range: $\underline{\text{if}}\ K_{12} > P_2(t) \geq K_7/K_8 \ \underline{\text{then}}\ x(t) = K_7 - K_8\ P_2(t)$

fully open: $\underline{\text{if}}\ P_2(t) > K_{12} \ \underline{\text{then}}\ x(t) = K_{15} = K_7 - K_8 - K_{12}$ 8

The descriptions given here are much simplified and are only valid within the normal operating range of the plant. The notation used is a mixture of arithmetic and logical symbols. Automatic manipulation of the descriptions is possible.

The statement which describes the relationship between input and output of a system is called the system predicate.

Classification of systems

It is possible to classify systems according to the mathematical properties of their descriptions. Two of these classifications are important here because they require a different treatment in failure mode analysis.

A memoryless system is one in which the output at time t is determined uniquely by the value of the input at time t. Many control system components are usually described as memoryless systems e.g. amplifiers. Memoryless systems have only one possible value for their 'state'.

Memoryless components are especially easy to treat in failure mode analysis because, in evaluating the effect of an event on the component, only the effect on output need be considered. There will be no effect on their state. Also, the effect of the event on output will be a function of the current value of input alone, and not of the past history of the input.

Very simple descriptions of memoryless systems can be given in standard form of an equation, giving the output of the system at time t explicitly in terms of the system inputs at time t.

Time dependent systems are those in which the output at time t is a function of input at time t, of the state of the system, and of the time t. In effect, value of time becomes an extra parameter of the state of the system. A classic example of a time dependent system is an alarm clock. But any system which ages noticeably with time, that is, has performance parameters

which change with time, is most conveniently described as a time dependent process.

Time dependent systems are difficult to treat in failure mode analysis, because not just the effect of a single failure event needs to be considered, but all the different effects that the event can have, corresponding to the different times at which the event can occur.

Cause and effect

Failure mode analysis is a technique which enables initial failure events, that is, causes, and their subsequent effects, to be studied. To this end, the models to be used in failure mode analysis must be cause-effect models.

The system descriptions provided by general system theory fulfill one of the basic requirements for a cause-effect model of a system. Cause must follow effect in time. General system descriptions must satisfy the condition that output at any time is a function of previous state, previous input, and time itself. Thus any change in output must follow a corresponding change in input, or a change in the system itself. (these are changes in the model of input and output. In a real system, effect always follows cause, by definition).

A causal model of a system also requires that the direction of cause and effect be determined, that is, inputs and outputs be distinguished clearly. This is not so straightforward, as many systems may work in either direction. For example, a d.c. rotating machine may be treated as either a motor or a generator, and the shaft torque may be considered to be either input or output.

Some system models have an inherent direction of causality, since one output may correspond to several different inputs. For example, a relay has many different values of coil current (input) but only two values of contact resistance (output). Descriptions in which there is a one to one correspondence between input and output functions of time are called 'bifunctional' or 'one to one'. The direction of flow of cause and effect is not determined if a system has a bifunctional description, and must be specified separately.

All energy flow systems are inherently bidirectional. Taking a 'force' P, a 'flow' F, and an 'impedance' R, then P = FR. Either force or flow can be considered a cause, but if one is a cause, the other necessarily is an effect.

Generally a variable in a system is considered to be a cause, if small changes in its value correspond to large changes in other variables; and if large changes in other variables correspond to small changes in its value. For example, varying load on an amplifier hardly affects the input signal, relative to its normal range of variation. But changes in the input signal cause wide variations in output.

The energy flow aspects of information and control systems are usually neglected, and design is such that the direction of cause and effect is clear. For those parts of a system involving important energy or mass flows, it is desirable to have a method of assigning directions of causality, as far as is possible.

A component of a system may be described in terms of a set of variables, without designating any of the variables as input or output variables. If the system is potentially bidirectional, than it is possible to describe the system by means of one or more equations (possibly implicit). For each equation, there will be one dependent variable and one or more independent (cause) variables.

System described by equations can be represented in the form of a graph, with two types of nodes - variable nodes and equation nodes. A variable node is connected by an arc to an equation node, if the variable appears in the corresponding equation. Fig. 3 shows the compressed air supply system expressed in this way.

Initially a graph constructed in this way is undirected, and assigning causal directions to the graph turns it into a directed graph.

First, directions associated with control components such as relays and amplifiers are marked on the graph. Then causal directions may be assigned according to the rule that each equation may serve to determine only one variable, and hence must have just one are leaving it. Similarly each variable must be determined by just one equation.

These rules serve to assign directions in all cases except those in which there is a loop in the graph, with each equation in the group being bidirectional (The associated system is bifunctional). In these cases, the direction of causality could be considered arbitrary.

Loops in cause-effect graphs

Given a set of equations which specify a graph with a single loop, there are two possible ways of assigning causality to the loop. (Fig. 4). A rule which can be used for assigning causal directions, is to make the assignment in the direction of 'greatest sensitivity'. This means, taking fig. 4 for example, that causal directions are first of all assigned in both possible ways, ACB and BCA. Taking the case ACB, the loop is broken after variable B, and the variation of A for a given variation in x is determined. Then the variation in A for a given variation in x is determined, with the loop closed. The ratio of the two variations is taken. The ratio is also determined for B, with the loop broken after variable A. The variable for which the ratio is greatest becomes

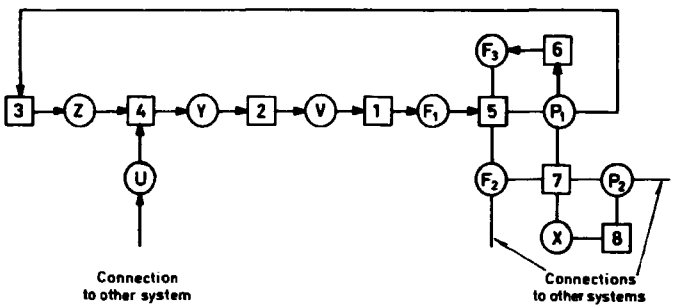


Fig. 3 Variable relationship graph for compressed air supply system

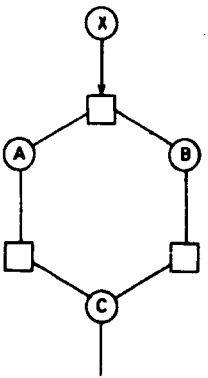


Fig. 4 Loop in variable relationship graph

the first in the causality chain.

This method of assigning causality can be applied to linear continuous systems (where the ratios in the example become

$$\left(\frac{\partial A / \partial x \mid B \text{ const}}{dA/dX} \;,\; \frac{\partial B / \partial x \mid A \text{ constant}}{dB/dx} \right)$$

It can also be applied to other types of system, provided that the concept of variation is defined appropriately. The method can be generalized to multivariable, multiloop systems of equations (Bristol 1965).

For cause effect analysis, the way of assigning causality described above will give an approximation to the effect of a cause on the variables in the loop. If cause-effect analysis is iterated around the loop, the iteration will converge, to give closer approximations to the true effects.

A special case of this method is to assign causality in the direction of integration, rather than differentiation, when there is a choice. In this way, a step function 'cause' results in 'effect' variations which are finite. In many cases, the method of assigning causality described above will give no clear cut answer, because the ratios of variations involved are of similar magnitude. In these cases, the concept of causality will also be of little use - the 'cause' results in a reaction 'effect' almost as strong as the cause itself.

In such a case, it is better to regard the variables involved in the loop as a subsystem in their own right, and to consider just the inputs and outputs of the loop as being causally determined. On meeting such a system in failure mode analysis, the only available method is to rely on 'solving the balance equations of the loop', without appealing to cause and effect.

Careful attention to the logical basis for assigning cause effect relationships is important in failure mode analysis, because very often models of a system are appropriate only over their normal working range. In failure situations, a model may cease to be appropriate, and even the direction of causality may change. For example, for a pump with output at atmospheric pressure, flow is considered a dependent variable. If the output is now shut off by a valve at the pump outlet, flow is fixed, and pump output pressure becomes a dependent variable

Chapter 3

CAUSE AND CONSEQUENCE

Cause - Consequence analysis

Cause - consequence analysis (D.S. Nielsen 1973) is a technique for tracing all the conditions which can lead to a failure event; and for tracing the consequences of that event through the various components of the system. A method is needed for representing not only the system itself, but also for representing conditions and events. These will be used for deducing the effect of an event at the input to a system component, on the state of the component itself, and on the output of the system.

A condition can be described by a statement, or predicate. This statement gives the value of some function of time, over a period of time. Alternatively, the statement may not describe the time function completely, but only some properties of the function which restrict the range of possible functions. For example statements that a function is constant over a period of time; that the value of a function is equal to a given value over a period; or that a function is monotonically increasing over a period; or that the function is less than a given value; all of these constitute descriptions of conditions. Such statements can be expressed formally, using the notation of logic and of set theory. The condition takes the form of a predicate, with a span of time as a parameter.

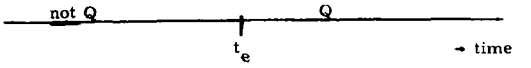
Events too can be expressed by means of a predicate, similar in form to the description of a condition. The event description is associated with a point in time, and the meaning is that before the event time, the predicate is untrue. At the event time, the event predicate becomes true. (see fig. 5).

As an example of an event description, the description of the event in which the pressure in the compressed air reservoir increases over the safety valve opening pressure is

$$\underline{\text{not}}\ P_1(t) \rangle K_4 \quad \rightarrow \quad P_1(t) \rangle K_4$$

This statement is not a complete description of an event in logical terms, but may be regarded as an abbreviation of the form given later in this chapter.

General scheme for events:



Q is event predicate

example of an event description:

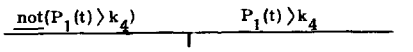


Fig. 5 Scheme for event descriptions, and an example

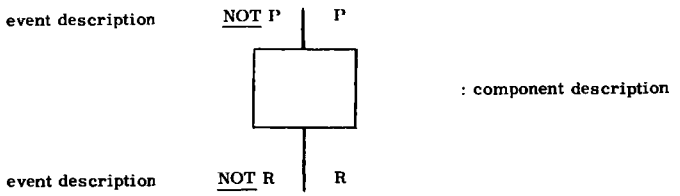
Deducing the consequences of an event

The simplest type of component which can be treated by failure mode analysis is one with a memoryless, time independent description, a single input, and a single output.

If an event occurs at the input to such a component, then an event may occur at the output of the component. The description of such an event can be obtained by a process of deduction.

An example of the deduction process is provided by the safety valve of the compressed air system. The event at the input to the valve is that in which air pressure rises above valve trip pressure. The event at the output of the valve component (Fig. 2) is that the flow through the valve ceases to be zero.

Figure 6b shows this example, and it seems an excessive amount of description for a conclusion which is almost obvious. The important point is that the output event description can be deduced according to a standard procedure. If necessary, this procedure can be automated, using a computer.



