Cross-viewpoint Consistency in Accident Investigations

Alastair J. Telford and Christopher W. Johnson

Dept. of Computing Science, University of Glasgow,

17, Lilybank Gardens, Glasgow, G12 8QQ

Tel: +44 141 339 8855 ext. 8094
Fax: +44 141 330 4913
Email: {alastair,johnson}@dcs.gla.ac.uk
WWW: http://www.dcs.gla.ac.uk/~johnson/gaag/gaag.html

August 1996

Abstract

Accident reports are typically divided into chapters which reflect the different perspectives of various specialists. It is sometimes the case that these alternative viewpoints lead to inconsistencies and omissions where an incident covered at length by one specialist may be ignored by another. This creates a problem in that the conclusions of the report may thus be muddled or incomplete and consequently lead to further errors of comprehension amongst those who need to understand the findings. Similarly, several different reports of the same accident may be produced. This creates problems for the companies and regulators which must act on the findings of these inconsistent accounts. It is, therefore, desirable to have a rigorous method for checking consistency between different chapters and accounts.

We present the novel application of the idea of cross-viewpoint consistency, typically used in software engineering, to unify different accident perspectives. Furthermore, we propose a method of consistency checking that should ensure a new, more detailed view of the overall accident is produced. This will encompass two or more different viewpoints. In doing so, we show, for the first time, how cross-viewpoint consistency checking may be achieved for accident analysis. It is hypothesised that this will help to resolve the inconsistencies that frequently arise between different accounts of the same accident.

Keywords: Accident analysis, Temporal Logic of Actions, Cross-viewpoint consistency, Unifying specification, Internal consistency checking, Formal methods.

Contents

List of Tables

1 Introduction

Accident reports are typically separated into sections or chapters which each focus upon distinct aspects of the accident. This separation can lead to difference in emphasis. What is seen as important within one part of the accident can be disregarded in another. For example, in the report on the Piper Alpha oil rig explosion [7], different chapters deal with, respectively, the way that the disaster developed after the initial explosion and the operation of the "permit to work" system which operated on the rig. Different viewpoints can thus lead to incomplete or vague conclusions or, at the other extreme, findings which focus upon a specific aspect of the accident which would only be germane to that particular scenario, so being of little use in future accident prevention. The accident inquiry and reporting process is thus seriously weakened.

This problem with separate perspectives is exacerbated when there is more than one report upon an accident, as was the case with the Three Mile Island nuclear accident in March 1979. There were three major reports into that accident, one by the US Nuclear Regulatory Commission [22], one by the US President's special inquiry and one by the plant operators [17]. Furthermore, media articles, technical specifications and legislation are constructed from these reports which may perpetuate or, indeed, amplify inconsistencies within the original document(s). Indeed, there exists a bibliography of the pieces written about the Three Mile Island accident which itself extends to more than four hundred pages [23]. This copious amount of reporting therefore makes the Three Mile Island incident a valuable example for this paper. The cross-viewpoint consistency techniques we shall describe have been proven in the area of software engineering, especially in the building of operationally correct distributed and concurrent systems. We argue that they are of direct relevance to accident analysis in that disaster scenarios may be viewed as complex, concurrent systems. Moreover, specialists which investigate accidents from different perspectives are similar to requirements engineers who specify different components of a distributed software system.

As an example of cross-viewpoint consistency checking, we shall take two accounts, one by Bignell and Fortune [4] and the other by Fremlin [10]. We shall attempt to investigate them formally using Lamport's Temporal Logic of Actions (TLA) [16] and we will develop a methodology for cross-viewpoint consistency checking for accident analysis using TLA. We shall use TLA to develop a *unifying formalisation* from two existing reports. This idea is illustrated in Figure 1. We believe that this is a novel application of the idea of cross-viewpoint consistency.

1.1 The Three Mile Island Accident Overview

The incident, which took place in the number two reactor of the Three Mile Island nuclear plant in Pennsylvania, started on 28th March 1979. An error during maintenance led to valves switching off and, whilst most of the corrective measures worked properly, one relief valve remained open and allowed too much water to flow out from the primary water circuit which conducted heat away from the reactor core. This meant that a loss of coolant

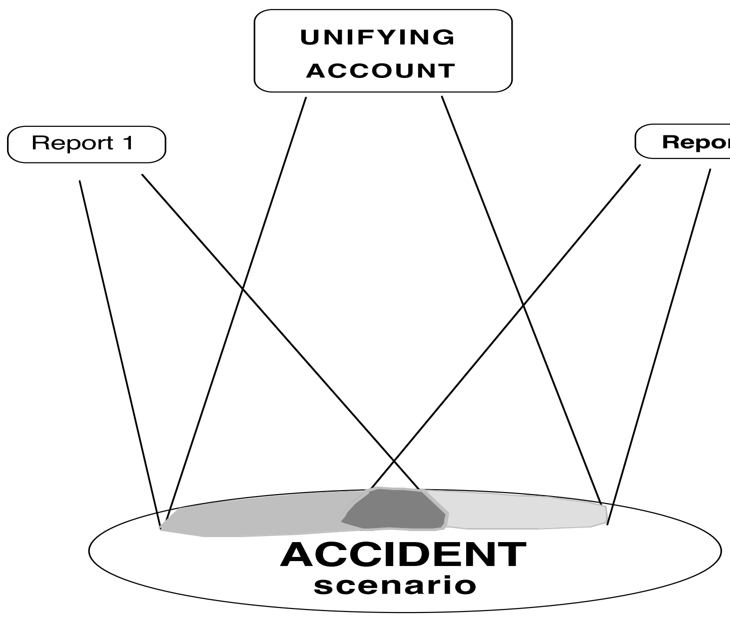

Figure 1: Diagram of different viewpoints of an accident and how a unifying account may resolve different reports.

accident (LOCA) occurred which led to a significant discharge of radioactive material into the atmosphere and almost caused a catastrophic explosion at the plant.

1.2 Structure of the paper

In Section 2 we describe the basic structure of the Temporal Logic of Actions and how it may be used to formalise accident reports. We describe the various components of the formalism and how they may be related to parts of an accident scenario. We show how variables may be "hidden" within a specification of a scenario and how this can be used for further refinements.

This aspect of TLA is vital for the cross-viewpoint consistency checking which is detailed in Section 4. Prior to this, in Section 3, we present formalisations within TLA of two separate accounts of the Three Mile Island disaster. The Section on cross-viewpoint consistency checking (4), as well as providing the theoretical basis for providing a unification, shows how the two example accounts may be resolved within one refining specification.

We present our conclusions and avenues for future work in Section 6.

2 Using TLA in Accident Analysis

We need a common formal framework for the two reports in order to develop a unifying formal specification which will be used to establish cross-viewpoint consistency. The formalisation of each account thus allows the *internal consistency* of each informal account to be checked and enables discrepancies to be deduced. We shall use the temporal logic of actions as the medium for our formalisation. TLA is an elaboration, by Lamport, of simple temporal logic [19, 20] where the elementary formulas of the logic are based upon *actions*. Actions are boolean valued expressions involving variables, their primed equivalents (where x' denotes the new value of x in an updated state) and simple values and constants. $\Box$ ("always" — in all states) and $\Diamond$ ("eventually" — in some state). More specifically, the basic formulas of TLA are either predicates (actions involving no primed variables) or of the form $\Box[\mathcal{A}]_f$ — this notation is explained in Section 2.1 below.

2.1 Fundamentals of TLA for accident analysis

The formalisations of the accounts are up to the point where the LOCA had started. (We define the LOCA to be the point where water flow stopped in the primary circuit surrounding the reactor core.) In our formalisation, we wish to establish an *eventuality* property, — namely that given the basic operating conditions of the nuclear plant, the catastrophic conditions that were *enabled* on 28th March 1979, eventually led to an LOCA event. At the top level, therefore, we wish to establish that,

$$(Opconds \wedge Enabled(Cat)) \Rightarrow \Diamond LOCA$$

In the above,

Opconds represents the operating conditions existing at the TMI plant, prior to and during the accident.

Cat represents the set of catastrophic conditions.

LOCA is the Loss of Coolant Accident event.

The *Enabled* predicate is explained in Section 2.5. $\leadsto$ is the temporal "leads to" operator: $A \leadsto B$ means that B is, sometime in the future, a consequence of A. We discuss this and the semantics of causality further in Section 2.7 below. This can be shown to be equivalent to:

$$Opconds \Rightarrow (Enabled(Cat) \leadsto LOCA)$$

If we cannot prove the above assertion then we may deduce that either *Opconds* or *Cat* is wrong. That is, either our model for the operating conditions of the plant or the set of catastrophic actions which led to the accident is incomplete in some respect. This does provide useful information for accident analysts since it should be possible for analysts to pinpoint the place(s) where the proof fails and consequently the missing components in the formal model of the accident. This reasoning can, therefore, be used to identify gaps in the original, informal account.

