Stepwise Development and Verification of a Boiler System Specification

Peter Bishop and Glenn Bruns and Stuart Anderson

Abstract

The rigorous development and verification of a boiler system specification is presented. Part I shows how the boiler system controller can be developed in a series of elaboration steps in which variables that directly reflect plant conditions are replaced by variables representing sensed, communicated values. Part II shows how the safety of the system can be assessed by first verifying safety relative to some failure assumptions and then estimating the likelihood that the assumptions hold.

1 General Introduction

In attempting to demonstrate the safety of the Generic Boiler System, two main problems are faced. First, there are a wide range of possible failures that can occur. For example, the physical devices themselves can fail, sensors can fail, and sensed values can be delayed or lost in transmission. Taking careful account of all possible failures is difficult. A second problem, common to all safety-critical systems, is that absolute safety cannot be shown. One can only hope to demonstrate partial or probable safety. However, estimates of the probability of safety are hard to calculate, and it is hard to know whether one can place much confidence in them.

The approach demonstrated here addresses both of these issues. We present a stepwise approach to the development of the boiler monitoring and control system. Initially, we present an idealised controller that observes plant variables directly. Successive steps make weaker assumptions, until finally we arrive at a specification in which only sensor values received from the data communications system are observed. At each step, safety of the boiler system is maintained. In this way, failures are treated systematically.

The second part of our approach is a separation of the deterministic and probabilistic parts of the safety analysis. Safety is proved of the boiler system absolutely, under certain assumptions that are believed to nearly always hold. Next, the likelihood of these assumptions actually holding is estimated to give an overall probability of safety.

Our report has two parts. In Part I, the technique of step-wise elaboration of the boiler controller is demonstrated. In Part II, verification of safety and failure properties is shown for a boiler system model developed at a late step of elaboration. We do not present code of the boiler controller, only a specification. However, this specification is realistic in the sense that device failure and shutdown conditions are determined by values received from the data communication system.

Part 1 - Step-Wise Derivation of a Boiler System Specification

Peter Bishop, Adelard, UK

1 Introduction

This part intended to illustrate the value of a step-wise approach for the derivation and validation of a specification using the Generic Problem as an example.

The basic strategy is to start with a model of the plant. Initially we model the unconstrained behaviour of the plant where the plant may exhibit any physically feasible behaviour. We then need to identify the permissible set of safe behaviours: the safety constraints and, for the successful working of the plant, we also need to identify the normal operating constraints (see figure 1).

Given these basic definitions we now need to consider how this top-level set of requirements is elaborated into a software specification. We do this by constructing successive models of the control system which can satisfy the top-level constraints.

In considering the physical plant and equipment we need to model failures as well as ideal operation. In order to demonstrate safety at a given level we may require extra assumptions that constrain the behaviour of that model. For example, it may be necessary to assume that only a single failure will exist at any given time, or that failures are always detected. Obviously these assumptions are not always correct but we should be able to compute the probability of violating the design assumptions.

So finally we should be able to derive a specification of the software function that should satisfy the plant safety constraints provided the additional equipment-related assumptions are true. We can then estimate the probability that the equipment-related assumptions are violated in order to determine whether the system meets some quantitative safety target.

It should be noted that the main intent is to illustrate the basic step-wise approach and the reasoning behind the safety arguments. While the approach is expressed in a formal notation, it does not claim to be rigorous.

The paper will primarily consider the elaboration of the safety specification for a controller. However we shall also consider some aspects of the availability requirements which have relevance to the Generic Problem Specification.

A comparison will be made between our specification and the controller required in the Generic Problem Specification. Possible inconsistencies and omissions are discussed. We also briefly consider the analysis required to determine the probability of dangerous failure that can occur when the design assumptions are violated.

2 Formal Notation Employed

Lamport's TLA (Temporal Logic Algebra) [2] is used as the formal notation to represent the boiler and control system behaviour. TLA can specify temporal formulae such as: $x = 0 \wedge \Box x' = x + 1$. This would describe the a variable x that, on each transition step, increments indefinitely from a value of zero. The reader is referred to Part II of this paper for a description of the semantics of TLA.

2.1 Definitions and conventions

We use *ranges* to represent most values because error and uncertainties need to be taken into account. For example, a meter reading of x might be manipulated as a range $(x - err_{min}, x + err_{max})$. We also use the following notational conventions:

V_p	The physical limits of a plant variable (usually fixed).
V_m	The measured value of a plant variable.
f_d	The failure state of plant component d.
r_d	The reported (diagnosed) failure state of plant component d.

Unless otherwise specified, ranges are normally represented by a single upper-case letter. For convenience, the instantaneous value of plant variable is regarded as a range i.e. $V = (v, v)$ where v is the scalar plant value.

2.2 Range Operators

A range A is represented as a pair (x, y) in which x and y are real and $x \leq y$.

We define some operations on ranges:

$$
\begin{array}{lcl}
(x, y)_1 & \underline{\triangle} & x \\
(x, y)_2 & \underline{\triangle} & y \\
A \subseteq B & \underline{\triangle} & A_1 \geq B_1 \wedge A_2 \leq B_2 \\
A + B & \underline{\triangle} & (A_1 + B_1, A_2 + B_2) \\
A - B & \underline{\triangle} & (A_1 - B_2, A_2 - B_1)
\end{array}
$$

3 Boiler System Safety Requirement

3.1 Defining the Plant Model

The TLA notation is used define a discrete-time model of the boiler (a schematic of the plant is shown in figure 2). We can choose to interpret the transitions as taking place at

fixed time intervals. The interval between time instants is arbitrary, but to make life easier, we shall interpret the successive instants as the plant interface sampling time points (5 second intervals). This is sufficiently small to capture most boiler dynamics, and simplifies the equations used for the various levels of modelling.

In defining the plant behaviour we will use the following plant variables:

L	boiler content level
S	steam flow per unit time
D	drain flow per unit time
P	pump flow per unit time
np	number of operating pumps
t	the current time

and the following constants:

$Safe_p$	safe static boiler level range
$Safe$	safe shutdown level range (dynamic operation)
K	pump flow per pump
L_p	physical limits of boiler level
P_p	physical limits of net pump flow
S_p	physical limits of steam flow
ΔS_p	physical limits on change of steam flow / unit time

The boiler model is defined as:

$$
\begin{aligned}
Boiler \quad \underline{\triangle} \quad & L' = L + P' - S' - D' \\
\wedge \quad & P = K \cdot np \\
\wedge \quad & L \subseteq L_p \\
\wedge \quad & S \subseteq S_p \\
\wedge \quad & S' - S \subseteq \Delta S_p \\
\wedge \quad & P \subseteq P_p \\
\wedge \quad & t' = t + 1
\end{aligned}
$$

The first line of the definition is a mass balance equation where the change in level is the difference between the input and output flows. Subsequent lines specify constraints on the physical values. The final line models the passage of time for successive transitions. This is not strictly needed for the plant model, but is relevant to time-dependent definitions in the controller. It is included in the boiler model since it is a physical quantity.