NOT P and Q implies not R

P and Q implies R

Fig. 6a Simple event deduction across a component

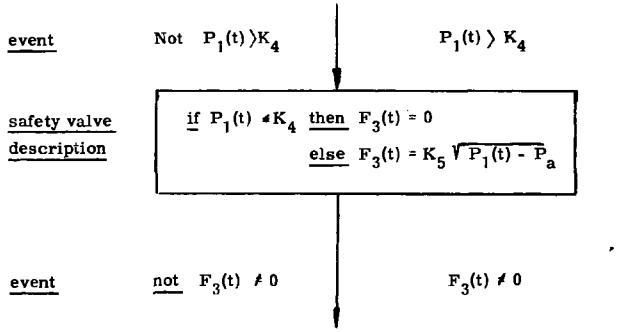


Fig. 6b Example

From the example, it can be seen that there are several possible output event descriptions, for example

E: $F_3(t) = 0 \rightarrow$ not $F_3(t) = 0$

E: not $F_3(t) = K_5\sqrt{P_1(t)} \rightarrow F_3(t) = K_5\sqrt{P_1(t)}$

Which is the correct output event description ?

The requirements for an output event description are that as much information as possible should be retained, concerning the output variables; that no unnecessary information concerning the input variables should be retained; and the output event description should not be redundant.

These requirements are satisfied by taking first of all the event description, and the system description and reducing them to a simplified standard form. Several different forms are possible, depending on the kind of notation used, but for simple logical notation, a form known as conjunctive normal form is used (see e.g. Nilsson 1971).

The result of a transformation to conjunctive normal form is an expression consisting of 'clauses', linked by 'and' operators. Each term consists of a comparison, an equality, or the negation of a comparison or equality. The result of transferring the safety valve example to conjunctive normal form is shown in fig. 7.

The next step is to draw as general conclusions as possible from the statements, and this can be done using a process called 'resolution' (Robinson 1965). If there is an expression of the form

(A or B or C)
and (F or G or (not C))

then an additional clause is added of the form

and (A or B or F or G)

This process is repeated (in its most general form) as often as possible (see fig. 8).

The next step is to remove as much redundant information as possible. This means that if there is an expression such as A and (A or B) , the clause (A or B) is deleted. The reasoning here is that information represented by (A or B) is uncertain, may be contradicted by more certain information in other clauses, and in any case does not indicate the certainty of occurrence of an event. The process of deleting clauses is called subsumption.

Finally, any clauses dealing with input variables alone may be deleted, as irrelevant.

The result of this process is a single logical expression giving the conditions at the output of a component at the time of an event. The process is repeated, with the negation of the input event description, to obtain the description of the conditions prior to the event.

The only remaining step is to ascertain whether any change is involved at the output as a result of the input event. If the description of conditions

Definitons

event: $\longrightarrow P_1(t) \rangle K_4$

safety valve: if $P_1(t) \leq K_4$ then $F_3(t) = 0$
else $F_3(t) = \sqrt{P_1(t)}$

Conjunctive normal form

1 not $P_1(t) \langle K_4$
2 and (not $P_1(t) = K_4$)
3 and ((not $P_1(t)$ K_4) or $F_3(t) = 0$
4 and ((not $P_1(t) = K_4$) or $F_3(t) = 0$
5 and $(P_1(t) \langle K_4$ or $P_1(t) = K_4$ or $F_3(t) = \sqrt{P_1(t)}$

Fig. 7 Conjunctive normal form for statement of safety valve trip event (conditions after event)

Resolution between 1 and 5

6 and $(P_1(t) = K_4$ or $F_3(t) = \sqrt{P_1(t)})$

Resolution between 2 and 6

7 and $F_3(t) = \sqrt{P_1(t)}$

Delete clauses 3, 4, 5, 6 by subsumption
Delete clauses 1 and 2 - do not affect F_3

Which leaves

$$F_3(t) = \sqrt{P_1(t)}$$

Fig. 8 Deduction of an output event description (conditions after event)

after the input event is the same as the description of the conditions before the input event, then there is no output event, since there has been no change in output conditions. Similarly, if output condition after the event time is a logical consequence of the output condition before the event time, then no output event can be recorded (no new information about the output conditions has been produced).

Whether the second condition is a logical consequence of the first can be decided by trying to deduce the second from the first, possibly using an automatic theorem prover (see e.g. Robinson 1965).

As the example shows, the correct output event description for the safety valve is

$$E: F_3(t) = 0 \longrightarrow F_3(t) = K_5 \sqrt{P_1(t)}$$

Formal description of events

For memoryless control system components, the effect of an event at the input to the component will be an immediate event at the output of the component, or no output event at all. For a component with memory, there may be output events which are delayed, and there may also be a change in state within the systems. Also, whether an output event occurs will generally depend on the state of the system at the time of the input event.

For these reasons, the event descriptions must be accompanied by a statement of the time at which the event occurs, and a more general form of event description is required. Also some more formal idea of what constitutes a condition and what constitutes the deduction of an event, is required. No simple definition can be given of what constitutes a condition description, but an idea of the range of what is possible is given by the following set of examples

' condition P holds for function f during period t_1 to t_2'
$\equiv$ holds (P, t_1, t_2, f)
$\equiv P'(t_1, t_2, f)$

At this point, there are many form the condition description could take. For example, let T be a set of instants of time. Then the following is a condition

for all t, t is a T and $t_1 \leq t \leq t_2$ implies $P''(t, f)$

Another possible form is

for all t_a, t_b, t_a is a T and t_b is a T and $t_1 \leq t_a \langle t_b \leq t_2$
implies $f(t_a) \leq f(t_b)$

In general, if $P(t_1, t_2, f)$ is a condition description, then

$$\underline{\text{for all}}\ t_a, t_b, [\, t_a\ \underline{\text{isa}}\ T\ \underline{\text{and}}\ t_b\ \underline{\text{isa}}\ T\ \underline{\text{and}}\ t_1 \leq t_a < t_b < t_2\,]\ \underline{\text{implies}}\ P(t_a, t_b)$$

This constitutes a test to discover if a statement is a condition description.

Given a definition of what constitutes a condition, an event can be defined. The idea is given by the following set of equivalences

'event E occurs for function f at time t'

≡ occurs (E, t, f)

≡ E' (t, f)

≡ E'' (P, (t_1, t, f), $P_2(t, t_2, f)$)

≡ there is a t_1 such that $P_1(t_1, t, f)$
and there isa t_2 such that $P_2(t, t_2, f)$
and not $(P_1(t_a, t_b, f)$ implies $P_2(t_a, t_b, f)$

where P_1 and P_2 are condition descriptions

An event occurs at time t if an expression of this form can be deduced from other true statements concerning the system.

Components with memory

A method is required which enables event descriptions to be deduced for components with memory. There are two major types of problem here. First, it is desirable for convenience that the deduction of an output event be obtained by considering only a single input event, rather than considering all possible sequences of one, two, events, etc. Secondly, it may be possible to deduce the occurrence of an output event at some time after the input event, but on the condition that nothing else happens at the input to counteract the (potential) output event.

A scheme for deduction of events accross a component with memory is given in figure 9.

The notation

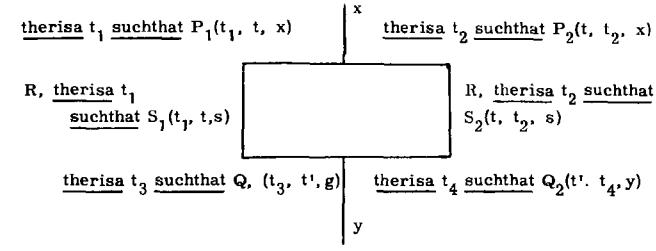
$$\frac{A,\ B}{C}$$

of figure 9. means that given statements A and B, statement C can be deduced.

The state condition S_2 can be deduced from the state condition S_1, the input conditions P_1 and P_2, and the component description R. By keeping track of the different state conditions during a sequence of events, state and output

changes can be deduced by combining information from a single event and the state, rather than making deductions from statements about a sequence of events. This will become especially significant in the next chapter.

The output conditions Q_2 can be deduced from the state condition S_1, the input conditions P_1 and P_2, and the component description R.



1 $\dfrac{(\text{therisa } t_1 \text{ suchthat } (P_1(t_1, t, x)) \text{ and } R \text{ and } (\text{therisa } t_1 \text{ suchthat } S_1(t_1, t, s))}{\text{therisa } t_3 \text{ suchthat } Q, (t_3, t', y)}$

2 $\dfrac{(\text{therisa } t_1 \text{ suchthat } S_1(t_1, t, s)) \text{ and } R \text{ and } (\text{therisa } t_2 \text{ suchthat } P_2(t, t_2, x))}{\text{therisa } t_2 \text{ suchthat } S_2(t, t_2, s)}$

3 $\dfrac{(\text{therisa } t_1 \text{ suchthat } S_1(t_1, t, s)) \text{ and } R \text{ and } (\text{therisa } t_2 \text{ suchthat } P_2(t, t_2, x))}{\text{therisa } t_4 \text{ suchthat } Q_2(t', t_4, y)}$

4 not $((\text{therisa } t_3 \text{ suchthat } Q_1(t_3, t', y))$ implies $(\text{therisa } t_4 \text{ suchthat } Q_2(t', t_4, y))$

P - input conditions Q - output conditions
R - Component description S - state conditions

Fig. 9 Scheme for deduction of an output event

In general, the output condition will take the form of a statement about y which is dependent on the initial state condition S_1, and on the length of time t to t_2 for which the input condition holds. Once again the rules of resolution (augmented with special rules for $\rangle$, $=$, etc.), subsumption and nonrelevance can be used to derive an output description. (Appendix 2).

Example

As an example of a deduction accross a component with memory, the pressure sensitive switch can be used. The description associated with fig. 2 is oversimplified. Such components usually incorporate hysteresis, to avoid over frequent switching of the air compressor. The description incorporating hysteresis is:

pressure switch

if $P_1(t) \langle K_9$ then $s(t) = 1$
and if $P_1(t) \rangle K_{10}$ then $s(t) = 2$
and if $(K_9 \leq P_1(t) \leq K_{10}$
and there isa t_1 such that $(t_1 \langle t$
and forall t', $t_1 \langle t' \langle t$ implies $s(t') = 1))$
then $s(t) = 1$

and if $(K_9 \leq P_1(t) \leq K_{10}$ and therisa t_1 such that
$(t_1 \langle t$ and forall t', $t_1 \langle t' \langle t$ implies $s(t') = 2))$

then $s(t) = 2$

and if $s(t) = 1$ then $z(t) = 1$
and if $s(t) = 2$ then $z(t) = 0$

The idea of this definition is to give the output of the pressure switch in terms of a state, S(t) and a condition on the input, $P_1(t)$. The description is somewhat unwieldy because of the difficulties in expressing facts about continuous functions in a simple system of logic. For general use, a more concise notation is desirable. This is used in figure 10, where the notation $s(t-)$ is used to represent the state of the system an instant before the event time t.

The event description deduced for the pressure switch is

$s(t-) = 2$ and if $P(t+) \langle K_9$ then $s(t+) = 1$

The event deduced is a state change event, and is dependent on certain conditions being fulfilled by the input function $P_1(t)$. To achieve this description a new rule was needed - there were two possibilities for the output, $s(t+) = 2$ or $s(t+) = 1$. One of these corresponded to an event, the other did not. Working backwards, the input conditions for an event to occur, were derived. The result is the description of a 'conditional event'.