We formulate the *Opconds* in a similar way to that which Lamport presents for a general formalisation of a concurrent program. That is, *Opconds* is defined as follows:-

$$Opconds \triangleq Init \wedge \Box[\mathcal{N}]_f \wedge F$$

In the above,

- *Init* represents the initial conditions of the operation of the plant.
- $\mathcal{N}$ is an *action* which represents the *next-state* relation of the operation of the plant.
- F is the *fairness* condition.
- f is a *state function* comprising a tuple of all the variables included in the formalisation.

This model provides a common framework for expressing the workings of a system (e.g. a nuclear power plant, a North Sea oil rig etc.) at which an accident took place. We circumscribe the state of components at the start of the scenario by *Init*, which clearly specifies what was thought to be occurring before the accident started. The next-state relation, $\mathcal{N}$, describes actual events as they occur within an accident situation. This includes actions that need not have disastrous consequences i.e. actions which would comprise part of normal operations such as the water turning to steam to drive the turbine in a power station. The fairness condition, F, gives equal weight to each sub-action comprising the entire model – this means that we ensure that the system is live in that an infinite series of stuttering steps does not occur. In our model, we do not want to didactically stipulate that each action must occur simultaneously, or in a pre-ordained sequence. The allowance for stuttering steps means that components of the accident may remain unaltered for several states in the execution of the model, but the fairness condition ensures that the components will eventually change their values if it ever becomes possible for them to do so.

2.2 The Initial conditions, *Init*

Init represents the operating conditions of the plant prior to the occurrence of the accident. For example, this will specify that the relief valve to the primary circuit was initially closed ($relief\_valve = \mathsf{closed}$) and that water flow in the primary circuit was normal ($primary\_flow = \mathsf{normal}$). When establishing cross-viewpoint consistency it will be necessary to establish that the initial conditions for a unifying formalisation may be considered a refinement of each *Init* formula.

2.3 The Next-state relation, $\mathcal{N}$

$\mathcal{N}$ represents the relationship of variables between states within a (discrete time) model of the nuclear plant operation. For example, we could give the temperature of the water in the primary circuit as it relates to the temperature of the water in the secondary circuit. f is a *state function* consisting of simply a tuple of all the variables used in the model: this allows us to specify *stuttering* steps where the values of the variables do not change. Allowing these stuttering steps is essential to enable refinement of the model at a later stage: see Section 8.4 of [16]. Briefly, the stuttering steps mean that certain variables will not be altered in some states. A refinement of the model can then be made by introducing new variables, which would not exist, and therefore would not alter in the original. If stuttering was not allowed then we could not say that the new model was a refinement of the first, since variables would have to be altered by every action. $[\mathcal{N}]_f$ is defined thus:-

$$[\mathcal{N}]_f \triangleq \mathcal{N} \vee (f' = f)$$

This states that the action $\mathcal{N}$ may occur or that all the variables included within f will be unchanged. $\mathcal{N}$ is the disjunction of all the operating actions, $\mathcal{R}$, together with the catastrophic actions, *Cat*. Some of the actions of $\mathcal{R}$ will only be enabled once some of the actions comprising *Cat* are enabled.

2.4 The Fairness condition, F

It may seem strange, at first, to include a fairness condition, F, for the operation of the nuclear power plant. However, it is through F that we formalise the notion that the plant either runs as planned or it fails: fairness conditions such as F allow eventuality properties, such as the one above, to be established. We simplify the fairness condition of the nuclear power plant operation to state that always either the actions of the plant must occur or eventually they become always impossible. This is a *strong* fairness condition (see [16, Section 6.3]) and is formulated thus:-

$$F \triangleq (\Box\Diamond\langle\mathcal{A}\rangle_f) \vee (\Box\Diamond\neg \mathit{Enabled}(\langle\mathcal{A}\rangle_f))$$

Again, f refers to the tuple of variables in our model of TMI. The angle brackets ($\langle\rangle$) indicate an action where stuttering steps are forbidden i.e.

$$\langle\mathcal{A}\rangle_f \triangleq \mathcal{A} \wedge (f' \neq f)$$

This is useful where we wish to stipulate the fact that stuttering steps are *not* occurring and that some alteration to components is definitely taking place within an accident model. It is from these notions that we obtain *liveness* within the system and this allows *eventuality* properties (of the form $A \leadsto B$) to be deduced.

The following equivalence can be easily established as a consequence of the fact that $\Diamond G \triangleq \neg\Box\neg G$:

$$\Diamond\langle\mathcal{A}\rangle_f \equiv \neg\Box[\neg\mathcal{A}]_f$$

Indeed, we have strong fairness conditions covering both the ordinary actions of the plant, which we denote $\mathcal{C}$, and the catastrophic actions. This ensures that *both* the catastrophic actions and the operating actions must occur in the model of the event.

2.5 State functions

In the above, the state functions used to subscript actions are simply tuples of all the variables in the specification. Generally, however, state functions consist of any mathematical function upon the variables of the system i.e. the variables (e.g. x) are the inputs to the function and their primed versions (e.g. x') are the outputs.

The *Enabled* Predicate The *Enabled* predicate is true for an action $\mathcal{A}$ and a state s if and only if it is possible to take an $\mathcal{A}$ step starting in that state. That is to say that there exists some other state which is related to s by the action $\mathcal{A}$.

The *Unchanged* Predicate Most of the actions will only change the values of a few of the many component variables of the accident formalisation. We *explicitly* denote which variables are left unchanged, since, as Lamport cogently argues, the mathematical validity of the formalisation will be compromised if it is only done implicitly. This is especially pertinent for accident analysis where we are expressly trying to avoid any "description by implication" where it is *assumed* that the reader will be able to fill in missing details — for example, in Fremlin's account it is mentioned that the emeregency core-cooling system started and "began to raise the coolant level". Implicit in this statement is the fact that the primary circuit water level must have been *significantly* lowered by the opening of a valve to the quench tank. The discipline of making every action explicit via *Unchanged* in our formalisation allows us to avoid such pitfalls which are provided by informal or mathematically incomplete representations. The Unchanged predicate is defined thus over a state function:

$$\mathit{Unchanged}\, f \triangleq f = f'$$

We also allow the same predicate to range over sets. (This is merely treating tuples as sets and vice versa.) The convenience of using sets is that it allows us to specify conveniently that all variables in the system, *Var*, are unchanged apart from one or two. For example, we express the fact that everything remains unaltered apart from the air-circuit valves:

$$\mathit{Unchanged}(\mathit{Var} \setminus \{\mathit{ac\_valves}\})$$

Hiding variables and refinement In some cases in accident analysis, the formalisation will include components whose details are inessential to the system as a whole. For example, in the account by Bignell and Fortune [4] no mention is made of the state of the demineralizer at the plant. Therefore, as a component variable of the formalised system, it may be hidden via an existential quantification, thus:

$$\exists\, \mathit{demineralizer}\,.\, \mathit{Opconds}$$

where, throughout *Opconds*, the *demineralizer* variable remains unchanged.

This idea is particularly useful for unifying two separate accounts of an accident where a component which occurs in one is not mentioned or is unimportant in another. The semantics of this existential quantification over hidden variables is given in [16, Section 9.2].

We can replace hidden variables (h, say), as necessary by state functions ($\overline{h}$) which are defined in terms of other variables or constants in the system. This enables us to *refine* formal accident specifications by replacing hidden variables with their refining functions. For example, in the above, we could substitute a function $\overline{demineralizer}$ for the hidden variable *demineralizer* in *Opconds* to produce a refined version of the specification, denoted $\overline{Opconds}$.

2.6 Types in TLA and Accident Analysis

Lamport rejects the notion of explicitly typing the variables of the system arguing [16, Section 7.1.5] that they can be derived as *properties* of the program. We do not take this view since we believe that typing forms an important part of the documentation of the formalisation, particularly for accident analysis. The typing of a component of the accident instantly tells us the range of values that may be taken by that component and, perhaps more crucially, the operations that can be performed on that component. Our development of accident analysis encourages a form of *strong* typing whereby the operations upon each component are quite narrowly prescriptive. For example, the action of lowering the control rods into the reactor core should not increase the temperature of the primary circuit coolant, if other factors are constant. Therefore we may assert the type of this operation, which we call *lower*, as follows

$$\begin{aligned} lower \;\; : \;\; & \forall t : TempPress.((control\_rods = \mathsf{up}) \wedge (water\_temp = t)) \\ & \Rightarrow (control\_rods' = \mathsf{down}) \wedge (water\_temp \leq t) \\ & \wedge Unchanged(Var \backslash \{control\_rods, water\_temp\}) \end{aligned}$$

This restricts the values of the primary circuit water to be less than or equal to the temperature before the lowering operation. *lower* is a state function and the type reflects the pre-conditions of its operation by predicates involving unprimed variables, whilst predicates using primed variables form its post-conditions. N.B. We are not stating the above as a physical law – indeed, there may be circumstances in which the lowering of the control rods produces an increase in primary circuit temperature. However, all operations performed in the plant have a certain specification and should delimit the way that components may be altered. Typing reflects such specifications; if the typing is incorrect then a deficiency in the basic model of the operation of the plant has been discovered. Typing does, as Lamport says, reflect properties which could be established after the formal model has been created. We use typing to ensure that properties which have been *implicitly assumed* about the plant's operation are stated explicitly at the onset. This may be contrasted with the causal event formulas of Section 3.1.1 which represent properties of the *dynamic* sequence of events described in the accident reports. These formulas have to be proven, as opposed to types which are axiomatic assertions.