For convenience, the maximum change in level L per unit time due to physical flow limitations is defined as:

$$\Delta L_p = P_p - S_p$$

Note that this definition excludes the drain valve flow D since this only appears to be used at system start-up, and there is no specification for the physical limits of the maximum drain flow.

It should be noted that Part 2 uses a reduced boiler model which excludes timing and the drain valve flow. However the models are equivalent under under the analysis assumptions made in Part 2.

3.2 Establishing the Boiler System Safety Requirement

The plant model defined above can exhibit any arbitrary behaviour permitted by its definition. In practice we wish to constrain its behaviour to meet some operational criterion. The overall safety criterion for the boiler system is:

$$\Box\, L \subseteq Safe_p$$

where L is the boiler level, $Safe_p$ is the physical range of levels for boiler safety.

At an abstract level we should be able to identify a 'constraint behaviour' that can meet this criterion. We then have to show that, given some safe initial condition, this constrained behaviour always satisfies the plant safety property.

A more restrictive set of behaviours can be defined for maintaining plant availability, but this not the major property of concern here. Some aspects of availability will be discussed later.

3.3 Identifying the Shutdown Safety Boundary

At the top level of implementation, we have to identify a suitable constraint behaviour. In the Generic Problem Specification, some form of shutdown action is taken. There is no information about how the boiler actually shuts down, but since the plant cannot shutdown immediately, we have to establish another range $Safe$ such that:

$$L \subseteq Safe \wedge \Box(Boiler \wedge ShutBehaviour) \Rightarrow \Box\, L \subseteq Safe_p$$

Where $ShutBehaviour$ is some arbitrary shutdown behaviour. One rather unlikely example of $ShutBehaviour$ could be:

$$ShutBehaviour \;\; \underline{\triangle} \;\; \neg(L \subseteq Safe) \Rightarrow (S' = 0 \wedge P' = 0 \wedge D' = 0)$$

With this extremely idealized system, it is fairly clear that excursions will be halted one time step beyond the $Safe$ region. Since the shutdown occurs in a single time step, $Safe$ must satisfy:

$$Safe + \Delta L_p \subseteq Safe_p$$

where ΔL_p is the maximum change in level that is physically possible in one step.

In practice, the shutdown behaviour would be more complex and there would have to be a greater margin between $Safe$ and $Safe_p$, i.e.:

$$Safe + \Delta L_{Shutdown} \subseteq Safe_p$$

but the same approach should be applicable. Note that the dynamic aspects of shutdown have already been taken into account in the Generic Problem Specification, so the specified range of $Safe$ is assumed to have been verified.

3.4 Partitioning the Control System

Having identified the shutdown safety boundary, we can consider the structure of control system to implement the safety and normal operation requirements. We can divide the control system into two parts: a system which implements the plant shutdown behaviour $PlantShutdown$, and a controller unit $Shutdown$ which detects unsafe plant conditions and also performs normal plant control actions. These two components are connected by a signal *up* which is monitored by the $PlantShutdown$ system. When the *up* signal is false the boiler shutdown behaviour is triggered.

We can now consider these two systems separately. In particular, we can construct a sub-model containing the $Boiler$ model and the $ShutDown$ model for which we wish to show that:

$$\Box\, up \;\Rightarrow\; L \subseteq Safe$$

Or in other words, whenever the signal true, there is a safe level in the boiler. Provided *up* is true initially, this specification, combined with the $PlantShutdown$ specification should ensure the overall safety criterion is always satisfied.

3.5 Elaboration Strategy

We now need to elaborate the design of the controller. The requirements for normal operation are not considered, only those aspects relevant to safety, i.e. the value of *up*. The basic approach to the design is a re-definition of $Shutdown$ in a sequence of design elaborations i. In the physical system hardware failures can occur, so we wish to ensure the system is *fail-safe* i.e. shutdown will occur if the boiler limits are exceeded or if the controller fails. so we define the abstract signal *up* to be implemented by:

$$up \;\; \underline{\triangle} \;\; up_i \wedge ok_i$$

Where variable ok_i is used to represent an acceptable physical failure status. and up_i represents the safety limit check implemented in terms of information available at that level of elaboration. The implementation $BSys_i$ is defined as:

$$BSys_i \;\; \underline{\triangle} \;\; (up_i \wedge ok_i) \wedge \Box(Boiler \wedge ShutDown_i)$$

In producing that implementation we may also make a set of design assumptions $Assump_i$. In order for the implementation to satisfy the abstract safety specification we have to show that:

$$BSys_i \wedge Assump_i \Rightarrow \Box\,(up_i \wedge ok_i \Rightarrow L \subseteq Safe)$$

It will be seen that at each stage of elaboration, additional uncertainty (or 'fuzz') is introduced regarding the water level position. It will also be seen that additional assumptions have be made to accommodate the physical characteristics and failure modes of the various controller subsystems.

3.6 Elaboration 1 - Plant Measurement

Obviously in the physical implementation, measured values will be used rather than the true plant variables. This data acquisition process is modelled as:

$$\begin{aligned} Measure \;\underline{\triangle}\quad & t_m = t \\ \wedge\quad & (\neg f_l \Rightarrow L \subseteq L_m) \\ \wedge\quad & (\neg f_s \Rightarrow S \subseteq S_m) \\ \wedge\quad & (\neg f_p \Rightarrow np = \sharp i.(pm_i \wedge pi_i) \end{aligned}$$

where the L_m and S_m are nominal accuracy ranges, pm_i and pi_i are independent indications of whether pump i is on, and f_l, f_s and f_p refer to the failure states of the respective sensors.

In the simplest case we can just take the measured level L_m. In this case this it is clear that if $up_1 = L_m \subseteq Safe$ then:

$$(\neg f_l \wedge up_1) \Rightarrow L \subseteq Safe$$

If $ok_1 = \neg f_l$ then the safety requirement is met. However the controller has no absolute knowledge of device failure status: it has to rely on a reported failure status, r_l, which is based on some (unspecified) diagnosis $Diag_l$ of the measured values, i.e.:

$$ok_1 = \neg r_l$$

So we need an assumption:

$$Assump_1 \;\underline{\triangle}\; f_l \Rightarrow r_l$$

to obtain a valid value for ok_1.

