A Logical Approach to Data Fusion *

Glenn Bruns and Stuart Anderson
Laboratory for Foundations of Computer Science
University of Edinburgh

Abstract

Safety-critical systems often incorporate fault-tolerance techniques in their design, particularly the technique of redundancy. The idea of consistency checking found in redundancy techniques can be more broadly applied by using knowledge about system parameters and their relationships. Here we present a model of *data fusion*, which detects failures and provides good estimates of plant parameters by checking sensor data for consistency. We illustrate our approach with a boiler system example, proving that the water level in the boiler is always within its safe range.

Keywords: safety-critical systems, fault tolerance, program verification, temporal logic

1 Introduction

Safety-critical systems must often avoid, detect, and tolerate failures, and so often incorporate fault-tolerance techniques in their design. In particular, redundancy is often used [1, 2, 3]. For example, in triple-modular redundancy [4], a value is obtained by taking the majority of results from three identical components. This technique provides for both the detection and tolerance of faults, provided that at most one component fails at a time.

The idea of consistency checking found in redundancy techniques can be more broadly applied by using knowledge about system parameters and their relationships. For example, if a single output falls outside its expected range, a failure must have occurred. Similarly, two outputs can be compared. If they are not related as expected, a fault must have occurred. The change of outputs over time can also be taken into account. The goal of *data fusion* is to use sensor data to detect sensor failures and to provide good estimates of process parameters.

Here we present a formal model of data fusion. The basic idea is to detect sensor failures by estimating plant parameters from sensor outputs and then

* To appear in the *Journal of Computer and Software Engineering*.

checking them for consistency. Accurate final estimates of parameters are made by combining estimates calculated from non-failed sensors. The main advantage of our approach is its flexibility. It can be used for systems without physical redundancy, and to improve the fault-tolerance of systems with physical redundancy. Another advantage is the ability to combine logical and probabilistic reasoning. By estimating the likelihood that the failure assumptions of our model hold, and then proving safety properties from these assumptions, we can calculate the likelihood that the system will fail.

We illustrate our approach with a boiler system example, proving that the water level of a boiler system is always in its safe range. In what follows we describe our data fusion model in the temporal logic TLA [5], instantiate the model for a boiler system, and prove that the resulting model satisfies certain important safety properties. We also show how triple-modular redundancy can be treated as a data fusion problem.

2 The Boiler System

The problem we consider is derived from the Canadian Institute of Risk Research generic programming competition problem [6]. The aim of the competition was to construct safe control software for the boiler described in the problem specification [7]. Figure 1 is a diagram of the boiler. Water enters the boiler vessel through four pumps, which are either fully on or fully off, and exits the vessel as steam. The boiler is instrumented with a level sensor, a steaming rate sensor, and a monitor for each pump. Additionally, there is a sensor that shows the state of each pump switch. The safety requirement given in the problem description is that the boiler level must always be within its safe range. A too-high boiler level risks damage to upstream units; a too-low boiler level risks boiler vessel fracture.

Although the competition asked for a boiler control system, the precept that safety should depend on simple, isolated components led us to design instead a boiler shutdown system. This system provides no control; it simply monitors the boiler level and signals shutdown if the level has reached its top or bottom limit. The shutdown system is designed to work correctly regardless of boiler control regime.

In the absence of sensor failure, it is simple to design a boiler shutdown system. However, the problem states that all sensors can fail, and that no properties are guaranteed of failed sensors. For example, all sensors can fail and give consistent readings suggesting that the boiler level is acceptable, while in fact the level is outside its safe limits. This example shows that, without making failure assumptions, we could not hope to design a safe shutdown system. Our approach is to assume that failures lead to inconsistencies between sensor readings. We also assume that sensors that have not failed always report correct values up to the given sensor accuracy.