2.7 Causality

In accident analysis we typically make extensive use of the "leads to" logical connective, $\leadsto$. This is defined as follows:

$$A \leadsto B \triangleq \Box(A \Rightarrow \Diamond B)$$

This has the intuitive interpretation that it is always the case that if A is true then B will eventually be true. This thus enables us to represent causality in accident accounts formally.

However, the above only represents a form of *weak causality* where B may become true even when A is always false. (This will be the case when B is inevitable i.e. that $\Box\Diamond B$ holds.) We may occasionally require a stronger form of causality where some formula B, representing some *effect*, only occurs due to some *cause*, A. This is particularly the case where we wish to axiomatise the underlying assumptions about the interactive behaviour of components of the plant where we wish to stipulate that one action must or must not be the consequence of another. We represent this by the following definition:

Definition 1 (Strong causality) *We say that A is the* cause *of B if and only if A leads to B and if A does not hold then B cannot follow. This is represented as $A \hookrightarrow B$ and is defined formally thus:*

$$A \hookrightarrow B \triangleq (A \leadsto B) \wedge \neg(\neg A \leadsto B)$$

This stronger notion of causality will be useful when establishing cross-viewpoint consistency relative to a set of axioms about each of the components of the plant. For instance, we might wish to stipulate that opening the relief valve, *RVO*, to the primary water circuit would cause water to runoff, *WRO*:

$$Enabled(RVO) \hookrightarrow Enabled(WRO)$$

Similarly, we wish to stipulate that opening the relief valve does not cause the control rods in the reactor core to be lowered, *CRD*:

$$\neg(Enabled(RVO) \hookrightarrow Enabled(CRD))$$

3 Formalisation of accounts of the disaster

In this section we give the formalisation of two accounts of the Three Mile Island Accident. We have already formalised an account by Bignell and Fortune [4] in [24]. The details of the formalisation of Bignell and Fortune's report is supplied in the Appendix to this report. We thus concentrate on describing the formalisation of the account by Fremlin of the disaster. In each case we take the formalisation as far as the start of the loss of coolant accident event (LOCA).

3.1 Fremlin's account of the accident

The report given by Fremlin of the Three Mile Island nuclear accident is given in [10, pages 138–146]. Fremlin's account is itself a distillation of a piece that appeared in *Nuclear News* of 6th April 1979.

3.1.1 Causal events in Fremlin's account

Here we formalise the temporal behavioural properties which are described in Fremlin's account. We present a series of "causal event" predicates which attempt to model the chain of events being described and give the quotes used to arrive at the formulas given. It should be expected that a formalisation at the *system level*, where we describe changes in variables representing components of the accident scenario, will allow us to prove the top-level properties listed below. It should be noted that the operating conditions of the plant, *Opconds*, forms a premise to each of the formulas below i.e. without *Opconds* holding the formulas below will not necessarily be valid. We present here a sample of the causal event formulas. The informal definition of all the individual actions is given in Table 1 and the full list of causal event actions is given in Tables 6–7 which are in the Appendix.

- $(\mathit{Enabled}(DMF) \vee \mathit{Enabled}(ASV)) \leadsto \neg \mathit{Enabled}(MFS)$

  The above is derived from the following on page 139 of [10]:-

  > At about 4AM the main feedwater system malfunctioned, apparently as a result of failure either of the demineraliser or of the air supply to an air-operated valve.

  Note that we do not specify real-time constraints such as "4AM" in our formalisation. This is discussed in the section on future work in this area.

- $(\mathit{Enabled}(MFS) \wedge \mathit{Enabled}(AFE)) \leadsto \mathit{Enabled}(AFS)$

  This resulted from the quote [10, page 140] below:-

  > With the main feedwater supply system out of operation, the auxiliary system was to have started automatically. It did not, however, because a number of manual valves in the auxiliary system had inadvertently been left closed after a test of the system in the days prior to the accident.

- $(\mathit{Enabled}(PLH) \wedge \mathit{Enabled}(ECS)) \leadsto \mathit{Enabled}(ECO)$

  We derive this from a quote [10, page 140]:-

  > Accordingly, an operator manually overrode first one of the two emergency core-cooling water injection pumps and, in a few minutes, the other, believing that already too much water had been fed in.

Action	*Description*
AFE	This denotes the auxiliary feedwater system being enabled.
AFS	The action of the auxiliary feedwater system starting.
ASV	The air-supply to an air-operated valve.
CRS	The shutting down of the chain reaction.
CTS	The continued transmission of heat from the core to the primary circuit after shutdown.
CWR	Refers to the continuous runoff of water into the quench tank.
DMF	The failure of the demineraliser.
ECS	The operation of the emergency core-cooling system.
MFS	The (correct) operation of the main feedwater system.
PIF	The fault in the pressure level indicator.
PLH	The indication of high-pressure in the primary circuit.
RVF	Indicates that the relief valve was faulty.
RVO	The relief valve opening.
SGD	The steam generators drying out.

Table 1: Actions and their informal descriptions for the Fremlin account.

Note that we are treating the operators as automatons and are not considering the range of options at their disposal; we mention in the conclusion how belief systems may be formalised by epistemic logics and we discuss how they may be integrated within our TLA framework. This will allow epistemic considerations also to be checked for cross-viewpoint consistency between reports.

- $(\neg Enabled(ECS) \wedge PPL) \leadsto LOCA$

  This says that an *LOCA* occurred as a result of the emergency cooling being shut-off by the operators and the pressure being low (*PPL*). We define the *LOCA* as consisting of zero water level and pressure. The above formula is derived from the following quote [10, page 140]:

  > Within two minutes of the emergency cooling shut-off, it appears that at the reduced pressure water in the pressure vessel began to boil into steam, lowering the water-level.

  This is as far as we take the formalisation i.e. the formal model which we develop should merely be sufficient to show that the *LOCA* eventually takes place after other actions have taken place.

It should be stressed that these causal events are properties which should be provable from the actual system-level modelling of the accident. These properties represent the descriptive flow of events in an accident report. It is necessary to show that the formalisation of such a report will lead to a consistent and complete set of provable temporal formulas. If this is not so then we have consequently detected inconsistencies or omissions in the report. Types, on the other hand, indicate information about the plant which is *independent* of the actual accident scenario. They indicate how each component of the scenario should behave. They indicate *static* properties of the system, as opposed to the dynamic properties indicated by the causal event formulas. It is possible to deduce flaws in the typing of components and operations that will indicate errors in the assumptions about the basic operation in the plant rather than inconsistencies in the reporting mechanism.

3.1.2 System modelling of the accident

Having given the top-level behaviours that we expect to be able to prove, we now give a formal specification of the accident as a concurrent computational system. We aim to eliminate any explicit idea of sequence in the formalisation. If this was desired then it could be done by introducing a variable which indicated the stage reached during the accident. This is the approach adopted by Lamport when formalising a sequence of commands in an imperative programming language. The point of *not* having an explicit idea of sequence is that we do not restrict the model to having a time-flow which is implicit in a given accident report — only some relative timings or sequences may be fully described whilst the reader is left to infer others. Instead, sequences of actions (the causal event formulas) will result as provable properties from the logic of the specification i.e. certain actions

Action	*Equivalent to*
AirDuct	{clear,water_filled}
AvailStat	{offline,online}
RodPos	{up,down}
Switch	{on,off}
TempPress	{none,low,normal,high}
ValvePos	{open,closed}
ValveStat	*WorkStatus*
WorkStatus	{working,faulty}

Table 2: Types of the formalisations.

will be enabled only once certain other actions have taken place. The causal "leads to" relations given above can thus be derived as a result of this logic, rather than due to an explicit sequencing. There is a particular point in making the sequencing implicit rather than explicit: it allows greater freedom in producing a formal specification which corresponds to the accident report and consequently enables cross-viewpoint consistency between reports to be established more easily. If we had an explicit sequencing variable then it may be impossible to unify two equivalent reports. We are therefore assuming that, in general, the sequence of events is non-deterministic, whilst the types of components and the operations upon components are completely stated and are fixed by the model. This is reasonable since the types and operations represent the knowledge and assumptions about the accident scenario at the time of its occurrence, whilst the sequencing of events is a *dynamic* product of the accident itself.