Like all assumptions, this may not be valid in practice, and the chance of the assumption being violated will have be evaluated probabilistically. So the overall definition for shutdown at this elaboration is:

$$\begin{aligned} ShutDown_1 \;\underline{\triangle}\; & Measure \wedge Diag_l \\ & \wedge\; up_1 = L_m \subseteq Safe \\ & \wedge\; ok_1 = \neg r_l \end{aligned}$$

3.7 Elaboration 2 - Data Fusion

For availability reasons, the Generic Problem requires operation to be maintained for as long as is safely possible using available data. In order to achieve this we require a *data fusion* approach where of faulty measurements are diagnosed based on the consistency of the measurements. This topic is quite complex and is discussed in detail in Part 2. However the general features of the data fusion process are:

- A consistency model for measured values.
- 'Fusion' which identifies all potentially failed devices (i.e./ the values of r_l, r_s, r_p).
- A computation which utilizes the failure reports and measured values to compute a bounding range, L_c such that:

$$L \subseteq L_c$$

For convenience this whole process is termed the $FusionProcess$. With data fusion we have to assume that the failure of all the sensors are detectable. In addition, data fusion is probably impossible if there is flow through the drain valve (since it is not measured), so we assume:

$$\begin{aligned} Assump_2 \triangleq \quad & f_l \Rightarrow r_l \\ \wedge \quad & f_s \Rightarrow r_s \\ \wedge \quad & f_p \Rightarrow r_p \\ \wedge \quad & D = 0 \end{aligned}$$

It is assumed that the fusion process can be characterized by a variable $FusionOK$, such that:

$$FusionOK \Rightarrow L \subseteq L_c$$

$FusionOK$ will be false when it is impossible to compute L_c. One example of this is when the level measurement is initially faulty ($t = 0 \wedge r_l$), in this case there is no basis for extrapolation using the steam and pump flow measurements.

So the overall shutdown definition is:

$$\begin{aligned} ShutDown_2 \triangleq \quad & Measure \wedge FusionProcess \\ \wedge \quad & up_2 = L_c \subseteq Safe \\ \wedge \quad & ok_2 = FusionOK \end{aligned}$$

3.8 Elaboration 3 - Effect of Communications

The communications system can corrupt a message, so correct measurements are only received when there are no communications failures, i.e.:

$$Comms \triangleq \neg f_x \Rightarrow Measure$$

where f_x is the message corruption state. We assume there is sufficient redundancy in the message to perform a diagnosis of a faulty message, i.e.:

$$f_x \Rightarrow r_x$$

where r_x is the diagnosis.

Loss of messages effectively increases the time between measurement samples, and hence the uncertainty in the current value of the level. In order to guarantee safety, we require an upper limit on the interval between valid messages, Δt_x, otherwise the worst case change in level cannot be predicted, hence.

$$ok_3 \triangleq ok_2 \wedge (t'_m - t) \leq \Delta t_x$$

where t_m is the time of the last received message and t is the current time.

To detect the existence of valid messages, we need an extra assumption for this level:

$$\begin{aligned} Assump_3 \triangleq\ & Assump_2 \\ & \wedge\ f_x \Rightarrow r_x \end{aligned}$$

Taking the worst case changes permitted by the physical constraints, the range of possible values for the level will expand by: $\Delta L_p \cdot \Delta t_x$ where L_p is the maximum possible change in level per unit time. Thus the definition of the shutdown function is:

$$\begin{aligned} ShutDown_3 \triangleq\ & Comms \wedge FusionProcess2 \\ \wedge\ & up_3 = L_c + \Delta L_p.\Delta t_x \subseteq Safe \\ \wedge\ & ok_3 = ok_2 \wedge (t_m - t) \leq \Delta t_x \end{aligned}$$

$FusionProcess2$ is an extension of the original fusion process which takes account the variable communication delays.

3.9 Elaboration 4 - Computer System Hardware

Obviously computer hardware failures (represented by f_h) will affect the safety behaviour. In terms of the available information the system has to rely on r_h, the reported hardware failure status. Thus we can define

$$ok_4 \triangleq ok_3 \wedge \neg r_h$$

Hence we need to assume:

$$Assump_1 \triangleq f_h \Rightarrow r_h$$

in order to ensure that ok_4 is valid.

So at this level of the elaboration, the controller is defined as:

$$\begin{aligned} ShutDown_4 \triangleq\ & \neg f_h \Rightarrow (Comms \wedge FusionProcess) \\ \wedge\ & up_4 = L_c + \Delta L_p.\Delta t_x \subseteq Safe \\ \wedge\ & ok_4 = ok_3 \wedge \neg r_h \end{aligned}$$

3.10 Summary of the Elaboration Sequence

The following table shows the evolution of the design. For each implementation i, $(ok_i \wedge up_i)$ is necessary to avoid shutdown.

Model	*ok*	*up*	**Assumptions**
Ideal	**true**	$L \subseteq Safe$	
$ShutDown_1$ (measurement)	$\neg r_l$	$L_m \subseteq Safe$	$f_l \Rightarrow r_l$
$ShutDown_2$ (+fusion)	$FusionOK$	$L_c \subseteq Safe$	$f_l \Rightarrow r_l$ $f_s \Rightarrow r_s$ $f_p \Rightarrow r_p$ $D = 0$
$ShutDown_3$ (+comms)	$ok_2 \wedge (t_m - t \leq \Delta t_x)$	$L_c + \Delta L_p \cdot \Delta t_x \subseteq Safe$	$Assump_2$ $\wedge f_x \Rightarrow r_x$
$ShutDown_4$ (+computer)	$ok_3 \wedge \neg r_h$	$L_c + \Delta L_p \cdot \Delta t_x \subseteq Safe$	$Assump_3$ $\wedge f_h \Rightarrow r_h$

It should be noted that further work is required to complete the elaboration. In particular, the chain of communication of the up_i signal to some physical actuation system has not been addressed.

4 Availability Considerations

In practice, a controller that simply shuts down is of little practical use. So far there has been no consideration of the behaviour under normal control conditions ('bang-bang' control). To work indefinitely, excursions must be prevented by making the level move in the opposite direction (or at least stabilising the level) using the pumps. Inspection of the engineering data shows that, by switching all four pumps on and off, level reversal (or at least level stabilisation) is possible. This capability permits the bang-bang controller to maintain the level with some nominal operating range, provided there is no flow out of the dump valve. The status of the dump valve in normal operation is not stated in the Generic Problem Specification, but it clearly must be closed or flow reversal cannot be guaranteed at the lower safety limit.

5 Impact of Plant Failures

We now have to consider whether this control behaviour can prevent excursions when there are plant failures. The only physical plant failure is the failure of a pump, which can either be 'stuck-on' or 'stuck-off'. It can be shown from the engineering data that 'bang-bang' control can only be maintained indefinitely if there are no 'stuck-on' pumps and only one 'stuck-off' pump. What this means in physical terms is that if a pump is stuck-on and there is a low steaming rate, the water level keeps on rising until it hits the upper safety limit. Similarly if there are two stuck-off pumps, the pump flow may not keep up with the steaming rate so the water level can drop below the lower safety limit.