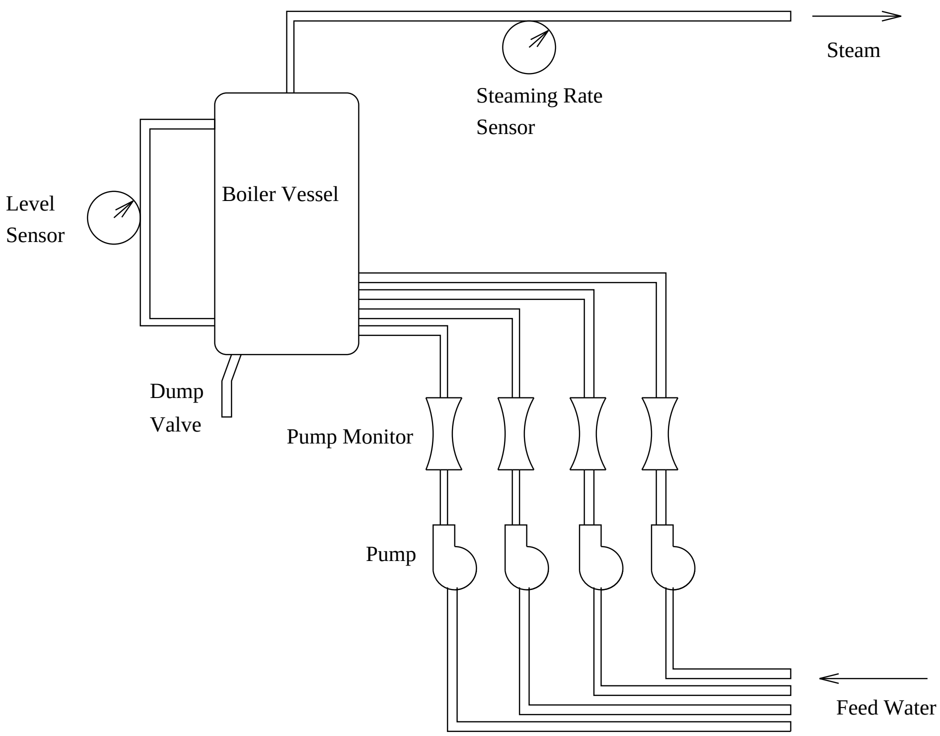

Figure 1: A Generic Boiler

Our shutdown system uses the following strategy. All sensors are periodically read. The readings are represented as intervals to take sensor accuracy into account. Estimates of boiler parameters are calculated from the readings. The estimates are then compared to discover inconsistencies, which, together with the failure assumptions, are used to report failures. Sensors not reported as failed are guaranteed to be functioning properly. From these sensors we make the most accurate estimate of water level that is sure to contain the actual water level. If the interval representing the level estimate is not within the safe interval, shutdown is signalled.

Consider a scenario in which the boiler is working normally. The sensors will report consistent values, so no failures will be reported. In these circumstances the level sensor provides the best estimate of boiler level. Suppose that the level sensor fails. By our failure assumptions it follows that some inconsistency involving the level reading must occur. If the pump and steam sensors are not reported as failed, then the boiler level can be estimated from the previous level reading and estimates of the average pump and steam rates. For example, to estimate the average pump rate, we assume that the average rate during a sample period lies between the initial and final values for that period. This assumption is valid if the sampling rate of the shutdown system is at least as frequent as the sampling rate of the control system. If the pump or steam sensors are also reported as failed, however, then our only safe choice is to assume that the maximum possible change has occurred in water level during the last period.

3 A TLA Model of Data Fusion

In this section we present a formal model of data fusion. It is a general model that can be instantiated for a particular application by supplying parameters that specify the plant parameters, the sensors, the mapping from sensors to plant parameters, and the relationships between plant parameters.

Our model is expressed in TLA – a Temporal Logic of Actions [5]. We briefly describe here the subset of TLA that we need.