Types We use the types listed in Table 2, apart from *Bool*. Although these are clearly isomorphic to *Bool* and other finite types, we give them separate identifiers in order to aid interpretation of the formalisation of the accident account. We assume that there is some "natural ordering" on each of these finite types so that a water temperature of low is less than high. We overload the comparison operators, $<, =, \leq, >, \ldots$ so that they are used for each such ordering. In addition, it makes a restriction upon operations over certain components and as such may compared with Dearden and Harrison's method of specifying "interactors" in Modal Action Logic [8].

Variable	*Type*
ac_valves	*ValvePos*
air_circuit_status	*AirDuct*
relief_valve_status	*ValveStat*
primary_flow	*TempPress*

Table 3: Variables common to both formalisations.

Common variables used in both formalisations Table 3 shows the variables that both the Fremlin and Bignell & Fortune formalisations have in common.

Components Certain variables are unique to Fremlin's account and are shown in Table 4.

The *Init* action Here we stipulate the values of the component variables of the plant at the start of the accident in Fremlin's account. This consists of the conjunction of all the initial values before the accident started. Obviously, establishing *Init* for a given accident formalisation is a matter of gleaning information both from the accident report itself and from prior knowledge of the workings of the plant. For Fremlin's account, we get the following:

$$\begin{aligned} Init &\triangleq (water\_runoff = \mathsf{false}) \land (primary\_flow = \mathsf{normal}) \\ &\quad \land (water\_press = \mathsf{normal}) \ldots \end{aligned}$$

The Next-state action This comprises the operation of the plant together with the actions which occurred during the accident. It should be noted that such actions are not automatically enabled: some of them will only occur if the *Cat* actions (see below) occur. The actions can be partitioned into two, the disjunction of which will form $\mathcal{N}$. The first action, $\mathcal{R}$, is simply the disjunction, to denote non-deterministic concurrency of actions within the accident system, of all the actions listed in Tables 9– 11 (given in the Appendix) which are not a part of *Cat* below. The second one is *Cat* itself. Thus we have:

$$\begin{aligned} \mathcal{N} &\triangleq \mathcal{R} \lor Cat \\ \mathcal{R} &\triangleq AFE \lor AFS \lor CRS \\ &\quad \lor CTS \lor CWR \lor ECS \\ &\quad \lor MFS \lor PLH \lor RVO \lor SGD \end{aligned}$$

Variable	Type
air_supply_status	$WorkStatus$
air_valve_pos	$ValvePos$
$auxiliary\_valves$	$ValvePos$
$auxiliary\_feedwater$	$Switch$
$core\_reaction$	$Switch$
$demineralizer\_status$	$WorkStatus$
$emergency\_cooling$	$Switch$
$pressure\_indication$	$TempPress$
$pressure\_indicator\_status$	$WorkStatus$

Table 4: Variables of the Fremlin formalisation.

Note that we have included some predicates in the above, which do not actually indicate a change between states, but merely an assertion about one particular state. It is not always essential to include such predicates – the *LOCA* predicate should become true as a result of other actions, for instance – but again it does provide a complete reference for the accident analyst and a direct comparison with the original report.

The Catastrophic actions *Cat*, which denotes the catastrophic events which took place in the accident scenario, and which triggered the events leading to LOCA, consists of a conjunctive normal form made out of the contributory causes to the accident. Therefore, identifying *Cat* correctly is one of the prime objectives in the analysis of an accident account.

$$Cat \triangleq (ASV \lor DMF) \land PIF \land RVF$$

The Fairness condition The fairness condition gives equal weight to each action in the formal specification of the accident, including the catastrophic actions. There is thus a strong fairness condition for each action, both those in $\mathcal{N}$ and in *Cat*, listed in Tables 9–11, so as to give each one the same opportunity to occur.

4 Viewpoint Consistency Checking

Cross-viewpoint consistency checking for accident analysis comprises the following steps:

- Consistency checking of each account.
- Development of a unifying formal specification.
- Internal consistency checking of the unifying account.

4.1 Consistency checking of each account

The basic point of the formalisation of accident reports is to ensure that inconsistencies and omissions are avoided, which may occur with natural language reports. To ensure that each account is *internally relatively complete* (i.e. all theorems about the accident scenario are provable with respect to the assumptions made about the disaster), we must show that:

1. Each of the "causal event" formulae (see 3.1.1 above) is provable given the assumption that *Opconds* and *Enabled*(*Cat*) are valid premises. That is, for each formula C,

   $$\mathit{Opconds}, \mathit{Enabled}(\mathit{Cat}) \models C$$

2. The overall accident analysis predicate,

   $$\mathit{Opconds} \Rightarrow (\mathit{Enabled}(\mathit{Cat}) \leadsto \mathit{LOCA})$$

   is valid.

We have shown how such formulae may be proved for Bignell and Fortune's account in [24]. Such formulae are known as *eventuality properties* and are a form of liveness, except here, in accident analysis, we are showing that "something bad does happen" as a result of certain catastrophic predicates being true. In general, such formulae are proved by Lamport's *strong fairness* deduction rule (SF2 on page 22 of [16]):

$$\frac{\begin{array}{l}\langle \mathcal{N} \wedge \mathcal{B} \rangle_f \Rightarrow \langle \overline{\mathcal{M}} \rangle_{\overline{g}} \\ P \wedge P' \wedge \langle \mathcal{N} \wedge \mathcal{A} \rangle_f \Rightarrow \mathcal{B} \\ P \wedge \overline{\mathit{Enabled}(\langle \mathcal{M} \rangle_g)} \Rightarrow \mathit{Enabled}(\langle \mathcal{A} \rangle_f) \\ \Box[\mathcal{N} \wedge \neg\mathcal{B}]_f \wedge \mathrm{SF}_f(\mathcal{A}) \wedge \Box F \Rightarrow \Diamond\Box P \end{array}}{\Box[\mathcal{N}]_f \wedge \mathrm{SF}_f(\mathcal{A}) \wedge \Box F \Rightarrow \overline{\mathrm{SF}_g(\mathcal{M})}}$$

(In our proof in [24], we use a simpler version, SF1, of the above rule which is given in [16, page 22].) In the above deduction rule, $\mathcal{A}, \mathcal{B}, \mathcal{M}, \mathcal{N}$ are actions and P, P' are predicates, with P' being P with primed variables substituted for unprimed ones. Also, f and g are state functions. The barring (e.g. $\overline{\mathcal{M}}$) refers to a substitution of state functions for any hidden variables present in the formula. As already mentioned, this consistency checking for each individual account is discussed fully in [24]. We shall assume that the two accounts have been shown to be internally relatively complete.

Naturally, we shall also be concerned that the accounts are *internally relatively sound*: that is, for any formula, F, which is provable then F can be shown to be valid. To do this we require some logical model of the statements which are derivable for an accident scenario. In particular, we need to provide a set of axioms, Γ for each component of the accident which will thus restrict the formulas that can be classed as valid. This idea is discussed further below in the piece on internal consistency checking for the unifying specification. We shall assume that each of our formalised accounts has been shown to be internally relatively sound.

4.2 Development of a unifying specification

Having shown each account is internally consistent we need to produce some *unification* of the two accounts. We do this by showing the following holds:

If we have two specifications, S_1, S_2 of an accident then we can say that they are consistent formally by using:

$$S_1 C_{dev} S_2 \Leftrightarrow dev(S, S_1) \wedge dev(S, S_2) \wedge \Psi_L(S)$$

S is a specification which is a development of both S_1 and S_2 (as given by the *dev* predicate) which is *internally valid* ($\Psi_L(S)$) with respect to the formal description technique, L — in this case TLA.

This approach to cross-viewpoint consistency has been proposed in [6]. Other approaches to cross-viewpoint consistency checking have included:

- Provide a semantics for the FDT use in an appropriate logic and then perform consistency checking in that logic. Consistency can be shown if there are no contradictions in the logical representations of the specifications. See [9, 26].

- The Viewpoint Oriented Software Engineering (VOSE) approach defines a set of relations that should hold between viewpoints. Consistency checking then involves showing that these relations do hold. However, completeness is not guaranteed and the best that can be established in general is partial completeness. This approach is again described in [9].

We believe that the approach of [6] is most appropriate approach for accident analysis since it explicitly *constructs* a new specification from the two already given. Thus a new, unifying account is produced which includes issues raised in both reports.

4.3 A general methodology for accident analysis

We prescribe the following methodology for formalising the ideas of cross-viewpoint consistency for accident analysis. We suppose that, for an accident, A, we have two formal specifications S_1 and S_2, written in TLA, which correspond to natural language accounts, R_1 and R_2 respectively.

1. Need to identify those variables which occur only in one specification but not in the other. These will correspond to processes or objects which are mentioned in one account but not in the other. Let $Var(S)$ denote the set of variables declared for a specification S.

2. Produce *new* TLA formulas, S_1^h and S_2^h which are versions of the originals with certain hidden variables. Here, for example,

$$S_1^h \triangleq \exists(Var(S_2) \setminus Var(S_1)).S_1$$

   The quantification is over all variables which occur in S_2 but not in S_1.