6 Comparison with the Generic Problem Specification

The Generic Problem Specification identifies a number of states for the software implementation. There are transition conditions between the states and certain specific control actions are required in each state. In order to make a comparison, we analysed the transition conditions in order to determine the 'residence condition' and actions in each state. The residence condition is essentially the union of all the entry conditions, with all possible exit conditions removed. One feature that was noted when analysing the state transition diagram was the variation of entry conditions for the same node when entered from different source nodes. For instance, the 'degraded mode' can be entered with all sensors working from the 'normal mode', but with a sensor missing when entered from 'emergency mode'.

The following features were observed this analysis:

- there are 'health checks' on the hardware which can prevent normal operation altogether. This is similar to our initial condition, where the system must be $ok \wedge up$ before the system can operate.

- If the checks are satisfactory, the safety limit check (similar to our up_i) is applied in all operational states using different data fusion methods. Limit excursions cause shutdown.

- Assuming there is no limit violation, normal control actions are performed.

- There are also transitions based on pump availability. This is directly related to our availability analysis which shows that three pumps are required to maintain availability.

- The 'emergency' mode relates to the loss of the main level measurement, so some form of data fusion is needed to calculate the current level.

- The check on loss of communication $(t_m - t)$ enforces the requirement about the

maximum interval between messages.

6.1 Potential Problems

In reviewing this specification, a number of potential problems have been identified.

- There is no recognition of the fact that a pump can be 'stuck-on' since the Generic Problem specification only includes fully usable pumps. As shown in the earlier availability analysis, continued availability cannot be guaranteed with a stuck-on pump. It might therefore be desirable to prevent start-up or immediately shut down when the condition is identified (as it does when there are less than three pumps working).

- There is no information about what happens to the drain valve during operation. In order to maintain safety and availability it is essential for the drain valve to be shut. There is no explicit control of this valve by the computer. Ideally it should only be possible to open it in the 'system test mode', but nothing is specified.

- The variation in the node entry conditions may indicate an inconsistency about when to continue operating in the event of failures. Ideally there should be a set of invariant conditions for the safety of the plant after initialisation. The only aspect that might need to vary between modes is the operating status of the pumps.

7 Probabilistic Analysis

From the design assumptions and the modes of operation, we can construct a tree of failures leading to a dangerous failure (i.e. a failure to shut down when the level is potentially outside the safe limits). The actual structure of the tree will vary with the approach to data fusion. Figure 3 shows an example tree where the fusion relies on the level until it fails, then it uses the steam and pump flow measurements as an alternative. Provided the deterministic reasoning is correct, the top event can only occur when an assumption is violated. In our case, all these events are undetected failures, since detected failures can be rendered safe.

Provided we can assign probabilities to these base events, we should be able to compute the probability of the top event. To determine the probabilities we need to quantify:

- the failure modes and failure rates of the sensors and the other hardware systems.
- the detection efficiency of the diagnostic algorithms.
- the repair time for reported failures.
- potential sources of common cause failure.

While no data are provided to determine these base probabilities, some qualitative observations can be made.

(1) There are many sources of common cause failure within the basic architecture because the control and shutdown functions both rely on L_c. A better system design would have made the shutdown and control systems independent (e.g. independent sensors and hardware) so that there would be a higher probability of trapping control excursions (i.e. there is an AND at the top of the failure tree).

(2) The communications system is one potential source of failure that affects all measured variables and yet the specified protocol has no simple way of checking the integrity of the message. Reliance has to be placed on application-specific knowledge (e.g. of the message contents, and credible values for variables). It would have been better if some standard message integrity check could be incorporated (e.g. CRC-32) whose diagnostic efficiency is easily calculated for any specified level of 'noise'.

(3) The level measurement is obviously very important since it is the main line of defence; the other variables are only significant as standbys or consistency checkers. Safety and availability could be improved by the use of multiple level sensors.

8 Conclusions

The stepwise approach to the design allows the system to be reasoned about at a number of different levels. It allows the software specification to be related to specific features of the design which impinge on safety, and there is a logical connection with the probabilistic safety analysis.

With regard to the Generic Problem Specification itself, many of the features derived by the stepwise analysis are analogous to features in the Problem Specification. However it was noted that certain aspects of the plant that could affect plant safety were not addressed, notably the control of the drain valve, and the treatment of 'stuck-on' pump failures.

From a more pragmatic viewpoint, I feel that the application of such an approach on a real system would encourage designs that are simpler to analyse and directly address the top-level safety objective. For example, in the present design, reliance is placed on a knowledge of boiler dynamics in order to diagnose faults in the measurement devices, so the controller sub-model has to incorporate a boiler model for analysis purposes. An alternative controller design with internal measurement diagnostics (e.g. multiple level sensors) could be treated as a standalone entity, which would simplify the model and avoid the analysis complexities inherent in data fusion.