Complete deduction methods

The methods used is deducing output events so far are resolution and subsumption. These methods apply to expressions involving 'forall', 'there is a', 'and', 'or', 'not', 'implies'. Extra rules must be used for handling 'greater than', 'is a' and 'equals' operators, and here, such rules have been used on an ad hoc basis.

Resolution and subsumption, when applied to expressions involving logical symbols only, are 'complete'. The result of applying resolution is to find all possible deductions of a certain form from an initial set of clauses. This means that the corresponding output event description is certain to be found.

For expressions involving numeric and set operators, such complete methods are not generally possible. However, incomplete methods will generally give satisfactory results, more efficiently.

Pressure switch

if $P_1(t) < K_9$ then s(t) = 1
and if $P_1(t) > K_{10}$ then s(t) = 2
and if ($K_9 \leq P_1(t) \leq K_{10}$ and s(t-) = 1) then s(t) = 1
and if ($K_9 \leq P_1(t) \leq K_{10}$ and s(t-) = 2) then s(t) = 2

event description

$P_1(t-) > K_{10}$
and $P_1(t+) < K_{10}$

Conjunctive normal form

not $P_1(t) < K_9$ or s(t) = 1 1

and $P_1(t) < K_{10}$ or $P_1(t) = K_{10}$ or s(t) = 2 2

and $P_1(t) < K_9$ or not $P_1(t) = K_{10}$ or not s(t-) = 1 or s(t) = 1 3

and $P_1(t)$ K_9 or not $P_1(t)$ K_{10} or not s(t-) = 1 or s(t) = 1 4

and $P_1(t)$ K_9 or not $P_1(t) = K_{10}$ or not s(t-) = 2 or s(t) = 2 5

and $P_1(t)$ K_9 or not $P_1(t)$ K_{10} or not s(t-) = 2 or s(t) = 2 6

and not P(t-) K_{10} 7

and not P(t-) = K_{10} 8

and P(t+) K_{10} 9

Resolve 7 and 2: and $P_1(t) = K_{10}$ or s(t-) = 2 10

Resolve 8 and 10: and s(t-) = 2 11

Resolve 9 and 6: and P(t+) K_9 or not s(t-) = 2 or s(t+) 12

Resolve 11 and 12: and P(t+) K_9 or s(t+) = 2 13

Resolve 13 and 1: and s(t+) = 1 or s(t+) = 2 14

11 and 14 together provide an event description, but a better form can be obtained by combining with 1

s(t-) = 2 and if $P_1(t+) < K_9$ then s(t+) = 1

Fig. 10 Deduction accross a component with memory - pressure switch example

Chapter 4

CAUSE - CONSEQUENCE ANALYSIS

Event chains

The methods described in the last chapter enable the consequences of a single event on a single component to be evaluated. Consequence analysis is a method of tracing through networks of components, calculating output events for one component, and then treating these as input events for the next components in the network. In this way, branching chains of events can be recorded. (Fig. 11). The start of such a chain of events will be a simple initial event in the normal operation of a piece of equipment; or a failure event.

The occurrence of some events will depend on the conditions prevailing within a component, or within the rest of the system, when the event occurs. In some cases, complete 'trees' of coincident conditions must be built up, in order to analyse under which conditions a particular event can take place. The process of building these trees is called 'cause analysis'.

In failure mode analysis, one is interested in discovering if there are any conditions of normal operation which can cause a failure event; and in finding which failure events lead to further serious damage events.

Failure events may usefully be divided into three classes. 'Spontaneous failure events' occur as a result of no recognised cause, on a statistical basis. 'Situation induced failure events' occur as a result of chance coincidence of otherwise normal conditions within a system. 'Cascade failure events' occur as the result of other previous failures.

For failure mode analysis, we need a model of all the components in a system, under normal operating conditions. A description is also required of the conditions under which the model is accurate, and the conditions under which another, failure model is required.

A model is also required to express the seriousness or cost, of damage to a component. For example, for the compressed air system, the compressed air reservoir dangers can be represented by

if Pi(t) $2 \times K_4$ then cost is high

The objective of a failure analysis will be to find those possible failure events with a high cost.

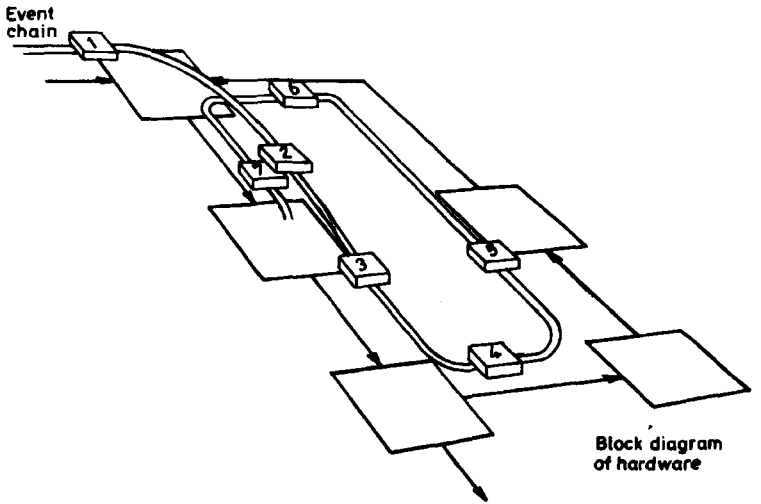
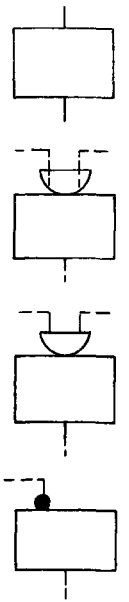


Fig. 11. 'Unwinding' an event sequence chain from a hardware block diagram.

Tracing event chains

Formal methods of cause - consequence analysis can be regarded as translating from a block diagramdescription of a system to an event sequence diagram, by a process of 'unwinding'. A notation for event sequence diagrams is given in figure 12 (Probability rules for each box are also given, see appendix 3).

The simplest case of event tracing is for a block diagram with a set of components connected in series (fig. 13). For each possible initial event (spontaneous or situation induced) there will be a simple chain of events. However, if there are any components with memory, then the event chain may branch, depending on the state of the component. At any point an event chain may simply stop, because there are no further events to be observed.



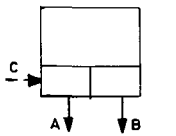
event description box

or box for conditions

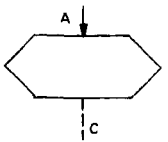
and box for conditions

NOT BOX FOR CONDITIONS

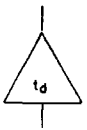
Fig. 12.



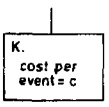
event A or event B occurs depending on condition C



condition C is the condition that event A has happened



delay involved in passing from one event to another



expression for cost of a failure event.

Fig. 12. Continued

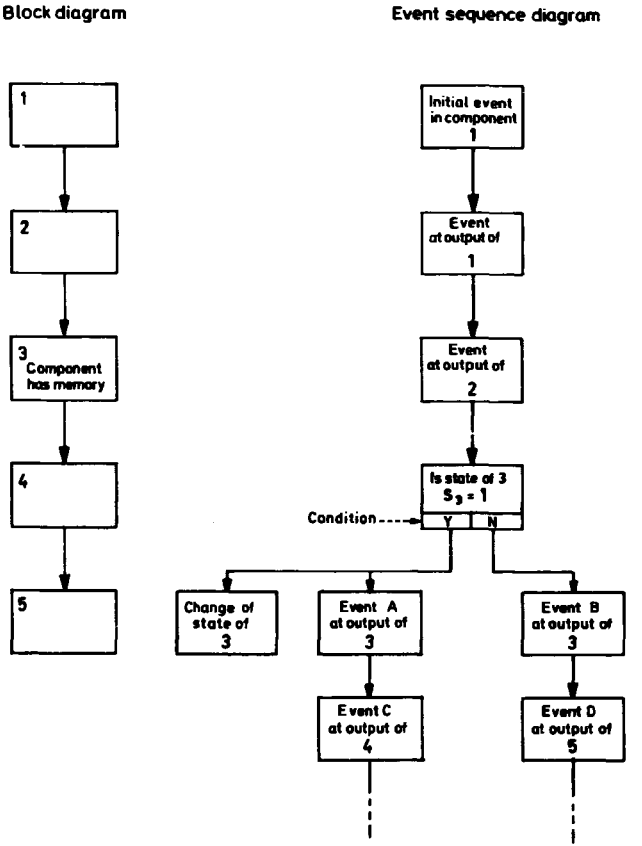


Fig. 13. Event tracing in series connected components.

Other cases important in event tracing are given in figure 14. If a component has two output connections, it will give rise to a fork in the event sequence diagram and to two subsequent event chains.

If a component has two input connections, then the consequences of an event at one input connection will depend on the conditions at the other connection. The result is a conditional fork in the event sequence diagram, and also a systematic evaluation of the conditions may be required.

Feedback loops in a block diagram can result in long sequences of events, involving some delay between events. The sequence may continue indefinitely (that is, until some other spontaneous event interrupts the sequence) or the system may reach a stable state in which no further events occur.

If a loop in a block diagram involves components with memory, especial care is needed. Several 'delayed event' chain can occur as the result of one input event. If the effect of the first of these event chains is evaluated, it may lead to the conclusion that the conditions for the other event chains are not maintained. The consequences of the later 'delayed events' may not need to be evaluated.

Multiple failures

One of the most useful aspects of cause - consequence analysis is the help it provides in evaluating the consequences of multiple failures. At a formal level, there are two practical aspects of multiple failure problems - firstly, in a cascade of failure events which influence each other, the relative timing of the events is important - and secondly, a failure event may not lead to subsequent damage events as a direct result, but only to a change of state of a component, an 'unrevealed fault', which may make its presence felt under later changed conditions.

The problem of relative timing of events in several 'parallel' chains of events, can be solved by first evaluating the chains of events, tentatively - and then considering whether the different chains can represent reality by considering the logical consequences of different event timings.