3. Produce S so that:

$$S \Rightarrow \overline{S_1^h}$$

   and

$$S \Rightarrow \overline{S_2^h}$$

   where, if A is of the form,

$$\exists\, x_1 \ldots x_n \,.\, B$$

   then

$$\overline{A} \triangleq B(f_1/x_1, \ldots, f_n/x_n)$$

   where each f_i is some state function. Note that we should be careful to make a distinction between parts of S which *add non-determinism* (uncertainty) relative to one of the original specifications and those which are *genuine refinements* of both formalised accident scenarios. For instance, if one account says that the cause of some event was due to action A whilst another ascribes it to *either* A or B then we have no choice in our unifying model but to use the latter. However, the resulting account is less precise than the first one.

4. Check for *internal consistency* within S. This is done by showing that a predicate, $\Psi_{TLA}(S, \Gamma)$ is true for S and a certain set of assumptions, Γ. This is discussed further below.

4.4 Internal consistency

We not only have to exhibit the unifying accident specification, S, but we have to ensure that S is *internally consistent*. This will mean that S reflects only possible real-world behaviour and not nonsensical scenarios where, for example, water pumps are switched off but the water level continues to rise steadily.

Consistency checks for viewpoints in software engineering consist of showing that the specification is implementable. For some FDTs, such as LOTOS, this check is trivial since the specifications are guaranteed to be, in some sense, executable. For others, such as Z and TLA, it is possible to write formulae down which are impossible to realize.

We introduce the idea of *accident scenario realisability*. Firstly, we develop a set of axioms, Γ, in TLA, which constrains the possible behaviours of the components of the plant so that, for example, a pump cannot be simultaneously both on and off. These axioms should reflect the specification of the plants operations as laid down in regulatory and technical documents. So, for example, included within the axioms will be a formalisation of the assumption that the walls of the containment building surrounding the reactor will have a certain thickness of concrete. Moreover, it limits the possible interactions between components, so that the act of switching the high-pressure pumps off did not simultaneously lead to the control rods being removed from the core. This restriction will typically use the negation of causal formulas and the *Unchanged* predicate so that operations are restricted in scope: this is a variant of the idea of strong typing for accident components that was mentioned in Section 2.6. Consequently, we may partition Γ into two subsets, Γ_c and Γ_i, to denote the axioms which are internal to the components and those covering interactions between components, respectively. The point of this partition is to try to resolve any consistency failures. It is necessary to show that each axiom comprising Γ will be shown to hold for the specification, S. We suggest that if a failure occurs for an axiom within Γ_c then this simply suggests that realisability has failed and that it is not possible to unify the two accounts of the accident or that an error has occurred in the specification process. A failure to establish a Γ_i formula, however, may be more revealing. It is possible in this case that we may expose the fact that one of the underlying assumptions about the interactions within the system is false. In other words, if a Γ_i formula does not hold then we may have discovered some faulty reasoning that led to the accident occurring originally.

We can thus produce sets Γ_c^f and Γ_i^f which denote which formulae are not satisfied by S. If these sets are non-empty, then we can develop an internally valid account by modifying S and/or removing axioms from Γ. As we have noted above, this latter option reveals assumptions about the accident scenario which have been shown to be invalid.

4.4.1 Development of the axiom set, Γ

It is extremely important to realise that, in general, the set Γ results from the assumptions made by the analyst of the two viewpoints, possibly based on data provided by other specialists. For example, our expectation that switching off the high pressure pumps would not result in the control rods being raised would be based upon the specification of the plant given by the nuclear engineers. Therefore, we are realising the accident reports with respect to this background information which comprises Γ. Normally, however, Γ is not included within the two viewpoints. We denote the assertion of consistency by $\Psi_{TLA}(S,\Gamma)$ which means that S is shown to be internally consistent relative to the set Γ in TLA. We shall omit the *TLA* subscript since we are only using the one specification language and frequently we shall miss out Γ from the predicate where the context makes it obvious. Conversely, the predicate can be expanded to a 3-place one where we partition Γ into Γ_c and Γ_i.

Γ', the set of axioms which are preserved by S, should be included as part of the resulting formal specification of the accident, S', so as to make explicit the background assumptions about the scenario. It is formally defined by:

$$\Gamma' \triangleq \Gamma \backslash (\Gamma_c^f \cup \Gamma_i^f)$$

Also, those formulas not satisfied by S (i.e. $\Gamma_c^f \cup \Gamma_i^f$) should also be explicitly mentioned.

4.5 Consistency checking for our example

We now proceed to check the cross-viewpoint consistency of the formalisations of the two reports given in [4] and [10]. The former we shall denote as S_B and the latter as S_F.

4.5.1 Consistency checking of Accounts

We shall assume that each account has been shown to be internally relatively both sound and complete. A demonstration of how completeness may be established is given in [24].

4.5.2 Development of a unifying specification

We shall now outline how a unifying specification, which theoretically may be interpreted as a unifying natural language account, may be derived from the two reports.

Identifying common variables We have identified those variables which are common to both accounts in Table 3. Those which are unique to Fremlin's account are given in Table 4 (which we shall denote Var_F) and those which are unique to the Bignell and Fortune account are given in Table 5 (which we shall denote Var_B).

Develop new specifications Now we let,

$$S_F^h \triangleq \exists\ Var_B . S_F$$

and vice-versa for S_B. Bounded by the existential quantification will be variables such as *steam_turbine* which occur in the Bignell and Fortune account but not in the Fremlin account. Of course, these variables need to be added to the main formulas, such as S_F, as well. However, these variables will still be invariant throughout S_F.

Variable	*Type*
e_pumps	$Switch$
$control\_rods$	$RodPos$
$hpp\_enabled$	$AvailStat$
$steam\_turbine$	$Switch$

Table 5: Variables of the Bignell and Fortune formalisation.

Unifying specification development As a first step, we can *coalesce* different variables from the two accounts which perform identical functions and are of equivalent types. For instance, we can establish that $control\_rods$ (present only in the Bignell and Fortune account) and $core\_reaction$ correspond to one another. Thus, for S_F^h we can produce a state function $\overline{control\_rods}$ where

$$\begin{aligned} \overline{control\_rods} \;\triangleq\;& \mathit{if}\ (core\_reaction = \mathsf{off}) \\ & \mathit{then}\ \mathsf{down} \\ & \mathit{else}\ \mathsf{up} \end{aligned}$$

This process is important to accident analysis since helps to document formally the relationship between informal accounts.

Similarly, we can provide state functions which can mimic actions, particularly catastrophic actions which are present in one account but not the other. For example, relative to a parameter w of type $WorkStatus$,

$$\begin{aligned} \overline{demineralizer\_status}(w) \;\triangleq\;& \mathit{if}\ (w = \mathsf{working}) \\ & \mathit{then}\ \mathsf{faulty} \end{aligned}$$

which allows a possible failure of the demineralizer (given in the Fremlin account) to be resolved within the Bignell and Fortune account. This is important since in the unifying development, S, we have to include the uncertainty of whether the accident was caused by the failure of the demineralizer or by a failure in the air-supply circuit. Indeed, in the unifying account, S, the catastrophic actions will be represented by:

$$Cat_S \triangleq Cat_B \wedge Cat_F$$

where Cat_B represents the catastrophic actions of the Bignell account and Cat_F those of Fremlin. Note that we are forming the conjunction of the two sets of catastrophic actions: effectively we are saying that both must occur in the accident scenario of our unifying account.

We need also to compose actions of the two accounts to form new actions in S.

The development of these state functions is necessary to establish the fact that the unifying specification, S, is a refinement of both S_F^h and S_B^h. To establish each refinement, such as,

$$S \Rightarrow S_F^h$$

it is necessary using the deduction system of TLA and the aforementioned state functions to establish that

1. The initial conditions of $\overline{S_F^h}$ follow from the initial conditions of S.

2. Each step of S simulates a step of $\overline{S_F^h}$.

3. Fairness conditions are preserved.

Internal consistency Once we have developed S we need to establish its internal consistency, relative to a set of axioms. The set Γ of assumptions on which internal consistency should be based consists of the basic axioms about the working of the plant and thus is, essentially, a union of the full types of all the state functions that are possible within S. For example, an operation which simply opens an air-valve at the plant should not, it is hoped, cause the control rods to be lowered into the core. Thus we should have:

$$\neg(valve\_opening \hookrightarrow (control\_rods = \mathsf{down}))$$

Of course, it is infeasible to rule out every eventuality, so typically the *Unchanged* predicate is used instead to indicate that each operation only has a limited effect. For each of S, therefore, it is necessary, to prove internal consistency, that each of the axioms of Γ can be satisfied.