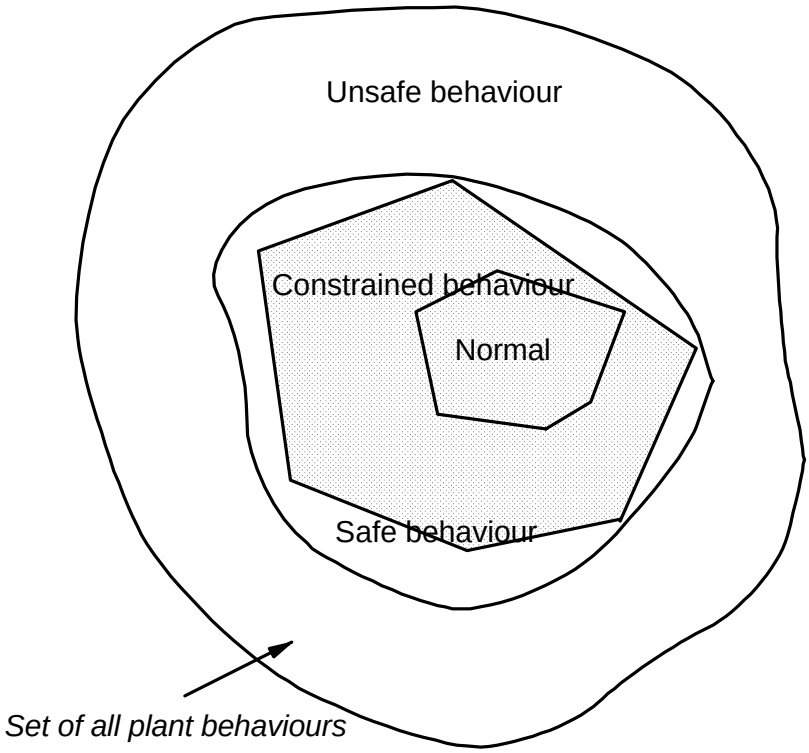

Fig 1. Behaviour of Hazardous Plant

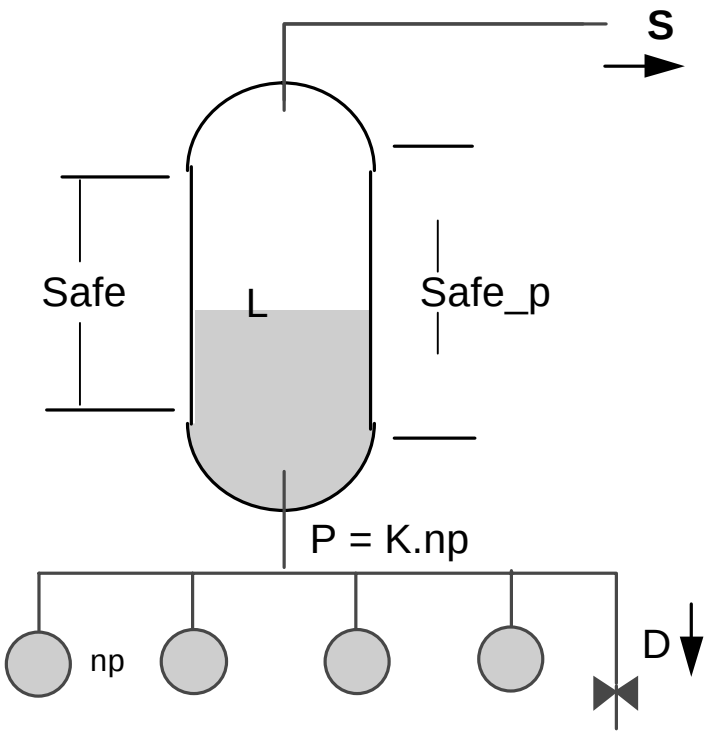

Fig 2. Generic Problem Plant Model

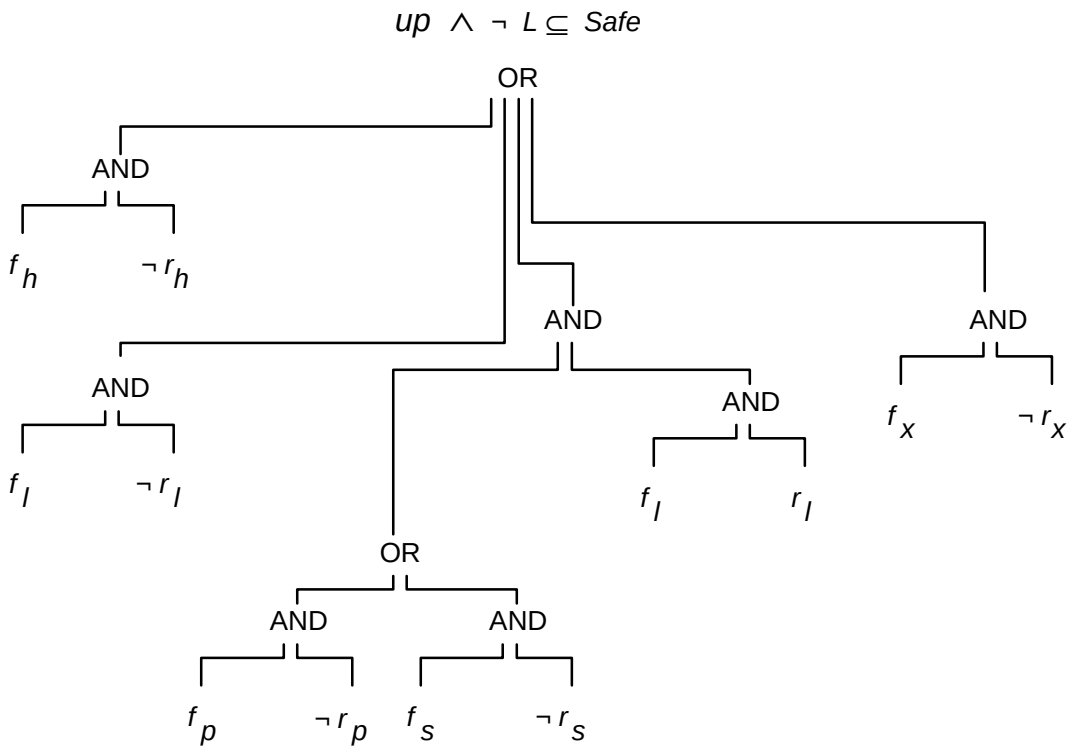

Fig 3. Failure Tree for Probabilistic Evaluation

Part II — Verifying Properties of a Boiler System

Glenn Bruns and Stuart Anderson
Laboratory for Foundations of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, UK

1 Introduction

To rigorously show that a safety-critical system is safe, one would like to take advantage of the many existing system verification techniques. Unfortunately, this cannot be done, at least directly, because no real system is absolutely safe — there is always at least a remote chance that a component will fail. However, it is not necessary to reject logical techniques in favour of wholly probabilistic ones. In this paper we show that the safety of systems *can* be proved, relative to failure assumptions that hold only in a probabilistic sense.

We demonstrate our approach by proving safety and other properties of a generic boiler system [**?**], composed of a boiler, feed pumps, sensors, and a monitoring and control system. We formally specify the plant and monitoring sub-systems, and show that the system as a whole is safe relative to failure assumptions that state, for example, that if all monitored values are consistent, in a precise sense, then no device has failed. We also show some important failure reporting properties of the system.

2 Notation

We will use Lamport's Temporal Logic of Actions (TLA) to describe both the boiler system and its properties. We briefly describe TLA here; for more details see [**?**].

The atomic formulas of TLA are *predicates* and *actions*. A predicate is a boolean expression build from variables and values, such as $x > 1$. A predicate can be regarded semantically as a function from states to booleans, where a *state* is a mapping from variables to values. An action is a boolean expression built up from variables, primed variables, and values, such as $x' = x + 1$. An action can be regarded semantically as a relation on states, in which primed variables refer to a "new" state and unprimed variables to an "old" state. Thus, $x' = x + 1$ holds between two states if the value of x in the new state is one greater than the value of x in the old state.

The syntax of TLA formulas is as follows, where P ranges over predicates and $\mathcal{A}$ ranges over actions:

$$F ::= P \mid \Box\mathcal{A} \mid \neg F \mid F_1 \wedge F_2 \mid \Box F$$

A TLA formula is *valid* if it is satisfied by every infinite sequence of states. Predicate P is satisfied by sequence σ if P holds for the first state of σ. Action $\mathcal{A}$ is satisfied by σ if $\mathcal{A}$ holds for the first pair (s_0, s_1) of states in σ. Formula $\neg F$ is satisfied by σ if F is not satisfied by σ. Formula $F_1 \wedge F_2$ is satisfied by σ if both F_1 and F_2 are satisfied by σ. Finally, $\Box F$ is satisfied by σ if F is satisfied by every suffix of σ.