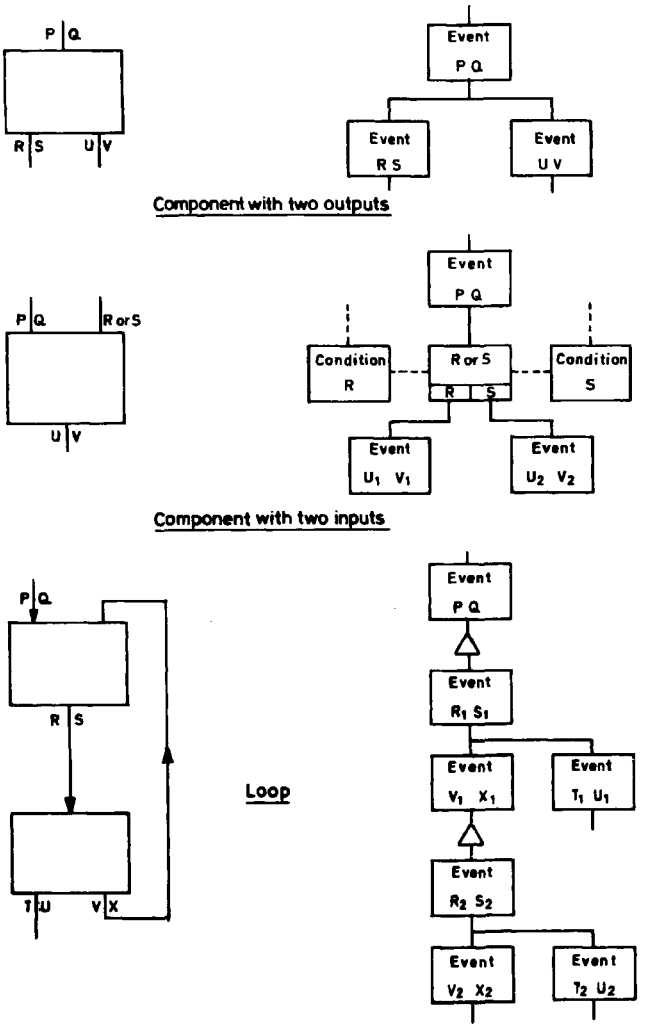


Fig. 14. Block diagrams and event sequence diagrams.

The second problem, of unrevealed faults, is solved by recording any possible state changes, or permanent changes in component model, which are possible as a result of some initial event. Then subsequent event chain evaluations must take into account not only the different normal conditions for the component, but also the possible failure conditions. An iterative process is required to provide a complete analysis of all unrevealed faults.

An example of systematic failure mode analysis

The example of the compressed air supply system will be used to illustrate systematic failure mode analysis. The analysis provided here is not thorough, since it involves only two failures, in the relay, and in the safety valve. An attempt has been made to follow a systematic algorithm for the analysis (fig. 15) but the needs of presentation require the use of some heuristic rules.

1	Record initial conditions for system and set time $t_i = t_o$
2	Select initial event, call it A
3	Apply event A to relevant system component X
4	Deduce changes of state, if any, and record them
5	Deduce different events B, C, ... on different outputs of X, and at different times (on some output possibly).
6	If there were no events B, C ..... then take an event from an unfinished event chain, call it A, and go to 3.
7	If there are no unfinished event chains, go to 1. If there are no more initial events, stop.
8	Select the output event which occurs first on B, C ..... at t_j and call it F.
9	Check event F to ensure that it is consistent with conditions at time t_i - if not, delet F from B, C ..... and go to 7
10	Check other events on B, C ..... to see if they are compatible with F if not, delete them. If they are, record them as unfinished event chains.
11	Record F on the event sequence diagram. Update conditions to time t_j. Replace t_i by t_j, event A by event F.
12	Go to 3.

Fig. 15 Algorithm for consequence tracing

Note: this algorithm does not cover all situations, but does cover those situations met in the example of fig. 16.

The description of the system given with fig. 2 needs to be augmented, with information about the values of the constants involved, with information about the initial state of the system, and with information about load on the system.

The additional information required is provided by the following statements.

load if $P_2(t) > K_{11}$ then $F_4(t) > 0$
else $F_4(t) = 0$;

10

The constant values obey the following relations constants

$$K_4 > K_{10} > K_{11} > K_9 > K_{12}$$

where K_4 is safety valve trip pressure
K_{10} is pressure control switch upper trip pressure.
K_{11} is air supply minium useable pressure.
K_{12} is pressure at which regulating valve opens fully.
K_9 is pressure control switch lower trip pressure.

damage

if $P_1(t) > K_4 \times 1.5$ then cost is high

initial state

$P_i(t_o) = P_2(t_o) = F_1(t_o) = F_2(t_o) = F_3(t_o) = F_4(t) = 0$
$x(t) = K_{15}$

initial event

u(t) = 'off' → u(t) = 'on'

Any automatic process will experience some difficulty in making deductions about a set of non linear equations. For this reason, the model of the regulating valve and actuator need to be reformulated. This can be seen, by deducing from statements 7 and 8

$$\text{if } K_{12} < P_2(t) < K_7/K_8 \text{ then } P_2(t) = P_1(t) - \frac{K_6 (F_2(t))^2}{(K_7 - K_8(P_2(t))^2}$$

$$\text{if } P_2(t) \geq K_7/K_8 \text{ then } (x = 0 \text{ and } F_2(t) = 0)$$

$$\text{if } P_2(t) < K_{12} \text{ then } P_2(t) = P_1(t) - K_{16}(F_2(t))^2$$

The problem is caused by the implicit equation in the first line, which may be replaced by

$$\text{if } K_{12} < P_2(t) < K_7/K_8 \text{ then } (P_2(t) = g_1(P_1(t) - g_2(F_2(t))$$
$$\text{and } P_2(t) \neq P_1(t)$$
$$\text{and } (F_2(t) = 0 \text{ implies } g_2(F_2(t)) = 0))$$

If the regulating valve is within its normal working range, the valve actuator component may then be ignored.

The first task in the failure mode analysis is to investigate the normal

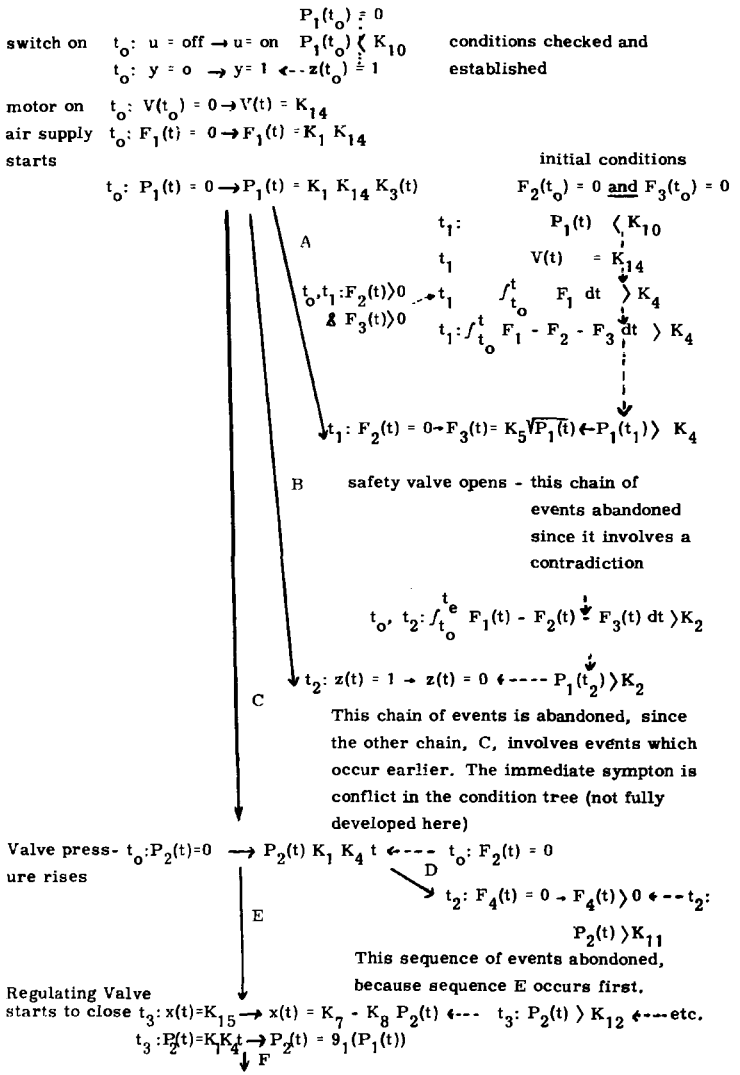


Fig. 16 Normal operation of compressed air system

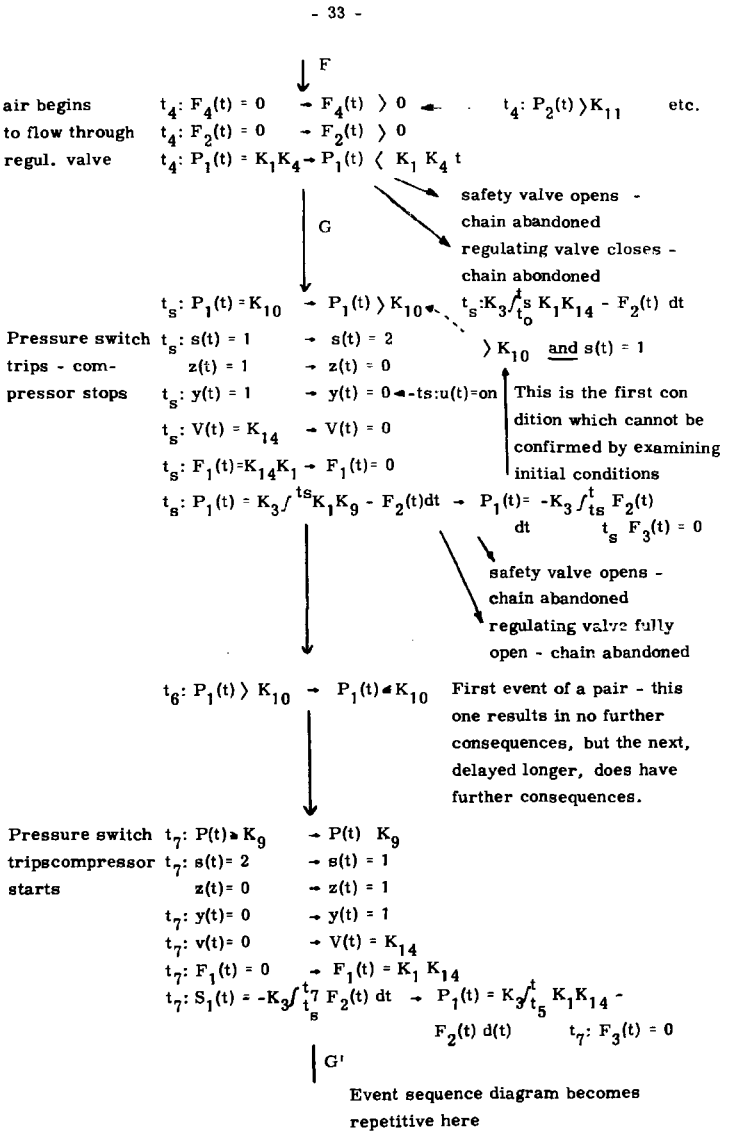


Fig. 16. Continued

operation of the system. The event sequence diagram for this is shown in fig. 16. The notation used is as follows

event t_i: $P_1 \rightarrow P_2$

condition at time t_j t_j: P_3

condition between time t_j and time t_k t_j, t_k: P_4

Failure descriptions

Two types of failure event will be treated for the example of fig. 16. The first is failure of the safety valve by 'sticking', with the result that the valve does not open when it should.

The valve failure can be described as follows.

t_e: if $P_1(t) < K_4$ then $F_3(t) = 0$ else $F_3(t) = K_5 \sqrt{P_1(t)}$

$F_3(t) = 0$

Applying this event to the safety valve component, under all the normal conditions derived in fig. 16, there are no output events to record for $F_3(t)$, because $F_3(t) = 0$ under all normal conditions. However, the event should be recorded, in the same way that an event would be recorded for a change of state in a component with memory.