5 Pragmatics

Our novel application of formal methods, and Lamport's TLA in particular, raises a number of pragmatic questions. For example, we have not demonstrated that our approach can be used to guide the detailed generation of accident reports. Our focus has been on the analytical comparison of existing documents. We do, however, believe that these techniques can be of constructive use during the reporting process. For example, mathematical models are already used to analyse and simulate the physical behaviour of materials during major accidents. The Air Accident Investigation Branch used analytical models to draft its conclusions on the engine failure and passenger injuries in the Kegworth report [2]. Our novel application of mathematical specification techniques might be used in a similar manner. We have shown that an abstract notation can be used to represent and reason about the normal, $\mathcal{R}$, and abnormal, Cat, events that lead to system failure. It seems reasonable to expect that discrete mathematicians might use these models in much the same way that accident analysts currently use continuous models of physical processes.

A related question to whether these techniques can be used constructively is *how can the findings of our analysis be communicated to non-formalists.* Formal descriptions in TLA will have little meaning for many of the lawyers, forensic scientists, meteorologists etc who contribute to accident investigations. The communication of our formal models is a subject for on-going research [15]. It should be noted, however, that a number of researchers have developed temporal logic simulation tools [12]. These can be used to directly derive executable models from the descriptions presented in this paper. It is, hypothesised, that these techniques will provide direct parallels to simulation tools that currently support the continuous models mentioned above.

6 Conclusions and Future Work

6.1 Summary of the above work

We have demonstrated how the idea of cross-viewpoint consistency checking, which has been developed for software engineering, may be fruitfully applied to accident reports. In particular, we have shown how the technique may be applied when formalising accounts using the temporal logic of actions. We have prescribed how consistency checking between two views can be carried out and have demonstrated its use with two separate accounts of the Three Mile Island nuclear accident.

6.2 Future work

6.2.1 Expanding the TLA model

In the above formal representations of the accident accounts we have ignored "real-time" (as in the Three Mile Island accident starting at 4am and the emergency core coolant system starting up two minutes later) and the knowledge/beliefs/feelings of the operators. Indeed, we have treated the actions of the latter as automatic responses to the accident, rather than based upon decisions resulting from the assimilation of knowledge.

In some cases, it may prove beneficial to incorporate such real-time and *epistemic* properties into our accident analysis. We believe that the Temporal Logic of Actions provides a useful framework for integrating such additional factors. In our formalisation above we have treated the accident as a *closed system* where the events of the accident were unaffected by any outside events. Abadi and Lamport have shown how real-time considerations can be incorporated into TLA [1], by modelling time as part of an external environment. We propose that, in addition to, or separately from, this, the epistemics of operators can be integrated within the TLA specification of the accident. We have been using epistemic logic to model the beliefs of operators [14] and propose to incorporate such ideas within an overall framework for the formalisation of accident analysis. In particular, we intend to adapt Abadi and Lamport's ideas for incorporating real-time within TLA to do the same for an environment of operator beliefs.

6.2.2 Other FDTs

In the above we have concentrated upon consistency checking between different views which are represented within one Formal Description Technique (FDT), the temporal logic of actions. Whilst we believe, as remarked above, that TLA is an extremely versatile and appropriate tool for accident analysis, it is conceivable that other FDTs will be used to describe formally disaster scenarios. For instance, LOTOS has been used to describe the mistakes made with the editing interface to the Therac-25 machine [25]. Different viewpoints of an accident may lead naturally to different techniques being used, as is the case with software engineering projects where, for instance, five different types of viewpoint are presented by the RM-ODP model [18] — enterprise, information, computational, engineering and technology. We believe that TLA can be used to capture the information and computational viewpoints and we have outlined above how we think that some human factors elements, the enterprise viewpoint, can be integrated within the TLA structure. However, since other FDTs may be appropriate in some cases it will be necessary to produce mappings between different formalisms in order to show consistency between accident perspectives. (A discussion of the use of different FDTs in the ODP model is given in [5].) In future work, therefore, we intend to show how viewpoints of accidents using different FDTs may be unified. A possible problem here is the wide range of different considerations that are present within accident analysis which may not be so apparent within software engineering projects: it is unclear which FDTs may be used and for what purpose. Previous research has investigated the applicability of various formal description techniques to accident analysis [11, 13]. Another difficulty is that we have to find a common semantics into which to map each FDT [21].

6.2.3 Safety cases

We believe that formalised cross-viewpoint consistency checking can be usefully employed in the *safety cases* regime, recommended for offshore oil or gas platforms by Lord Cullen's report into the Piper Alpha disaster [7]. The notion of a safety case is that each offshore installation lists every possible source of operational error and hazard and the actions that would consequently be taken to cover any resulting emergency. Consequently, each platform should have a safety procedure for every conceivable fault that could occur. Formalisation and cross-viewpoint checking of safety cases would help make more rigorous the offshore safety regime and avoid deficiencies (including omissions and failures of comprehension) such as those highlighted in a recent television documentary [3].

6.3 Concluding remarks

We believe that the techniques that we have described will become a useful tool in future accident reporting in order to ensure that interpretations by different analysts are compatible and that the findings of accident inquiries are disseminated correctly to other parties. The complexity and dynamic nature of many systems is such that the approach we have described should make reports easier to verify with regard to the basic findings of an accident inquiry. The formal establishment of cross-viewpoint consistency between separate parts of a report will be necessary, we believe, in order to structure accident inquiry findings successfully.

Acknowledgements

We would like to acknowledge the advice and assistance given by various members of the GIST and Formal Methods and Theory groups at Glasgow University. This work was supported by UK EPSRC grant number GR/K55040.

References

[1] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In *Proceedings of a REX Real-Time Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 1–27, 1992.

[2] Air Accidents Investigations Branch, Department of Trans port. *Report On The Accident To Boeing 737-400 G-OBME Near Kegworth , Leicestershire on 8th January 1989*, number 4/90, London, United Kingdom, 1990. Her Majesty's Stationery Office.

[3] Frontline Scotland, 16/5/96. BBC Scotland television programme, May 1996. Programme on the offshore installation safety regime in the aftermath of the recommendations of the Cullen report into the Piper Alpha disaster.

[4] V. Bignell and J. Fortune. *Understanding Systems Failures.* Manchester University Press, 1984.

[5] H. Bowman, J. Derrick, P. Linington, and M. Steen. FDTs for ODP. *Computer Standards and Interfaces*, 17:457–479, 1995.

[6] H. Bowman, J. Derrick, P. Linington, and M. Steen. Cross-viewpoint consistency in Open Distributed Processing. *Software Engineering Journal*, pages 44–57, January 1996.

[7] W. Douglas Cullen. *The Public Inquiry into the Piper Alpha disaster.* HMSO, 1990. Department of Energy report into the accident which took place in July 1989.

[8] A.M. Dearden and M.D. Harrison. Risk analysis, impact and interaction modelling. Technical report, Department of Computer Science, University of York, York, YO1 5DD, UK, 1996.

[9] S. Easterbrook, A. Finkelstein, J. Kramer, and B. Nuseibeh. Coordinating distributed viewpoints: the anatomy of a consistency check. *Concurrent Engineering: Research and Applications*, 2(3), 1994.

[10] J.H. Fremlin. *Power Production: what are the risks?* Adam Hilger Ltd., 1985.

[11] C. W. Johnson, J.C. McCarthy, and P.C. Wright. Using a formal language to support natural language in accident reports. *Ergonomics*, 38(6):1265–1283, 1995.

[12] C.W. Johnson. Specifying and prototyping dynamic human-computer interfaces for stochastic applications. In J.L. Alty, D. Diaper, and S. Guest, editors, *People And Computers VIII*, pages 233–248. Cambridge University Press, Cambridge, United Kingdom, 1993.

[13] C.W. Johnson. The application of Petri Nets to represent and reason about human factors problems during accident analyses. In P. Palanque and R. Bastide, editors, *The Design, Specification And Verification Of Interactive Systems*, pages 93–112. Springer Verlag, 1995.

[14] C.W. Johnson. The epistemics of accidents. Submitted to the Journal of Human Computer Systems, 1996.

[15] C.W. Johnson. Literate specification. *Software Engineering Journal*, 1996. Accepted and to appear.

[16] L. Lamport. The temporal logic of actions. Research Report 79, DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301, USA, December 1991.

[17] J.W. Lathrop, editor. *Planning for rare events nuclear accident preparedness and management proceedings of an international workshop January 28-31 1980*, volume 14 of *IIASA proceedings series*. Oxford Pergamon, 1981.

[18] P.F. Linington. RM-ODP: the architecture. In K. Raymond and L. Armstrong, editors, *IFIP TC6 Inter. Conf. on Open Distributed Processing, Brisbrane, February 1995*. Chapman and Hall, 1995.

[19] Z. Manna and A. Pnueli. *Verification of concurrent programs: the temporal framework*. Academic Press, 1981.

[20] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Symposium on the Foundations of Computer Science*, pages 46–57. ACM, November 1977.

[21] R. Reed et al. *Specification and Programming Refinement for Communication software*. North Holland, 1993.