Systems are always represented in TLA by formulas of the form $P \wedge \Box\mathcal{A}$, where P represents an initial condition. For example, $(x = 0) \wedge \Box(x' = x + 1)$ represents a system in which x is initially 0 and is incremented in every successive state. To express the correctness condition that a property F holds of a system *Sys*, we write $Sys \Rightarrow F$. For example, letting *Sys* be the formula $x = 0 \wedge \Box(x' = x + 1)$, we write $Sys \Rightarrow \Box(x' > x)$ to express that x increases in every successive state of *Sys*.

Some basic TLA proof rules, taken from [**?**], are listed in Appendix **??**.

In modelling the boiler system it is convenient to have variables that represent a range of values. For example, we model the measured boiler level as a range that takes the possible measurement error into account. All model variables of type range have names beginning with an upper-case letter. Formally, a range is a pair (x, y) in which x and y are real and $x \leq y$. We define some operations on ranges:

$$
\begin{array}{rcl}
(x, y)_1 & \stackrel{\text{def}}{=} & x \\
(x, y)_2 & \stackrel{\text{def}}{=} & y \\
A \subseteq B & \stackrel{\text{def}}{=} & A_1 \geq B_1 \text{ and } A_2 \leq B_2 \\
A \cup B & \stackrel{\text{def}}{=} & (\min(\{A_1, B_1\}), \max(\{A_2, B_2\})) \\
A + B & \stackrel{\text{def}}{=} & (A_1 + B_1, A_2 + B_2) \\
A - B & \stackrel{\text{def}}{=} & (A_1 - B_2, A_2 - B_1) \\
|A| & \stackrel{\text{def}}{=} & A_2 - A_1
\end{array}
$$

Notice that ranges are closed under the $-$ operator. We will later use the fact that $+$ and $-$ are monotonic with respect to $\subseteq$, so that $A \subseteq B \Rightarrow (A + C) \subseteq (B + C)$, and similarly for the $-$ operator.

For convenience, certain plant variables that really represent scalar values are also represented as ranges of the form (x, x).

3 Modelling the Boiler System

We now present a formal model of the boiler system, based on [**?**]. Roughly, our model contains a part that models the physical plant and a part that specifies a plant monitor. The plant monitor does not explicitly contain modes, but does perform the failure detection and shutdown condition checking needed for passing between normal, degraded,

emergency, and shutdown modes of operation. We model sensors and sensor failure, but do not model data communications. Also, we do not model the boiler control scheme; the safety properties we prove hold for any control scheme.

We begin by describing the model variables. Our specification contains the following plant variables:

L	boiler content level
S	steam flow per unit time
np	number of operating pumps
L_m	metered content level
S_m	metered steam flow rate
pi_i	motor on/off indicator for pump i
pm_i	monitor for pump i

the following control variables:

L_c	calculated boiler content level
S_c	calculated steam rate
P_c	calculated net pump rate
f_d	actual failure of device d
r_d	reported failure of device d
c_D	consistency of readings from devices in set D
up	shutdown variable

and the following constants:

L_p	physical limits of boiler level
S_p	physical limits of steam flow
P_p	physical limits of net pump flow
$Safe$	safe boiler level range
K	pump flow per pump

The subscript d of variables f_d and r_d range over elements of $Dev \stackrel{\text{def}}{=} \{l, s, p\}$, where l stands for the content level meter, s stands for the steaming valve meter, and p stands for the pumps. In writing a consistency variable c_D, we abbreviate device sets as strings of symbols from Dev, for example, we write c_{sp} instead of $c_{\{s,p\}}$.

The boiler system model has five components, which model the boiler behaviour, shutdown behaviour, data fusion, data consistency, and level determination. The boiler

model states the relationships between physical plant variables, and also contains failure assumptions:

$$\begin{aligned} \mathit{Boiler} \stackrel{\text{def}}{=} \quad & L' = L + (P' - S') \\ \wedge \quad & P = K \cdot np \\ \wedge \quad & S \subseteq S_p \\ \wedge \quad & P \subseteq P_p \\ \wedge \quad & \neg f_l \Rightarrow L \subseteq L_m \\ \wedge \quad & \neg f_s \Rightarrow S \subseteq S_m \\ \wedge \quad & \neg f_p \Rightarrow np = \sharp i.pm_i \\ \wedge \quad & \bigwedge_{d \in Dev} \left(f_d \Rightarrow \bigvee_{D \supseteq \{d\}} \neg c_D \right) \\ \wedge \quad & \bigwedge_{D \subseteq Dev} \left(\neg c_D \Rightarrow \bigvee_{d \in D} f_d \right) \end{aligned}$$

The model is a discrete approximation of the physical plant behaviour. The first three failure assumptions state that the meters are accurate in the absence of failure. The fourth states that failure of a device d produces some inconsistency involving d. The final assumption states that every inconsistency is produced by some failed device.

We use the notation $\bigwedge_{i \in I} F_i$ rather than $\forall i \in I.F_i$ because we always quantify over finite sets, and are therefore using only simple propositional logic, not predicate logic. The notation $\sharp i.F_i$ denotes the number of i for which F_i holds.

The shutdown model determines the value of the variable up, which holds whenever the calculated boiler level is within the safe bounds:

$$\mathit{Shutdown} \stackrel{\text{def}}{=} \quad up = L_c \subseteq Safe$$

The data fusion model determines the value of failure reporting variables. The model states that if an inconsistency exists for some device set D, then all devices in D are reported as failed. This pessimistic failure reporting strategy is necessary when calculating a level range that is guaranteed to contain the actual boiler level.

$$\mathit{Fusion} \stackrel{\text{def}}{=} \quad \bigwedge_{d \in Dev} \left(r_d = \bigvee_{D \supseteq \{d\}} \neg c_D \right)$$

Notice that the failure reporting strategy is given as a function of consistency conditions only.