The second failure event to be described is the failure of the relay by constant welding. The event can be described by.

t_e: (Y(t) = 0 implies V(t) = 0) and (Y(t) = 1 implies V(t) = K_{14})

This yields the event sequence diagram shown in fig. 17.

Note that now, the effect of the unrevealed safety valve fault becomes apparent.

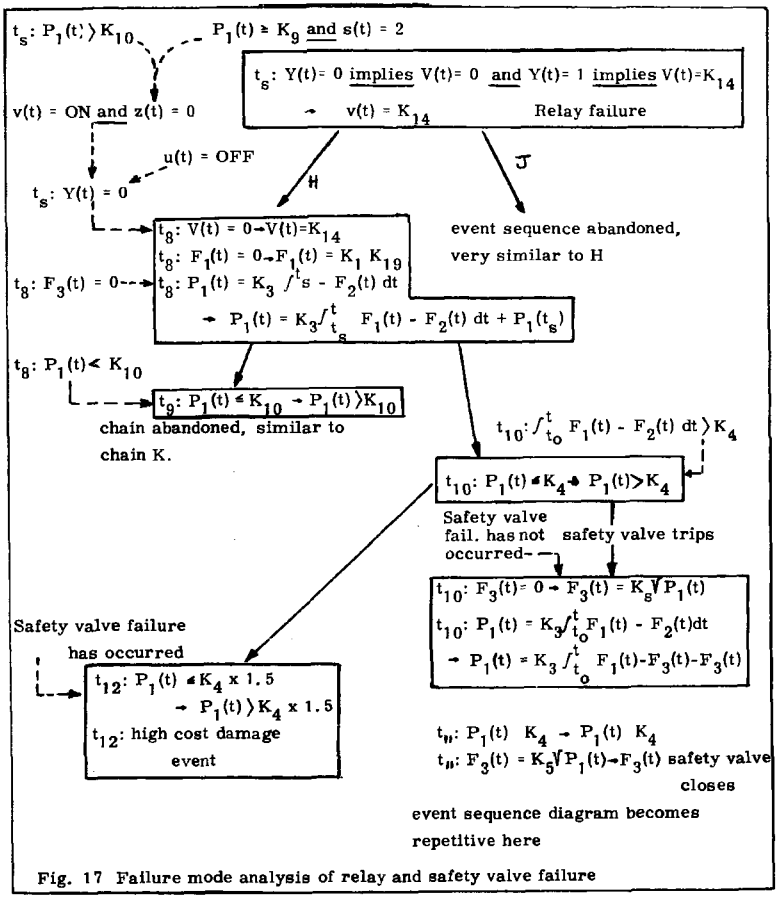


Fig. 17 Failure mode analysis of relay and safety valve failure

Chapter 5

PRACTICAL ASPECTS

Significance of formal methods

The techniques described in the previous chapters allow a rigorous meaning to be attached to the intuitive methods of failure mode analysis. The most important advantage is that a clear meaning can be given to thoroughness and completeness of a failure mode analysis. For example the statement 'An analysis has been made of all interactions between components, except interactions accross insulation and physical barriers' acquires meaning because it becomes possible to consider 'all interactions'.

Used as a tool in reliability oriented design, such methods are basically conservative. If all that is known about a component is the way that it works normally, and the fact that it can fail, then the formal methods provide a way of finding the 'worst' way in which it can fail. Unless information is forthcoming about the actual modes of failure in practice, a conservative policy is to provide protection against the effects of all possible failure modes. As more experience is gathered, more liberal design rules can be adopted.

The modelling techniques used here are particularly appropriate for engineering purpose. Very often, analogue modelling is constrained by the lack of data. Logical reasoning of the kind used here can often yield adequate results, and allow more accurate simulation methods to be used in just those areas where the economics of design are most serious. (For example, only relative sizes of constants for the compressed air supply system, not their numeric values, were needed).

The modelling techniques are also significant for the collection of failure data. The normal working of a component can be described, and its normal working range. Then failure properties can be described by a model, either as certain consequences of a situation or as events occuring under normal circumstances with a certain probability. Failures occuring in a situation (condition) and failures occuring during an event, can be clearly distinguished (for example, failure of solder connections depends on situation, usually. Failure of relay contacts usually occurs during switching). Again, more precise descriptions of failure modes allows more liberal design rules to be used.

It should be possible by using modelling techniques, to answer questions such as "can the sequence of events which happened 'there' also happen here' An interesting application would be to use accident reports in this fashion,

and extend the range of usefulness of case studies as far as possible to new situations.

The main problem with formal methods of failure analysis (or any form of failure analysis for that matter) is the cost. To achieve significant results for complex systems, with reasonably reliable components, requires a large amount of work. Use will generally be restricted to those areas where safety is involved. Formal methods have something to offer here. There is a possibility for at least semi-automation of the analysis.

There are three basic stages in failure mode analysis; modelling; cause consequence diagram construction; and mathematical analysis of event probabilities. The methods used here are applicable to the second step - constructing event sequence diagrams. Analysis of probabilities involves a further step. Some of the simpler rules involved are described in appendix 3.

Possibilities for automation of failure mode analysis

Comparing a modern control system with earlier examples, the modern system generally has more components, and these components are generally more reliable. The effect is that each failure mode has a lower probability. But the number of possible failure modes is much greater. A full analysis requires examination of a large number of unlikely circumstances.

Failure mode analysis itself is a partial answer to this problem. The technique described in chapter 4 can be further automated, using a computer.

The time taken to produce the diagram of fig. 16, by hand, was $1\frac{1}{2}$ man days. Most of this time was taken up in deducing events, and even more so in checking the conditions under which an event could take place. Procedures for performing such deductions on a computer exist (e.g. Robinson 1965).

Both automatic and manual procedures suffer a disadvantage - they cannot be guaranteed to produce an answer in a finite amount of time. This situation does not occur often in practice, and in any case is not an important problem for failure mode analysis. If a failure event is suspected to occur, but cannot be proved to occur, assume that it does occur. However, the problem means that human monitoring is required, because if event deduction takes too long, it usually means that an error has been made in modelling.

Further automation can help with recording information, and with plotting failure mode analysis. The technique can certainly not be automated completely, however. There is simply too much computation involved, and heuristic rules are needed to guide the analysis. These are best applied by a human being. Also, one of the prime effects of the analysis process is to refine the component models and to correct errors in them. Failure mode analysis is

best regarded as a method of helping an engineer to understand a system. As such, complete automation is meaningless.

Automation of parts of the failure mode analysis process is technically feasible. Whether such a step is worthwhile depends on how much of this kind of analysis is performed. Practical use would require use of interactive computing facilities, and the collection of a set of simple models for the common system components. It would also require some effort in improving the ease of understanding of the logical processes involved, and a better presentation of the component descriptions. Natural language translations of the logical expressions would be desirable.

More work is required in studying the individual steps in the deduction process. In particular, it is in principle possible for the event descriptions to become completely unwieldy, and for the deduction process to become very inefficient. The success in the examples chosen probably owes a lot to their simplicity.

More work is also required in studying the way heuristic rules are used to limit the size of the failure mode analysis task. A list of some of the heuristic rules observed during intuitive construction of cause-consequence diagrams (Nielsen 1972) is shown in fig. 18.

1	Having detected an event with a serious consequence, work backwards to find other event chains leading to the same consequence
2	Stop analysing an event chain, or a tree of condition combinations, when the probabilities involved become very low.
3	If a chain or group of components has a simple constant input output description for all event chains, it can be treated as a single component, to reduce effort. The new description is deduced from the old. This can be extended to a hierarchical structuring of a system, with reduced detail.
4	Standard situation combinations and event sequences can be recognized and stored for later use.
5	Analysis for a single fault of for a single direct and several unrevealed faults, is most useful (generally gives high probabilities)
6	Treat most frequent initial events first, and ignore low probability initial events.
7	For probability distribution analysis of states - recognise repetition in an event sequence, and restructure the event sequence as a loop. Then use techniques for modelling markoff processes.

Fig. 18 Heuristic rules used in simplifying failure mode analysis

Chapter 6

CONCLUSIONS

Some conclusions can be drawn from this study. On a theoretical level, the idea of formalizing failure mode analysis is primarily useful in that enables one to define what completeness means. A complete analysis is one which all possible sequences of events have been traced through a model. Different orders of completeness arise because one may take one, two, or more simultaneous failures into account. And any analysis is complete only with respect to a particular plant model (either a mental model or a formal model). A model will never be complete in explaining all possible features of plant behaviour, but it may be complete in explaining all observed forms of behaviour, or explaining a particular set of accident records for similar plant.

The idea of using component models as a basis for organising failure data collection is attractive. But the amount of work involved before a reasonably large set of data could be collected, is daunting. Some improvement in modelling procedure, over those used here, is required, if such work is to be made economic for complex systems.

The amount of computation (mental or automated) involved in producing a complete failure mode analysis (even with just a simple plant model) is seen to be very large. There is no doubt that engineers can produce qualitative analyses more cheaply than a computer system working alone. The main advantages from any automated approach would be in simpler data handling and presentation of results, and in enabling a greater level of confidence in the completeness of the analyses. (If a designer can make a logical error, so can an analyst).

Any automated procedure will require interaction between man and machine, if only to draw on the mans experience of modelling, and to correct modelling errors. Human aid in recognising failure patterns, and in redirecting analyses along more efficient paths, should also be of great help. The initial attempts at 'applying' automated procedures, using pencil and paper calculation, are encouraging. The amount of effort involved would be a trivial load for a computer.

REFERENCES

1) Chang, Manning, and Metze, Fault Diagnosis of Digital Systems, New York, Wiley 1970.

2) Nielsen. The Cause Consequence Diagram Methods as a Basis for Quantitative Accident Analysis. 1971.

3) Windeknecht, 'General Dynamical Processes' Academic press 1971. Risø-M-1374.

4) Bristol 1965 'On a new Measure of Interaction for Multivariable Process Control', IEEE Trans. Automatic Control, pp. 133-134, Jan. 1966.

5) Nillson 1971, Problem Solving Methods in Artificial Intelligence.

6) Nielsen, 1973 Private Communication

7) Robinson, 1965 'A Machine Oriented Logic Based on the Resolution Principle' J ACM Vol. 12 No. 1, Jan. 1965.

8) Cox 1962, Renewal Theory, Methven 1962.

9) D. S. Nielsen and B. Runge. Unreliability of a Standby System with Repair a Imperfect Switching, to be published in IEEE Trans Reliability, 1974

APPENDIX 1

SETS, FUNCTIONS, AND SYSTEMS

Notation

The concept of a system used here is based on mathematical logic, set theory, and general systems theory. The logical notation used is not standard, and so a short overview is given here

mathematical logic

The symbols used are shown in fig. 19.

Meanings for the letters used may be given as follows. Letters at the beginning of the alphabet represent 'individual constants', names of individual objects such as 'this girl', 'the colour green', 'the set of all integers'. Capital letters at the end of the alphabet represent 'propositions', that is statements which are either true or false e.g. 'this girl is young'; or predicates, that is, truth statements including variables e.g. 'x is young'. Lower case letters at the end of the alphabet represent variables. The letters P and Q, in this section, represent general strings of symbols, and are used to describe the way expressions are built up.