The atomic formulas of TLA are *predicates* and *actions*. A predicate is a boolean expression built from variables and values, such as $x > 1$. A predicate can be viewed as a function from states to booleans, where a *state* is a mapping from variables to values. An action is a boolean expression built from variables, primed variables, and values, such as $x' = x + 1$. An action can be viewed 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:

$$F ::= P \;\Big|\; \Box\mathcal{A} \;\Big|\; \neg F \;\Big|\; F_1 \wedge F_2 \;\Big|\; \Box F$$

Formulas are interpreted relative to an infinite sequence of states. Predicate P holds of sequence σ if the first state of σ satisfies P. Formula $\Box\mathcal{A}$ holds of σ if $\mathcal{A}(s_1, s_2)$ holds of every pair (s_1, s_2) of states such that s_1 is immediately followed by s_2 in σ. Formula $\neg F$ holds of σ if F does not hold of σ. Formula $F_1 \wedge F_2$ holds of σ if both F_1 and F_2 hold of σ. Finally, formula $\Box F$ holds of σ if F holds of every suffix of σ.

Systems are represented in TLA by formulas of the form $P \wedge \Box\mathcal{A}$, where P describes the 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*.

A notational feature of TLA not used here is $[\mathcal{A}]_f$, which holds of a pair of states if either $\mathcal{A}$ holds or the value of the expression f is the same in both states. This feature is needed only for program refinement, which is not considered here. We also do not use the built-in fairness conditions of TLA, since we prove no liveness properties. Some basic TLA proof rules, taken from [5], are listed in Appendix A.

We use the notation $\bigwedge_{i \in I} P_i$, where $I \stackrel{\text{def}}{=} \{i_1, \ldots, i_n\}$ is a finite index set and P_i is a predicate, as an abbreviation that stands for $P_{i_1} \wedge \ldots \wedge P_{i_n}$ when $n > 0$, and *true* otherwise. This notation, which is not defined as part of TLA, is used instead of $\forall i \in I.P_i$ to highlight that we are quantifying only over finite sets and predicates. Similarly we use $\bigvee_{i \in I} P_i$. Finally, $\sharp i.P_i$ stands for the number of i's for which the formula P_i holds.

Some model variables, such as the variable that represents the sensed boiler level, take real intervals as values to account for measurement error. All model variables of type interval have names with an initial upper-case letter. We define intervals and interval expressions in terms of sets of reals. In the following, the meaning of an interval expression e is given as a set $[e]$ of reals:

$$\begin{aligned}
[\emptyset] &\stackrel{\text{def}}{=} \emptyset \\
[(x, y)] &\stackrel{\text{def}}{=} \{z \mid x \leq z \leq y\} \\
[A \subseteq B] &\stackrel{\text{def}}{=} [A] \subseteq [B] \\
[A \cup B] &\stackrel{\text{def}}{=} \{z \mid \exists x, y \in [A] \cup [B].x \leq z \leq y\} \\
[A \cap B] &\stackrel{\text{def}}{=} \{z \mid \exists x, y \in [A] \cap [B].x \leq z \leq y\} \\
[A + B] &\stackrel{\text{def}}{=} \{x + y \mid x \in [A] \wedge y \in [B]\}
\end{aligned}$$

$$
\begin{aligned}
[A - B] &\stackrel{\text{def}}{=} \{x - y \mid x \in [A] \land y \in [B]\} \\
[A \cdot x] &\stackrel{\text{def}}{=} \{y \cdot x \mid y \in [A]\} \\
[A/x] &\stackrel{\text{def}}{=} \{y/x \mid y \in [A]\}
\end{aligned}
$$

Notice that the interval operators $\cup$, $\cap$, $+$, $-$, and $\cdot$ are monotonic, and $/$ is monotonic for A. For example, $A \subseteq B \Rightarrow (A+C) \subseteq (B+C)$. For convenience, certain plant variables that take scalar values are represented as intervals of the form (x, x).

Now we are ready to present a general TLA model of data fusion. We begin by describing the parameters of the model.