The consistency model states consistency conditions between device readings. Informally, c_D holds if the devices named in D give consistent values. Thus, c_l holds if the level

meter reading taken alone is consistent, i.e., if it is not outside the possible physical range. Similarly, c_{sp} holds if the steam and pump readings are consistent. Note that $\neg c_{d_1 d_2}$ does not imply $\neg c_{d_1}$ or $\neg c_{d_2}$.

$$\begin{aligned} \mathit{Cons} \stackrel{\text{def}}{=} \quad & c_s = S_m \subseteq S_p \\ \wedge \quad & c_p = \forall i.pm_i = pi_i \\ \wedge \quad & c_l = L_m \subseteq L_p \\ \wedge \quad & c_{sp} = \mathit{true} \\ \wedge \quad & c_{sl} = \mathit{true} \\ \wedge \quad & c_{pl} = \mathit{true} \\ \wedge \quad & c'_{spl} = L'_m \subseteq L_m + (K \cdot (\sharp i.pm'_i) - S'_m) \end{aligned}$$

The final component determines the estimated level variable from reported failures. The calculated level is the best estimate of the actual level given the current reported failure conditions. For example, if the level meter is not reported as failed, then the best estimate of the actual level is by the level meter.

$$\begin{aligned} \mathit{Level} \stackrel{\text{def}}{=} \quad & \neg r'_l \Rightarrow L'_c = L'_m \\ \wedge \quad & r'_l \Rightarrow L'_c = L_c + (P'_c - S'_c) \\ \wedge \quad & \neg r_s \Rightarrow S_c = S_m \\ \wedge \quad & r_s \Rightarrow S_c = S_p \\ \wedge \quad & \neg r_p \Rightarrow P_c = K \cdot (\sharp i.pm_i) \\ \wedge \quad & r_p \Rightarrow P_c = P_p \end{aligned}$$

The action *Step* represents the combined behaviour of the preceding components:

$$\mathit{Step} \stackrel{\text{def}}{=} \mathit{Boiler} \wedge \mathit{Shutdown} \wedge \mathit{Fusion} \wedge \mathit{Cons} \wedge \mathit{Level}$$

The initial conditions include the conjunct $L_c = L$, which states that we must initially know the actual level of the boiler.

$$\mathit{Init} \stackrel{\text{def}}{=} L_c = L \wedge L \subseteq \mathit{Safe}$$

The top-level boiler system description has the standard form of a TLA specification:

$$\mathit{BSys} \stackrel{\text{def}}{=} \mathit{Init} \wedge \Box \mathit{Step}$$

4 Failure Properties

Reports of device failure in the boiler system model are based on the consistency of sensor data. There are two important properties to show of the failure reports. First, if a

failure occurs it should be reported. This is critical to the proof of safety. Letting D_r be the the set of devices reported as failed, and D_f be the set of actually failed devices, the property can be formalised as follows:

$$Pessimism \stackrel{\text{def}}{=} \Box(D_f \subseteq D_r)$$

Theorem 1 *BSys* $\Rightarrow$ *Pessimism*

Proof.

$D_f = \{d \in Dev \mid f_d\}$	by definition of D_f
$f_d \Rightarrow \bigvee_{D \supseteq \{d\}} \neg c_D$	in def. of *Boiler*
$D_f \subseteq \{d \in Dev \mid \bigvee_{D \supseteq \{d\}} \neg c_D\}$	propsitional logic, set theory
$D_f \subseteq \{d \in Dev \mid r_d\}$	by definition of r_d
$D_f \subseteq D_r$	by definition of D_r

Clauses of *Step* were used to prove $D_f \subseteq D_r$, so by the deduction principle we have that $Step \Rightarrow D_f \subseteq D_r$. TLA rule STL4 then gives us that $\Box Step \Rightarrow \Box(D_f \subseteq D_r)$, and from this $BSys \Rightarrow Pessimism$ easily follows. □

The second property is that if one or more devices are reported as failed, then at least one of the reported devices must have actually failed. This property can be formalised as follows:

$$No\ False\ Alarms \stackrel{\text{def}}{=} \Box(D_r \neq \emptyset \Rightarrow \bigvee_{d \in D_r} f_d)$$

Theorem 2 *BSys* $\Rightarrow$ *No False Alarms*

Proof. By definition, $D_r = \{d \in Dev \mid r_d\}$. Expanding the definition of r_d and simplifying, we equivalently have that $D_r = \bigcup\{D \subseteq Dev \mid \neg c_D\}$. Writing the finitely many elements of the set $\{D \subseteq Dev \mid \neg c_D\}$ as $\{D_1, \ldots, D_n\}$, we have that $D_i \subseteq D_r$ and that $\bigvee_{d \in D_i} f_d$ for $1 \leq i \leq n$. Therefore, since $D_r \neq \emptyset$, $\bigvee_{d \in D_r} f_d$. □

5 Safety Properties

The most important property to show is that if the shutdown variable *up* is true, then the boiler level is within its safe bounds. This notion of safety can be formalised in TLA as the formula:

$$Safety \stackrel{\text{def}}{=} \Box(up \Rightarrow L \subseteq Safe)$$

Before proving the safety property we present a few lemmas. Of these, the third is the most important, stating that the actual boiler level is always within the calculated level range.

Lemma 1 *Step* $\Rightarrow S \subseteq S_c$

Proof. Assume that *Step* holds. If $\neg r_s$ then $S_c = S_m$. By the *Pessimism* property we know that $\neg r_s \Rightarrow \neg f_s$, and since $\neg f_s \Rightarrow S \subseteq S_m$, we have $S \subseteq S_c$. If r_s, then $S_c \subseteq S_p$, and since $S \subseteq S_p$, we have $S \subseteq S_c$. Thus $Step \Rightarrow S \subseteq S_c$. □

Lemma 2 $Step \Rightarrow P \subseteq P_c$

Proof. We use a similar argument as in the proof to the previous theorem. If $\neg r_p$, then $P_c = K \cdot (\sharp i.pm_i)$ and $np = \sharp i.pm_i$, so $P_c = P$. If r_p, then $P \subseteq P_c$ just as for r_s. □

Lemma 3 $BSys \Rightarrow \Box(L \subseteq L_c)$

Proof. To simplify the proof, we use the following derived TLA proof rule (where I is a predicate):

$$\frac{\mathcal{A} \Rightarrow Q \quad I \wedge (\mathcal{A} \wedge Q') \Rightarrow I'}{I \wedge \Box\mathcal{A} \Rightarrow \Box I}$$

To see that the rule is sound, observe that we get $\Box\mathcal{A} \Rightarrow \Box(\mathcal{A} \wedge Q')$ from the first premise by TLA rule STL4 and other rules of simple temporal logic, and that we get $I \wedge \Box(\mathcal{A} \wedge Q') \Rightarrow \Box I$ from the second premise and TLA rule INV1. Putting these together gives the conclusion.

We define R to be $(\neg r_l \Rightarrow \neg f_l) \wedge (S \subseteq S_c) \wedge (P \subseteq P_c)$. The first conjunct is a clause of *Step*, and the others were shown to be implied by *Step* in the previous lemmas, so $Step \Rightarrow R$.