Certain of the symbols used are regarded as basic symbols. The way in which the basic symbols may be combined to form expressions is shown in fig. 20. Other symbols are introduced by definition, in terms of the basic symbols. These are shown in fig. 21.

In addition to the symbols described in table 1 and table 2, the sets of real and natural numbers are assumed to exist, and also the predicates 'greater than' and 'less than' are used, and the functions 'plus', 'minus' 'times' etc.

constant letters		a, b, c .....
Proposition letters, Predicate letters		X, Y, X_1, Y_1 .....
variable letters		x, y, x_1, y, ...
notation used here		standard notation
not		¬
or		∨
for all		∀
=		=
is a		∈
implies		⇒
and		∧ &
if then		⇒
iff	if and only if	⇔
thereisa	such that	∃
the unique	such that	ι
therisa unique	such that	∃!

Fig. 19. Symbols used

1 each letter is a term (upper and lower case letters included, but not P, or Q)

2 if x and X are terms then x isa X is a formula

3 if x and Y are terms, then X = Y is a formula

4 if P is a formula then not P is a formula

5 if P and Q are formulae, then P or Q is a formula

6 if P is a formula, then forall x, P is a formula.

7 if P is a formula then the unique x such that P is a term.

8 The only terms and formulae are those given by rules 1 to 7

Fig. 20. Construction of logical formulae

P implies Q	is defined as	not P or Q
P and Q	is defined as	not ((not P) or (not Q))
if P then Q	is defined as	P implies Q
if P then Q else R	is defined as	(P implies Q) and ((not P) implies R)
P iff Q (if and only if)	is defined as	(P implies Q) and (Q implies P)
thereisa x such that P	is defined as	not for all x, (not P)

Fig. 21. New symbols introduced by definition

Sets and functions

Given the symbols already defined, it is possible to define 'sets'. The set of all objects with the property P (where P is a predicate) is written

$$\{ x \mid P(x) \}$$

and is read 'the set of all x such that P is true of x'
This expression is equivalent to:

The unique X such that forall x

(x isa X iff ((thereisa Y such that x isa Y) and P(x)))

Another way to describe a set is to list its members e. g.

$$\{ x, y, z \}$$

is the set which contains just x, y, and z .

An ordered pair is a set in which one member of the set is distinguished, being the first member of the set. An ordered pair is written as

$$(x, y)$$

and is defined as

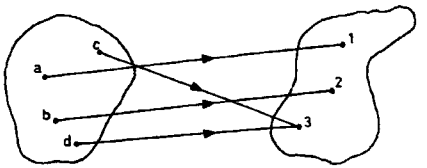
$$\{ x, \{ x, y \} \}$$

Note the way that x is distinguished as the first member of the pair, by including it in the definition in two different ways.

A function is a set of ordered pairs. The idea of a function is that, given the first member of an ordered pair, a single second member can be found. For example, given the function {(a, 1), (b, 2), (c, 3), (d, 3)} and the parameter b, the value of the function, 2, can be found.

Functions can be written using set notation, as above, or they can be given names e. g. F. Provision of a name for a function allows the value of a function for a particular parameter to be written.
e.g. F(x) is the value of the function F given the parameter x.

The set of 'first elements' in a function (a, b, and c in the example above) is called the domain of the function. The set of second elements is called the range of the function. A function should be thought of as providing a single member of the range set for each member of the domain set.



domain range

Fig. 22. A function

The operators plus, minus, times, etc. are functions with two parameters, since for each pair of parameters, a single value for the function is produced. The arithmetic operators can be written using functional notation

e. g. plus (1, 2) = 3

The usual notation 1 + 2, is regarded as on abbreviation of the functional notation.

Functions of time

In systems theory, different concepts of time are used for different purposes e. g. discrete time, continuous time. The concept of a 'time set' provides a basis for definition of these concepts.

Two sets are introduced here - a set of time instants and a set of time intervals. These two sets should obey certain properties, or axioms, given as follows

T is a set of time instants
τ is a set of time intervals
t_1, t_2 etc. are time instants
τ_1, τ_2 etc. are time intervals

1 all time instants can be compared

forall t_1, forall t_2, $t_1 < t_2$ or $t_1 = t_2$ or $t_1 > t_2$

There is a corresponding rule for time intervals.

2 The sum of a time instant and a time interval is a time instant

forall t, forall τ, $t + \tau$ isa T

There is a corresponding rule for a pair of time intervals τ_1 τ_2

3 the sum relation is associative and reflexive

$$(t + \tau_1) + \tau_2 = t + (\tau_1 + \tau_2)$$

$$t + \tau = \tau + t$$

there is a corresponding set of rules for pairs of time intervals τ_1, τ_2

4 consistency

$$t + \tau_1 = t + \tau_2 \text{ iff } \tau_1 = \tau_2$$

5 closure

a forall t_1 forall τ thereisa t_2 such that $t_2 = t_1 + \tau$

b forall t_1 forall t_2 thereisa τ such that

$$t_1 = t_2 + \tau \text{ or } t_2 = t_1 + \tau$$

6 definition of subtration

$$t_1 = t_2 + \tau \text{ iff } t_2 = t_1 - \tau$$

If an instant of time t_o is chosen, then for any other time instant t_1, rules 5 b and 6 guarantee that there is a time instant τ such that

$$t_1 = t_o + \tau \quad \text{or} \quad t_1 = t_o - \tau$$

Rule 4 guarantees that τ is unique. As a result, the set of time intervals, together with one 'initial time', can serve as a time set.

The real numbers may be used as a time set. So may the integers, or the positive integers, etc. Generally, either two time sets are isomorphic, (one to one correspondence between their elements, and obeying the same rules) or one time set can be treated as a subset of the other.

A function of time is a function for which the domain is a set of time instants, T, and the range is a set of values, V. A vector function of time is a time function for which the set of values is a set of ordered pairs, a set of ordered triples, etc.

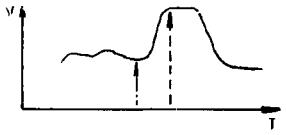
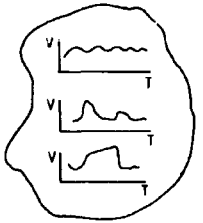


Fig. 23. A function of time

Processes

The treatment here is based on (Windeknecht 1971). A process is a set of functions of time, in which all of the elements (that is, all of the time functions) are defined on the same time set



T is the set of real numbers

Fig. 24. A process

If p is a time function p(t) is the value of p at time t. If P is a process, P [t] is the set of possible values of processes in P at time t.

$$P\,[\,t\,] = \{\, p(t) \mid p \text{ isa } P \}$$

P [t] is called the attainable space of P at time t.

P is the set of all values at which the functions in P may take, and is called the attainable space of P.

The product of two time functions p, and g is defined by

$$pg = \{(t, (x,y)) \mid \text{forall } t,\ t \text{ isa } T \text{ implies } (p(t) = x \text{ and } g(t) = y)\}$$

In other words, if p and g are functions of time, pg is a vector function of time formed by taking the pairs of values p(t) and g(t), for each time t.
The composite of two processes is defined by

$$P\,Q = \{\, pg \mid p \text{ isa } P \text{ and } g \text{ isa } Q \,\}$$

The composite of two processes is formed by taking all possible pairs of functions of time, and combining each pair to produce a vector function of time.

systems

A system description is a description of a composite process, as defined above, or a description of a subset of a composite process. More meaning can be given to this statement if special cases of system descriptions are considered.

A composite process PQ is uncoupled if

(g isa Q and p isa P) implies pg isa PQ

In effect, if p is an input function, it provides no information about the output function.

A composite process PQ is functional if

$$P_1 = P_2 \text{ implies } g_1 = g_2$$

Interpreting p and g as input and output functions of time, the output of a functional processor is completely determined by the input function.

A composite process is bifunctional if

$$p_1 = p_2 \text{ iff } g_1 = g_2$$

Output is determined by input, input is determined by output. The direction of causality is not determined for a bifunctional system description.

A composite process PQ is free if the process P is constant, that is, if p has only one member. In input output terms, there is only a constant input.

In general, a system is simply a composite process. However the example of 'uncoupled composite processes shows that in some cases, the systems may appear somewhat strange. An uncoupled composite process corresponds to a system where input has no effect on output.

memoryless systems

A composite process PQ is static, or memoryless if for all t, PQ(t) is a function.

That is, if p_1 g_1, p_2 g_2 are members of PQ

$$p_1(t) = p_2(t) \text{ implies } g_1(t) = g_2(t)$$

In other words, at each time t, there is only one output value to be associated with each input value. Such systems can be described by means of the function F

forall t, $g(t) = F(t, p(t))$

If the function F is a function of p(t) only, and not of t, the equation can be written

forall t, $g(t) = F(p(t))$

Such systems are both memoryless and time invariant or uniform.

State

The examples of functional, and free systems give some idea of the concept of state. For a free system, there is only one possible value for input. Any differences in output are therefore explained in terms of differences in 'initial state'.

A functional system has only one possible output function of time, for each input function of time. It is then natural to say that a functional system has only one initial state.

In general, a state description of a system P, will consist of two further systems, R and Q, connected in series. R will have an output which depends only on past input. The value of the output of R at time t, will correspond to state at time t. Q will be memoryless, and so correspond to an output function for the system. Windeknecht has shown how these ideas can be formalised, and that a state description can be given for any system. At the very worst, a state can be provided at each time t for every future output function. This corresponds to an explanation of all outputs in terms of changes of state alone.

If a system is such that its output at time t is determined solely by inputs up to time t (and not by any differences in 'initial state'), then the set of different input functions up to time t can serve as the state of the system at time t. This is about as far as one can go, in general, in providing state descriptions of systems. It is always possible in principle, for a system to 'record' inputs up to time t, and to change the output at time t so that it depends uniquely on this information.

APPENDIX 2

DEDUCTION OF CONDITIONS AND EVENTS

Conditions

A predicate $P(t_1, t_2, f)$ is a condition description if

forall t_a, t_b, [(t_a isa T and t_b isa T and $t_1 \neq t_a < t_b < t_2$) implies $P(t_a, t_b, f)$]

This effectively states that a condition description is a predicate of such a form, that if it holds during a period, it holds during any subset of that period. To deduce a condition accross a component means to deduce a condition description with the component output signal as parameter, and which

contains as much irredundant information about the output signal as possible.

The idea of deduction accross a component can be made more precise, if the language for expressing condition descriptions is restricted, so that the only symbols to be used are forall, and, not, function symbols, predicate symbols, variables, and constants. The form of statements required is called skolem normal form, and is defined as follows.

A constant is a term

A variable is a term

A function symbol followed by a string of terms is a term(the number of terms following the function symbol is the degree of the function)

A predicate symbol followed by a string of terms is a formula.

A formula preceded by a not symbol is a literal

A set of formulae, separated by or symbols is a clause.

A set of clauses separated by and symbols is a matrix.

A forall symbol followed by a variable is a quantification.

A matrix, preceded by a set of quantifications, so that there is one quantification for each variable in the matrix, is a statement in skolem normal form.