[22] M. Rogovin et al. A report to the commissioners and to the public (The Rogovin Report). Technical Report USNRC report NUREG/CR-1250-V, 1980. Report by the United States Nuclear Regulatory Commission Special Inquiry Group, directed by M. Rogovin.

[23] P. Starr and W.Pearman. *Three Mile Island sourcebook annotations of a disaster*, volume 144 of *Garland reference library of social science*. Garland Publishing, 1983.

[24] A.J. Telford and C.W. Johnson. Extending the application of formal methods to analyse human error and system failure during accident investigations. Technical Report TR-1996-6, Department of Computing Science, University of Glasgow, March 1996.

[25] M. Thomas. The story of the Therac-25 in LOTOS. *High Integrity Systems Journal*, 1(1):3–17, 1994.

[26] P. Zave and M. Jackson. Conjunction as composition. *ACM Transactions on Software Engineering Methodology*, 2(4):379–411, 1993.

A Formalisation details

In this appendix we present details of the formalisation of the accounts which we have elided in the main body of the paper. We believe that it is useful to give a complete presentation of the formalisation of accident reports in order to illustrate fully the structure of a formalised accident analysis system and how the ideas that have been given work in practice. For ease of reference we give the information in tabular form.

A.1 Actions in Fremlin's account

In Table 1 informal descriptions of actions taken from Fremlin's account were given.

The associated causal event formulas are spread over Tables 6–8. For each causal property which we expect to be able to derive from our detailed formalisation we give its justification by referring to Fremlin's text. In each case an action $\mathcal{A}$ in a causal formula is shorthand for $Enabled(\mathcal{A})$. This is done in the interests of space.

The formal definitions of the actions in Fremlin's account are given in Tables 9–11.

A.2 Details of the Bignell and Fortune Account

A description of the formalisation of Bignell and Fortune's account [4] of the Three Mile Island accident is given in [24]. For the sake of completeness, we shall present the formalisation details of that account here.

Table 12 gives the informal description of the actions in the account, whilst Tables 13–15 give the causal event formulas which are implied by Bignell and Fortune's report. The formal definitions of the actions are given in Tables 16–18. Finally, we present the *Init* and *Cat* components of the formalisation in Table 19.

Formula	*Justification for formula*
$(DMF \vee ASV) \leadsto \neg MFS$	*Quote*: At about 4AM the main feedwater system malfunctioned, apparently as a result of failure either of the demineraliser or of the air supply to an air-operated valve.
$(MFS \wedge AFE) \leadsto AFS$	*Quote*: With the main feedwater supply system out of operation, the auxiliary system was to have started automatically. It did not, however, because a number of manual valves in the auxiliary system had inadvertently been left closed after a test of the system in the days prior to the accident.
$MVC \Rightarrow \neg AFE$	This models the fact that, as mentioned in the quote above, if some manual valves were closed (MVC) then the auxiliary feedwater system must not have been enabled.
$(\neg MFS \wedge \neg AFS) \leadsto SGD$	This formula, which says that if neither feedwater system is operational then the steam generators will eventually dry out, is justified by the following *quote*: Without feedwater supply, the steam generators dried out, ...
$SGD \leadsto PCE$	This comes from the continuation of the above *quote*: ...resulting in a rise in the primary coolant temperature and pressure.

Table 6: Causal event formulas in Fremlin's account (part 1).

Formula	*Justification for formula*
$PCP \leadsto (CRS \wedge CTS)$	*Quote*: The rise in pressure automatically shut down the reactor, stopping the chain reaction, but the intensely radioactive core continued to give out heat.
$(CTS \wedge PCP) \leadsto RVO$	*Quote*: Within seconds the pressure of water rose enough to trigger the opening of a relief valve.
$(RVO \wedge RVF) \leadsto CWR$	*Quote*: This valve stuck in the open position, providing a continuous release of radioactive feeding water from the main steel pressure vessel into a quench tank.
$PCE \leadsto ECS$	*Quote*: The emergency core-cooling system started automatically at two minutes into the accident sequence, and began to raise the coolant level.

Table 7: Causal event formulas in Fremlin's account (part 2).

Formula	*Justification for formula*
$(CWR \wedge \neg ECS) \leadsto (PWL \wedge PPL)$	We glean from the above quote also that the primary circuit water-level must also have been low (PWL), due to the valve being open. From the quote below, we also deduce that the pressure in the primary circuit must have been low also.
$PIF \leadsto PLH$	*Quote*: Within a few minutes the level indicator for the pressuriser misbehaved and went off scale on the high side when actually the pressure was too low.
$(PLH \wedge ECS) \leadsto ECO$	*Quote*: Accordingly, an operator [a] manually overrode first one of the two emergency core-cooling water injection pumps and, in a few minutes, the other, believing that already too much water had been fed in. [a]Note that we are treating the operators as automatons and are not considering the range of options at their disposal; we mention in the conclusion how belief systems may be formalised by epistemic logics and we discuss how they may be integrated within our TLA framework.
$(\neg ECS \wedge PLH) \leadsto LOCA$	This says that an $LOCA$ occurred as a result of the emergency cooling being shut-off and the pressure being low. We define the $LOCA$ as consisting of zero water level and pressure. The formula is derived from the following *quote*: Within two minutes of the emergency cooling shut-off, it appears that at the reduced pressure water in the pressure vessel began to boil into steam, lowering the water-level.

Table 8: Causal event formulas in Fremlin's account (part 3).

Action	*Definition*
AFE	$\wedge\ (auxiliary\_valves = \mathsf{closed})$ $\wedge\ (auxiliary\_valves' = \mathsf{open})$ $\wedge\ Unchanged(Var \backslash \{auxiliary\_valves\})$
AFS	$((auxiliary\_feedwater = \mathsf{off}) \wedge (secondary\_flow \leq \mathsf{low})$ $\wedge\ (auxiliary\_valves = \mathsf{open}))$ $\Rightarrow ((auxiliary\_feedwater' = \mathsf{on})$ $\wedge (Unchanged(Var \backslash \{auxiliary\_feedwater\})))$
ASV	$\wedge\ (air\_supply\_status = \mathsf{working})$ $\wedge\ (air\_supply\_status' = \mathsf{faulty})$ $\wedge\ Unchanged(Var \backslash \{air\_supply\_status\})$
CRS	$\wedge\ (core\_reaction = \mathsf{on})$ $\wedge\ (core\_reaction' = \mathsf{off})$ $\wedge\ Unchanged(Var \backslash \{core\_reaction\})$
CTS	$\neg((core\_reaction = \mathsf{off})$ $\Rightarrow (water\_temp \leq \mathsf{normal}))$

Table 9: Actions of the Fremlin account and their formal definitions in TLA (Part 1).

Action	*Definition*
CWR	$(\mathit{water\_runoff} = \mathsf{true})$
DMF	$\wedge\ (\mathit{demineralizer\_status} = \mathsf{working})$ $\wedge\ (\mathit{demineralizer\_status}' = \mathsf{faulty})$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \setminus \{\mathit{demineralizer\_status}\})$
$LOCA$	$\wedge\ (\mathit{primary\_flow} = \mathsf{none})$ $\wedge\ (\mathit{water\_press} = \mathsf{low})$
MFS	$\mathit{feedwater\_line} = \mathsf{online}$
PIF	$\wedge\ (\mathit{pressure\_indicator\_status} = \mathsf{working})$ $\wedge\ (\mathit{pressure\_indicator\_status}' = \mathsf{faulty})$

Table 10: Actions of the Fremlin account and their formal definitions in TLA (Part 2).

Action	*Definition*
PLH	$\land$ $((water\_press = \mathsf{high})$ $\lor (pressure\_indicator\_status = \mathsf{faulty})$ $\Rightarrow (pressure\_indication' = \mathsf{high}))$ $\land$ $Unchanged(Var \setminus \{pressure\_indication\})$
RVF	$\land$ $(relief\_valve\_status' = \mathsf{faulty})$ $\land$ $Unchanged(Var \setminus \{relief\_valve\_status\})$
RVO	$\land$ $(relief\_valve = \mathsf{closed}) \land (relief\_valve' = \mathsf{open})$ $\land$ $Unchanged(Var \setminus \{relief\_valve\})$
SGD	$\land$ $(secondary\_flow > \mathsf{none})$ $\land$ $(secondary\_flow' = \mathsf{none})$ $\land$ $Unchanged(Var \setminus \{secondary\_flow\})$

Table 11: Actions of the Fremlin account and their formal definitions in TLA (Part 3).