We now show that $(L \subseteq L_c) \wedge (Step \wedge R') \Rightarrow (L \subseteq L_c)'$. If $\neg r'_l$, then $L'_c = L'_m$ by a clause of *Level*. Knowing $\neg r'_l$ also gives us $\neg f'_l$, by a conjunct of R', and therefore $L' \subseteq L'_m$, giving $L' \subseteq L'_c$. If r'_l, then $L'_c = L_c + (P'_c - S'_c)$. By $L \subseteq L_c$, R', and since $+$ and $-$ are monotonic on ranges, we have that $L + (P' - S') \subseteq L_c + (P'_c - S'_c)$, so $L' \subseteq L'_c$.

Applying the derived proof rule we get $(L \subseteq L_c) \wedge \Box Step \Rightarrow \Box(L \subseteq L_c)$. Furthermore, $L \subseteq L_c$ holds initially, giving $Init \wedge \Box Step \Rightarrow \Box(L \subseteq L_c)$. □

Theorem 3 $BSys \Rightarrow Safety$

Proof.

$BSys$	$\Rightarrow$	$\Box Step$	by def. of *BSys*
	$\Rightarrow$	$\Box(up \Rightarrow L_c \subseteq Safe)$	by def. of *Shutdown*
$BSys$	$\Rightarrow$	$\Box(L \subseteq L_c)$	Lemma **??**
$BSys$	$\Rightarrow$	$\Box(up \Rightarrow L \subseteq Safe)$	by TLA rule STL5 and trans. of $\subseteq$

□

6 Failure Assumptions and Consistency Conditions

We have shown some important safety properties of the boiler system relative to some failure assumptions. As mentioned in the Introduction, the probability that the assumptions hold could be estimated, allowing an overall estimate of system safety to be made. We will not perform a probabilistic analysis here, but will review the failure assumptions we have made and the related consistency conditions.

First, we have assumed that devices report values within their specified accuracy when they have not failed. Estimating the likelihood of this condition holding would probably be possible.

Second, we have assumed that failed devices report inconsistent values. Calculating the likelihood of this assumption holding is likely to be difficult, as a detailed knowledge of the likelihood and behaviour of failure modes is needed. Furthermore, the system context is relevant. For example, a meter that fails to a 0 reading will produce a consistent failure in contexts where a 0 reading is expected.

Third, we have assumed that inconsistencies arise only in the presence of failures. The difficulty of assessing this assumption depends on the particular consistency conditions adopted.

The consistency conditions play no part in the deterministic analysis, but do affect the probabilistic analysis. The conditions chosen in our model could certainly be augmented. For example, the conditions c_p and c_l could be strengthened by additionally requiring that the change in pump and level values in the last step are within a certain range.

The condition c_{spl} holds when the level meter reading is consistent with the old level reading and the calculated net flow. Note that this condition depends on values in both the current and previous states. The use of values from the current and previous state in determining consistency could be extended so that a sequence of past values is used. For example, a sequence of flow values could be recorded and tested. An important part of the design of the boiler system is the selection of consistency conditions that are easy to compute and likely to uncover device failures. Furthermore, a good consistency condition is one for which it is possible to estimate the probability that the condition fails just when some device fails.

7 Conclusions

We have presented a specification for a boiler system model and have shown that it possesses some important safety properties. The process of formalising the system properties helped to clarify some key issues. For example, should all possibly failed devices be reported, or just the the smallest set of devices having at least one failed device? The latter reduces unnecessary diagnosis and repair, but was found to be

if level meter ok then L_c := level meter reading
else if steam meter ok then S_c := steam meter reading
else S_c := physical steam limit
if pump mon's ok then P_c := sum of pump mon. readings
else P_c := sum of physical pump limits
$L_c := L_c + (P_c - S_c)$

FIGURE 1: THE LEVEL CALCULATION IN PSEUDO-CODE

inadequate to ensure the main safety property.

The process of proving the properties was also beneficial. Attempting the proofs often uncovered missing details, such as initial conditions. On the other hand, superfluous parts of the specification were found by examining the finished proofs, and noting the dependencies on the specification. For example, we had initially included a single-failure assumption in the model, and realised later, after looking at our proofs, that this assumption was not needed.

Some parts of the model are generic and could be used for other systems. The data fusion model embodies a general strategy for detecting failure from inconsistencies between measured values. We chose a pessimistic strategy that reports a device failed if a measurement from the device is inconsistent with other values. Less pessimistic strategies may be better in other contexts. For example, another strategy is to select the smallest set of devices such that some device in the set is guaranteed to have failed. Thus if the $\neg c_{spl}$ and $\neg c_{pl}$, the devices p and l might be reported as failed, but not s. Failure assumptions could also be incorporated into data fusion model, allowing a smaller set of devices to be reported as failed. For example, if $\neg c_{sp}$ and $\neg c_{sl}$, and if f_l implies $\neg f_s$ and $\neg f_p$, then one can conclude that s has definitely failed.

The level calculation strategy is also quite generic. This strategy is related to various software fault-tolerance schemes, such as recovery blocks [**?**, **?**]. The resemblance may be easier to see if the strategy is given in pseudo-code (see Figure **??**) rather than in logic. In a recovery block, alternative computations are tried until an acceptable result is delivered. Here, alternative computations are ordered according to the accuracy of the result. A specific computation is chosen according to which devices have failed.

Acknowledgements

This work was supported by SERC/IED project 1224, "Mathematically Proven Safety Systems".

Appendix A Some TLA Proof Rules

STL1 $\dfrac{F \text{ provable by propositional logic}}{F}$

STL2 $\vdash \Box F \Rightarrow F$

STL3 $\vdash \Box\Box F \equiv \Box F$

STL4 $\dfrac{F \Rightarrow G}{\Box F \Rightarrow \Box G}$

STL5 $\vdash \Box(F \wedge G) \equiv (\Box F) \wedge (\Box G)$

STL6 $\vdash (\Diamond\Box F) \wedge (\Diamond\Box G) \equiv \Diamond\Box(F \wedge G)$

INV1 $\dfrac{P \wedge \mathcal{A} \Rightarrow P'}{P \wedge \Box\mathcal{A} \Rightarrow \Box P}$

INV2 $\vdash \Box P \Rightarrow (\Box\mathcal{A} \equiv \Box(\mathcal{A} \wedge P \wedge P'))$

Appendix B Bibliography

[1] T. Anderson and P.A. Lee, editors. *Fault Tolerance: Principles and Practice*. Prentice Hall, 1981.

[2] Specification for a software program for a boiler water content monitor and control system. Institute for Risk Research, 1992.

[3] Leslie Lamport. The temporal logic of actions. Technical Report 79, Digital Systems Research Center, 1991.

[4] B. Randall. System structure for software fault tolerance. *IEEE Transactions on Software Engineering*, SE1(2), 1975.