Any mathematical statement can, if necessary, be translated to skolem normal form, although methods which avoid doing this are preferable because of the inconvenience and complexity of the resulting statements. If a statement does not involve variables, then skolem normal form simplifies, to become conjunctive normal form.

A method, called resolution, (Robinson 1965) exists for deducing all of the clauses which follow, or can be deduced from, a given initial set of clauses.

For clauses which do not involve variables, the method is quite simple e.g.

1 (A or B or C) and (D or E or (not C))

2 A or B or D or E

Which means that line 2 can be deduced from line 1.

For clauses which do involve variables, it involves finding a substitution for the variables, so that a deduction of the kind given above can be made.

e.g.

Forall x, Forall y, (A or C(x)) and (B or not C(y))

Forall z, (A or C(z)) and (B or not C(z))

A or B

Once all possible clauses have been deduced, they can be separated, and each clause treated as a statement in its own right. Resolution, applied to a set of clauses consisting of an input condition description and a component description, produces an extended set of clauses. This extended set is an apt candidate for the role of output condition description.

There are two problems with this new output condition description. It contains too much information about the input conditions and the component itself. And it contains some redundant information about the output of the component itself.

The first problem can be solved by striking out all clauses which make no reference to the output of the component.

The second problem, of redundant information, arises because of pairs of clauses of the form

A or B or C and A or B

The second clause, A or B is said to subsume the first clause, A or B or C, because whenever the second clause is true, the first clause is inevitably true. Similarly the clause forall x, P(x) subsumes the clause P(a). Subsumed clauses add no additional information to the output condition description, and are not necessary for deducing subsequent output conditions for other components. Subsumed clauses may therefore be deleted. Resolution is a complete method, in that it will find all possible clauses which can be deduced (by any sound method) from an original set of clauses. However, resolution, in its own right, treats only those expressions containing and, or, not, and forall symbols. The set of expressions treated can be very quickly extended, to include if-then-else, implies, there exists. However extensions to include the symbols of set theory, isa, = , or to include the symbols of arithmetic, , , +, - , is much more difficult.

There are two basic ways of extending the scope of the resolution method. One is to add a set of new clauses, as axioms to describe the new operators. These are introduced for all subsequent deductions and lead to a considerable loss of efficiency. An alternative is to develop new methods of deduction, which are complete, like resolution, but allow more symbols to be used. This is a considerable technical problem, but some progress has been made (Robinson 196 Slagle 1972). For many practical purposes, incomplete methods may suffice, provided only that all the desired output conditions are obtained.

The general form of an output condition, if a component with memory and with several inputs, is treated, will be

If A (input) and B(state) then C(output)

and If A_1(input) and B_1(state) then C_1(output)

and If A_n(input) and B_n(state) then C_n(output)

This form will correspond to a condition tree in a cause consequence diagram, and will be called a contingent condition.

events

An ordered pair of condition descriptions,

$A = (P_1(t_{11}, t_{12}, f_1), P_2(t_{21}, t_{22}, f_2))$,

is an event description if and only if

$f_1 = f_2 = f$

and thereisa t_1, t_2 such that
forall t not$\{(P_1(t_1, t, f)$ implies $P_2(t, t_2, f)\}$

In other words P_1 and P_2 refer to the same time function f; and either $P_1(t, t, f)$ is inconsistent with $P_2(t, t_2, f)$, or $P_2(t, t_2, f)$ contains more information or is more precise, than anything which could be deduced from $P_1(t_1, t, f)$.

The fact that event A occurs at time t_e is written

$t_e: P_1(t_1, t, f) \rightarrow P_2(t, t_2, f)$

This is equivalent to:

there is a t_1 such that $P_1(t_1, t_e, f)$
and there is a t_2 such that $P_2(t_e, t_2, f)$

To deduce an event accross a component involves taking three sets of clauses

1 The component description
2 The set of conditions on any 'auxiliary inputs'
3 The set of 'state conditions'

and resolving these with the first part of an input event description, to produce the first output condition description, P_1'. In general, this will be a contingent condition description, and will be valid over a wider period of time (i.e. later) than the input condition.

The second step is to repeat this deduction, but using the second part of the event description, to obtain P_2'. The third step is to try to find if there are any points in time, greater than t_e, at which either P_1' is inconsistent with P_2', or at which P_2' cannot be deduced from P_1'. A theorem proving program can be used for both of these tasks, but there are problems. The time to deduce inconsistency cannot be predicted, and so in general it is

impossible to guarantee inconsistency within a finite period of computing time. (the condition descriptions may be consistent anyway). Similarly, the only way to prove that P_2' cannot be deduced from P_1' is to try to make the deduction, for a potentially infinite period of time.

However, in practice, consistency and deducibility can be judged 'by eye', if the formulae are presented in a reasonable way. And for more difficult cases, it is sufficient to try to prove consistency, for a reasonable period of time, and if unsuccessful, to try to prove inconsistency, again for a reasonable period. If neither attempt is successful, the 'safe' assumption, that an event takes place, is assumed.

The result is, in general, a contingent event description, the occurrence of the event being dependent on the state of the component; and on the conditions at any inputs other than the one associated with the input event description.

As an example of this process, a 'thermal relay' can be used. Such a relay has the property that if an input current is provided at time t, the relay closed at some later time, t + a. If the input current is shut off at time t', the relay opens at some later time, t' + b. A rapid sequence in which the relay current is cut, and then restored once more, will not result in the relay contacts opening.

An approximate model of a thermal relay, which shows this behaviour, is:

Forall t, $y(t) = 1$ iff
[(Forall t_1, $t - a < t_1 < t$ implies $x(t_1) = 1$)
or (thereisa t_2 such that $x(t_2) = 1$ and $y(t_2) = 1$
and $t_2 < t < t_2 + b$)]

The result of an input event

$t_e: x(t) = 0 \rightarrow x(t) = 1$

is evaluated in diagram 26

The resulting output event is.

Condition 1

Forall t, [not (therisa t' such that $x(t') = 1$ and $y(t') = 1$ and $t' < t < (t' + b)$
and $t'' < t < t_e + a$] implies $y(t) = 0$

Condition 2

Forall t, [(thereisa t' such that $x(t') = 1$ and $y(t') = 1$ and $t' < t < t' + b)$
or $t_e + a < t < t'''$] implies $y(t) = 1$

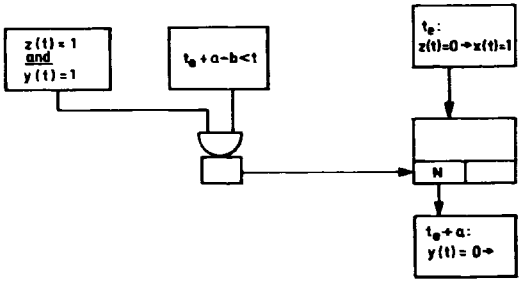


Fig. 25. Contingent event

This can be reformulated as two contingent events

1 not therisa t such that $t_e + a - b < t < t''$ and x(t) = 1 and y(t) = 1

$t_e + a$: y(t) = 0 → y(t) = 1

2 not thereisa t such that $t_e + a - b < t < t''$ and x(t) = 1 and y(t) = 1

and therisa t such that $t'' - b < t < t_e + a$ and x(t) = 1 and y(t) = 1

: y(t) = 0 → y(t) = 1

On checking the second event - it proves impossible to fulfill the necessary conditions, within the period of interest. After producing an output event description, it is necessary to check whether the conditions upon which the output event description are contingent, can actually exist. In principle, this means checking backwards through the component network at each step, is a similar fashion to event tracing. However, a simpler procedure will be described.

First, when begining a cause consequence analysis, an initial set of conditions are required for each of the variables of the system. These are either assumptions, or are derived from other assumptions by deduction. These initial conditions are recorded for each component.

In evaluating a sequence of events starting from, say, a single failure, it is only necessary to check that the event sequence produced is consistent with the initial conditions. In evaluating the event sequence, a new set of

Forall t,

[(forall t_1, $t-a < t_1 < t$ implies $x(t_1)=1$) implies $y(t)=1$]

and [(thereis t_2 suchthat $x(t_2)=1$ and $y(t_2)=1$ and $t_2 < t < t_2+b$) implies $y(t)=1$]

and not [(forall t_1, $t-a < t_1 < t$ implies $x(t_1)=1$)

or (thereis t_2 suchthat $x(t_2)=1$ and $y(t_2)=1$ and $t_2 < t < t_2+b$)] implies $y(t)=0$

Eliminate implication signs.

Forall t,

[not (forall t_1, not ($t-a < t_1 < t$) or $x(t_1)=1$) or $y(t)=1$]

and [not (thereis t_2 suchthat $x(t_2)=1$ and $y(t_2)=1$ and $t_2 < t < t_2+b$) or $y(t)=1$]

and [(forall t_1, not ($t-a < t_1 < t$) or $x(t_1)=1$)

or (thereis t_2 suchthat $x(t_2)=1$ and $y(t_2)=1$ and $t_2 < t < t_2+b$)] or $y(t)=0$

Reduce scope of negation signs

Forall t,

[(forall t_1, $t-a < t_1 < t$ or not $x(t_1)=1$) or $y(t)=1$]

and [(forall t_2, not $x(t_2)=1$ or not $y(t_2)=1$ or not $t_2 < t < t_2+b$) or $y(t)=1$]

and [(forall t_3, not ($t-a < t_3 < t$) or $x(t_3)=1$)

or (thereis t_4 suchthat $x(t_4)=1$ and $y(t_4)=1$ and $t_4 < t < t_4+b$)] or $y(t)=0$

Fig. 26. Deduction of an event accross a thermal relay component.

Eliminate existential quantifier, collect universal quantifiers

Set $P(t) = t_4$, the time (neither t and greater than $t-a$, at which $x(t)$ and $y(t)$ of — if such a time exists.

Forall t, Forall t_1, Forall t_2, Forall t_3,

$$\left[t_4 < t_1 < t \quad \underline{\text{or not}} \quad x(t_1)=1 \quad \underline{\text{or}} \quad y(t)=1\right]$$

$$\underline{\text{and}} \left[\underline{\text{not}}\ x(t_2)=1 \ \underline{\text{or not}}\ y(t_2)=1 \ \underline{\text{or not}}\ t_2 < t < t_2+b \ \underline{\text{or}}\ y(t)=1\right]$$

$$\underline{\text{and}} \left[\underline{\text{not}}\ t-a < t_3 < t \quad \underline{\text{or}}\ x(t_3)=1 \right.$$
$$\underline{\text{or}} \left(x(P(t))=1 \ \underline{\text{and}}\ y(P(t))=1 \ \underline{\text{and}}\ P(t) < t < P(t)+b\right)$$
$$\left.\underline{\text{or}}\ y(t)=0\right]$$

Translate to conjunctive normal form.