Action	*Description*
CRD	The control rods dropping into the reactor core. (Note that this did not stop all heat generation in the core.)
CTS	The continued transmission of heat from the core to the primary circuit after shutdown.
CWR	Refers to the continuous runoff of water into the quench tank.
EVC	The fact that the emergency feedwater line valves were closed.
FSD	The feedwater line to the steam generator being closed.
HPE	Indicates that the high pressure pumps were enabled.
PCD	The action of the pumps closing down.
PCE	Represents the emergency conditions of rising temperature and pressure which occurred in the primary circuit. Here, $PCE \triangleq PCT \wedge PCP$ which represents the conjunction of the temperature (PCT) and (PCP) pressure rise.
PLP	Denotes a loss of pressure in the primary circuit.
RVF	Indicates that the relief valve was faulty.
SBD	The steam generators boiling dry.
STS	The shutdown of the steam turbine.
VSO	The valves switching off in the air circuit.
WAC	Denotes water being introduced into the air circuit.
WRO	Water runoff into the drain tank from the primary circuit.

Table 12: Actions and their informal descriptions for the Bignell and Fortune account.

Formula	*Justification for formula*
$WAC \rightsquigarrow VSO$	The above is derived from page 12 of [4] where it is said: ...they [the maintenance crew] allowed some water to enter an air circuit that opened and closed some valves ...The affected valves shut off ...
$VSO \rightsquigarrow PCD$	Continuing with the quote above: The affected valves shut off and the pumps in the same circuit closed down in quick succession.
$PCD \rightsquigarrow FSD$	*Quote*: One of the pumping circuits affected [by the valves switching off] was the feedwater line to the steam generator in the secondary circuit.
$FSD \rightsquigarrow (STS \wedge EPS)$	The above was deduced from the *quotes* below:- ...so the safety system shut down the steam turbine and the electric power generator it drove. (*Page 12*) When the pumps supplying water to the steam generator stopped, three emergency pumps for the feedwater started. (*Page 13*)
$(FSD \wedge EPS \wedge EVC) \rightsquigarrow SBD$	*Quote* [4, page 13]: ...an operator ...noticed that these pumps were running but he did not notice two particular warning lights on a control panel. They signalled that valves were closed on each of the two emergency feedwater lines and so were preventing water from reaching the steam generators, which soon boiled dry ...Not until almost eight minutes later did the operators notice the closed valves and open them.

Table 13: Causal event formulas in Bignell & Fortune's account (part 1).

Formula	*Justification for formula*
$SBD \rightsquigarrow PCE$	*Quote*: Deprived of an outlet for the heat still entering from the reactor core the water in the primary circuit increased in temperature and pressure.
$PCE \rightsquigarrow CRD \wedge RVO$	*Quote*: A relief valve on top of the pressuriser opened and steam and water began to flow out to a drain tank ...To reduce the production of heat in the reactor core the control rods automatically descended into it ...
$RVO \rightsquigarrow WRO$	This also comes from the quote above.
$(CRD \wedge WRO) \rightsquigarrow \neg PCE$	This says that the lowering of the control rods, in conjunction with the water runoff into the drain tank led to an end to the heat/temperature emergency in the primary circuit. This comes from the following on page 13 of Bignell and Fortune: The combined effects of lowering the control rods and opening the relief valve brought the temperature in the vessel down to normal ...

Table 14: Causal event formulas in Bignell and Fortune's account (part 2).

Formula	*Justification for formula*
$\neg RVF \Rightarrow ((\neg PCE \wedge WRO) \leadsto \neg RVO)$	The above is a formalisation of the idea that, provided that the relief valve is not faulty, then it will be closed if both water runoff is taking place and there is not a pressure/heat emergency in the primary circuit. This comes from the sequel to the quote above: ...at which time the relief valve should have closed.
$\neg PCE \Rightarrow (WRO \leadsto PLP)$	The above says that if there is no pressure emergency in the primary circuit then water runoff will lead to a loss of pressure. This comes from the following statements on page 14 of [4] Through this valve more than one third of the contents of the primary circuit escaped. ...two high pressure pumps started automatically, triggered by a lowering of pressure in the primary circuit.
$HPE \Rightarrow (PLP \leadsto HPP)$	This is justified by the second of the quoted lines given above.
$(PLP \wedge LDO) \leadsto LOCA$	This says that a loss of pressure in the primary circuit, together with opening the let-down valve, led to the Loss of Coolant Accident event. It comes from the following: The let-down valve was opened to release what was believed to be an excess of water. The original fall in pressure and the failure of the injection to bring down the temperature should have alerted the operators that LOCA was underway but it did not.

Table 15: Causal event formulas in Bignell & Fortune's account (part 3).

Action	*Definition*
CWR	$(water\_runoff = \mathsf{true})$
CTS	$\neg((core\_reaction = \mathsf{off}) \Rightarrow (water\_temp \leq \mathsf{normal}))$
CWR	$(water\_runoff = \mathsf{true})$
EVC	$\wedge\ ((feedwater\_line = \mathsf{open}) \wedge (feedwater\_line' = \mathsf{closed})$ $\wedge\ Unchanged(Var \backslash \{feedwater\_line\})$
FSD	$\wedge\ (feedwater\_line = \mathsf{open}) \wedge (feedwater\_line' = \mathsf{closed})$ $\wedge\ Unchanged(Var \backslash \{feedwater\_line\})$

Table 16: Actions of the Bignell and Fortune account and their formal definitions in TLA (Part 1).

Action	*Definition*
HPE	$\wedge\ (hpp\_enabled = \mathsf{offline}) \wedge (hpp\_enabled' = \mathsf{online})$ $\wedge\ Unchanged(Var \backslash \{hpp\_enabled\})$
PCD	$\wedge\ (((ac\_pumps = \mathsf{on}) \wedge (ac\_valves = \mathsf{closed})) \Rightarrow ac\_pumps' = \mathsf{off})$ $\wedge\ Unchanged(Var \backslash \{ac\_pumps\})$
PCE	$\wedge\ (water\_temp' = \mathsf{high}) \wedge (water\_press' = \mathsf{high})$ $\wedge\ Unchanged(Var \backslash \{water\_temp, water\_press\})$
PLP	$(water\_press' = \mathsf{low}) \quad \wedge \quad Unchanged(Var \backslash \{water\_press\})$
RVF	$\wedge\ (relief\_valve\_status' = \mathsf{faulty})$ $\wedge\ Unchanged(Var \backslash \{relief\_valve\_status\})$

Table 17: Actions of the Bignell and Fortune account and their formal definitions in TLA (Part 2).

Action	*Definition*
SBD	$\wedge\ ((\mathit{feedwater\_line} = \mathsf{closed})$ $\wedge\ (\mathit{e\_pumps} = \mathsf{off} \vee$ $\wedge\ ((\mathit{e\_pumps} = \mathsf{on}) \wedge (\mathit{e\_valves} = \mathsf{closed}))) \Rightarrow \mathit{secondary\_flow}' = \mathsf{low})$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \backslash \{\mathit{secondary\_flow}\})$
STS	$\wedge\ (((\mathit{steam\_turbine} = \mathsf{on}) \wedge (\mathit{feedwater\_line} = \mathsf{closed}))$ $\Rightarrow \mathit{steam\_turbine}' = \mathsf{off})$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \backslash \{\mathit{steam\_turbine}\})$
VSO	$\wedge\ ((\mathit{ac\_valves} = \mathsf{open}) \wedge (\mathit{ac\_valves}' = \mathsf{closed}))$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \backslash \{\mathit{ac\_valves}\})$
WAC	$\wedge\ (\mathit{air\_circuit\_status} = \mathsf{clear}) \wedge (\mathit{air\_circuit\_status}' = \mathsf{water\_filled})$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \backslash \{\mathit{air\_circuit\_status}\})$
WRO	$\wedge\ (\mathit{water\_runoff} = \mathsf{false}) \wedge (\mathit{water\_runoff}' = \mathsf{true})$ $\wedge\ \mathit{Unchanged}(\mathit{Var} \backslash \mathit{water\_runoff})$

Table 18: Actions of the Bignell and Fortune account and their formal definitions in TLA (Part 3).

Action	*Definition*
Init	$\wedge\ (relief\_valve = \mathsf{closed}) \wedge (primary\_flow = \mathsf{normal})$ $\wedge\ (steam\_turbine = \mathsf{on}) \wedge (e\_pumps\_status = \mathsf{online})$ $\wedge\ (e\_pumps = \mathsf{off}) \wedge (air\_circuit\_status = \mathsf{clear})$ $\wedge\ (secondary\_flow = \mathsf{normal}) \wedge (ac\_valves = \mathsf{open})$ $\wedge\ (control\_rods = \mathsf{up}) \wedge (ac\_pumps = \mathsf{on})$ $\wedge\ (water\_runoff = \mathsf{false}) \wedge (water\_temp = \mathsf{normal})$ $\wedge\ (water\_press = \mathsf{normal}) \wedge (feedwater\_line = \mathsf{open})$ $\wedge\ (relief\_valve\_status = \mathsf{working}) \wedge (hpp\_enabled = \mathsf{online})$ $\wedge\ (letdown\_valve = \mathsf{closed}) \wedge (high\_pressure\_pumps = \mathsf{off})$
Cat	$WAC \wedge LDO \wedge ESO \wedge RVF \wedge EVC$

Table 19: Top-level actions of the Bignell and Fortune account.