1 $t-a < t_1 < t \ \underline{\text{or not}}\ x(t_1)=1 \ \underline{\text{or}}\ y(t)=1$

2 $\underline{\text{and}}\ \underline{\text{not}}\ x(t_2)=1 \ \underline{\text{or not}}\ y(t_2)=1 \ \underline{\text{or not}}\ t_2 < t < t_2+b \ \underline{\text{or}}\ y(t)=1$

3 $\underline{\text{and}}\ x(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or}}\ x(t_3)=1 \ \underline{\text{or not}}\ y(t)=1$

4 $\underline{\text{and}}\ y(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or}}\ x(t_3)=1 \ \underline{\text{or not}}\ y(t)=1$

5 $\underline{\text{and}}\ P(t) < t < P(t)+b \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or}}\ x(t_3)=1 \ \underline{\text{or not}}\ y(t)=1$

Fig. 26. Continued.

evaluate effect of event t_6: $x(t)=0 \rightarrow x(t)=1$

this translates to

there is t_5 such that forall t_4, $(t_5 < t_4 < t_6)$ implies $x(t_4)=0$

and there is t_7 such that forall t_8, $(t_6 < t_8 < t_7)$ implies $x(t_8)=1$

In skolem normal form

6 Forall t_4 not $(P_1(t_6) < t_4 < t_6)$ or not $x(t_4) = 1$

7 and Forall t_8 not $(t_6 < t_8 < P_2(t_8))$ or $x(t_8)=1$

Resolution between line 6 and line 3, $t_3 = t_4$

8 $x(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6 \ \underline{\text{or not}}\ y(t)=1$

Similarly

9 $y(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6 \ \underline{\text{or not}}\ y(t)=1$

10 $P(t) < t < P(t)+b \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6 \ \underline{\text{or not}}\ y(t)=1$

---

translate lines 8, 9, 10 to implicant form for $y(t)$

$$\left(\underline{\text{not}} \left[\langle x(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6\rangle\right.\right.$$
$$\underline{\text{and}} \langle y(P(t))=1 \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6\rangle$$
$$\left.\left.\underline{\text{and}} \langle P(t) < t < P(t)+b \ \underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6\rangle\right]\right) \underline{\text{implies}}\ y(t)=0$$

$$\left(\underline{\text{not}} \left[\langle x(P(t))=1 \ \underline{\text{and}}\ y(P(t))=1 \ \underline{\text{and}}\ P(t) < t < P(t)+b\rangle\right.\right.$$
$$\left.\left.\underline{\text{or not}}\ t-a < t_3 < t \ \underline{\text{or not}}\ P_1(t_6) < t_3 < t_6\right]\right) \underline{\text{implies}}\ y(t)=0$$

$$\left[\underline{\text{not}} \left[x(P(t))=1 \ \underline{\text{and}}\ y(P(t))=1 \ \underline{\text{and}}\ P(t) < t < P(t)+b\right]\right.$$
$$\underline{\text{and}}\ t-a < t_3 < t$$
$$\left.\underline{\text{and}}\ P_1(t_6) < t_3 < t\right] \quad \underline{\text{implies}} \quad y(t)=0$$

Fig. 26. Continued.

translate to 'event form'

forall t_3 forall t, there is $P_1(t_e)$ such that $P_1(t_e) < t_3 < t_e$

and $t-a < t_3 < t$

and not (there is $P(t)$ such that $x(P(t))=1$
and $y(P(t))=1$
and $P(t) < t < P(t)+b$)

implies $y(t) = 0$.

---

there is $P_1(t_e)$ such that

forall t, [(forall t_3, $P_1(t_e) < t_3 < t_e$ and $t-a < t_3 < t$)

and not (there is $P(t)$ such that $x(P(t)) = 1$
and $y(P(t)) = 1$
and $P(t) < t < P(t)+b$)] implies $y(t)=0$

This has the form of the first half of an event description.

It also has the form of a contingent condition

In a more meaningful form

there is $P_1(t_e)$ such that

forall t, [$P_1(t_e) < t < t_e+a$

and not [there is $P_e(t)$ such that $x(P(t)) = 1$
and $y(P(t)) = 1$
and $P(t) < t < P(t)+b$)] implies $y(t)=0$

Fig. 26. Continued

In a similar way, the second half of the event description becomes

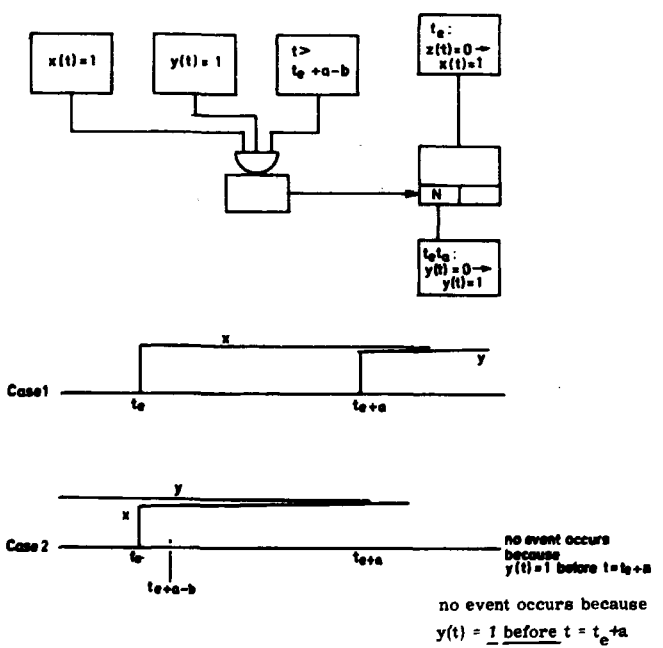
12 there is $P_2(t_e)$ such that

forall t, [$t_e + a < t < P_2(t_e)$

or (there is $P_1(t)$ such that $x(P(t)) = 1$
and $y(P(t)) = 1$
and $P(t) < t < P(t)+b$] implies $y(t)=1$

By inspection, this is, taken together with II, a contingent event.

Fig. 26. Continued.



no event occurs because
y(t) = 1 before t = t_e+a

Fig. 26. Continued.

conditions will be deduced for each component. These must also be stored, temporarily, for checking, in the case that there is a loop in the component network.

If a 'double failure' analysis is to be performed, not only the initial conditions must be checked, for each event, but also the set of conditions which might result from a previous failure.

APPENDIX 3

CALCULATION OF EVENT PROBABILITY DISTRIBUTIONS

Many of the events in a failure mode analysis are 'certain'. That is, they will certainly happen at some particular time during system operation. Many failure events, though, are best described by giving a probability of failure.

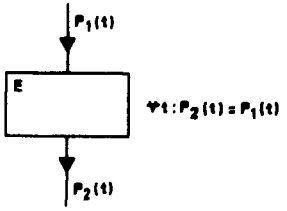
For use with automated failure analysis, the most convenient method is to work with an event 'probability density function', p.d.f. (Cox 1962).

If x is the event time

$$\text{p.d.f. } (x) = \underset{\Delta t \to o}{(t} \quad \frac{\text{prob } (t < x \leq t + \Delta t)}{\Delta t}$$

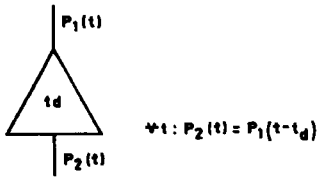
There will be a pdf associated with every event description box in an event sequence diagram. What follows is a set of rules for finding the pdf's for all the event descriptions, when the p.d.f.'s for a set of initial events are given.

1 Simple chain of events, without delay



The probability of occurrence of an event at time t, is the same for a cause event and a consequence event

2 Chain of events with delay



3 Chain of events with non deterministic delay

In some cases, one event follows another, with a delay which cannot be determined precisely, but a probability distribution can be given for the delay.

Let x_1 be the time for the first event, t_d be the delay time x_2 be the time for the second event. Let P_1, P_2, P_3, be the corresponding pdf's. Let p_1, p_2, p_3 be the corresponding 'incremental probabilities'.

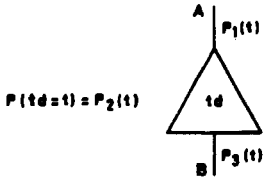
then

$$p_3(t \quad x_2 \quad t + \quad t) \approx \sum_{t'=t}^{t'=o} p\,[t - t' < x_1 \leq t - t' + \Delta t \text{ and } t' < t_d \leq t' + \Delta t]$$

$$= \int_{x_2}^{o} p_1(t - t').\ p_2(t')\ dt'$$

since event and length of delay are independent

$$P_3(t) = \int_{t}^{o} P_1(t - t').\ P_2(t')\ dt'$$



$$\forall t : P_3(t) = \int_0^t P_1(t-x)\ P_2(x)\ dx$$

4 Event depends on a prior condition

In some cases an event A, occurring at time t, will cause an event B, occuring at time t, only if some condition C is fulfilled at the time t.

Let P_1 be the probability that A occurs between times t and $t + \Delta t$.

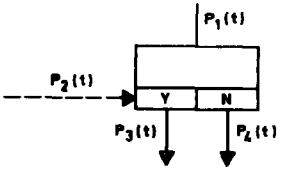
Let P_2 be the probability that condition C holds between times t and $t + \Delta t$

Let P_3 be the probability that event B occurs between times t and $t + \Delta t$

Then

$P_1 = P$ (A and C)

$= P_1 \cdot P_2$ given that A and C are independent



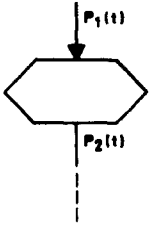
$\forall t \; P_3(t) = P_1(t) \cdot P_2(t)$

$\forall t \; P_4(t) = P_1(t) \cdot (1 - P_2(t))$

Decision box

5 Contingent events

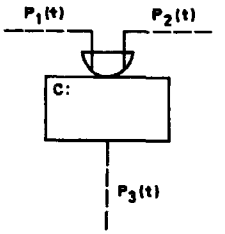
In many cases, an event B occurs as a result of an event A, but only if some other event, C, has already occurred. The probability that an event has occurred prior to time t, is the integral of the probability density function, and is called the cumulative distribution function.



$\forall t : P_2(t) = \int_0^t P_1(x)\, dx$

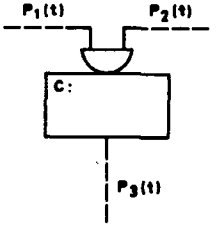
Event to condition box

Note the change of time variable involved.



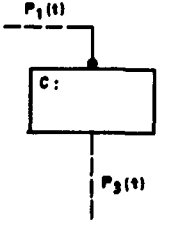
Or box

$\forall t : P_3(t) = P_1(t) + P_2(t) - P_1(t) \cdot P_2(t)$



And box

$\forall t : P_3(t) = P_1(t) \cdot P_2(t)$



Not box

$\forall t : P_3(t) = 1 - P_1(t)$

6 Compound conditions

Conditions are best described by the probability that the condition holds at time t. This will generally be a result of the fact that some event has occurred prior to time t. So conditions are described by cumulative distribution functions.

Combinations of independent conditions may be evaluated as in the following diagrams. As an example - the probability that two conditions, A and B, both hold at time t, is given by

let $P_1(t)$ be the probability that A holds at time t
let $P_2(t)$ be the probability that B holds at time t
let conditions A and B be independent.

Then

$P_3(t)$ = P (A holds at time t and B holds at time t)
= $P_1(t) . P_2(t)$