Par	a finite set of plant parameter names
Sen	a finite set of sensor names
param	a mapping giving the parameters of each sensor
Est	a finite set of estimators

Each estimator in *Est* is a function derived from a relationship between plant parameters. Let $e(x_1 : p_1, \ldots, x_m : p_m) : p \stackrel{\text{def}}{=} \mathit{expr}$ be a function, where x_i is a variable for plant parameter p_i, and *expr* is an expression containing only monotonic interval operators, constants, and free variables $x_1, \ldots, x_m$. If the following relationship holds:

$$x_p \subseteq e(x_1, \ldots, x_m)$$

then e is an estimator of p with *arity* $p_1 \times \cdots \times p_m \rightarrow p$. If $e(x_1 : p'_1, \ldots, x_k : p'_k, x_{k+1} : p_{k+1}, \ldots, x_n : p_n) : p' \stackrel{\text{def}}{=} \mathit{expr}$ is a function where *expr* is restricted as before, and the following relationship holds:

$$x'_p \subseteq e(x_1, \ldots, x_n)$$

then e is an estimator of p' with arity $p'_1 \times \cdots \times p'_k \times p_{k+1} \times \cdots \times p_n \rightarrow p'$. We write $e : a$ if e is an estimator with arity a. Relationships of the first form represent static constraints on plant variables; those of the second form represent dynamic constraints on plant variables. For example, suppose the static constraint $\mathit{out} \subseteq 2 \cdot \mathit{in}$ holds of a system, expressing that the output is no more than twice the input. The corresponding estimator, written $e(x : \mathit{in}) : \mathit{out} \stackrel{\text{def}}{=} 2 \cdot x$, shows how the output can be estimated in terms of the input.

The variables of the data fusion model are the following:

f_s	failure status of sensor s (boolean)
r_s	reported status of sensor s (boolean)
A_p	actual value of parameter p (real interval)

M_s	measurement from sensor s (real interval)
$ES_{p,S}$	estimates of parameter p from sensor set S (set of real intervals)
FE_p	final estimate of parameter p (real interval)
I_S	inconsistent estimates from sensor set S (set of real interval pairs)

The top-level structure of the model is as follows:

$$\begin{aligned}&\mathit{Fusion}((Par, Sen, param, Est)) \stackrel{\text{def}}{=}\\ &\qquad \mathit{Assumptions} \wedge \mathit{Plant} \wedge \mathit{Estimate} \wedge \mathit{Consistency} \wedge \mathit{Report} \wedge \mathit{FinalEst}\end{aligned}$$

The formula *Assumptions* models failure assumptions. We assume that if a sensor has not failed, then the interval it reports contains the actual value of the parameter it senses. We also assume that the failure of a sensor "causes" an inconsistency between estimates, in the sense that the inconsistency arises in the estimates of a sensor set containing s, but not in the same set with s removed.

$$\begin{aligned}\mathit{Assumptions} \;\stackrel{\text{def}}{=}\; & \bigwedge_{s \in Sen} (\neg f_s \Rightarrow A_{param(s)} \subseteq M_s)\\ \wedge\; & \bigwedge_{s \in Sen} \left(f_s \Rightarrow \bigvee_{S \subseteq Sen} I_S \subset I_{S \cup \{s\}} \right)\end{aligned}$$

The reasonableness of the second assumption depends on the ways in which estimates are generated, which in turn depends on the model parameter *Est*.

The formula *Plant* models the relationships between plant variables given by parameter *Est*. The formula is of the form $F_1 \wedge \ldots F_n$, where F_i is based on estimator e_i of $Est = \{e_1, \ldots, e_n\}$. If e_i has arity $e : p_1 \times \cdots \times p_m \rightarrow p$, then F_i is

$$A_p \subseteq e(A_{p_1}, \ldots, A_{p_m}).$$

If e_i has arity $e : p'_1 \times \cdots \times p'_k \times p_{k+1} \times \cdots \times p_n \rightarrow p'$, then F_i is

$$A'_p \subseteq e(A'_{p_1}, \ldots, A'_{p_k}, A_{p_{k+1}}, \ldots, A_{p_n}).$$

The formula *Estimate* models how plant parameters can be estimated. It has the form $G \wedge H$, where the following formula G expresses that estimates can be made directly from sensor measurements:

$$\bigwedge_{p \in Par} \bigwedge_{S \subseteq Sen} \bigwedge_{s \in S} (param(s) = p) \Rightarrow M_s \in ES_{p,S}.$$

The formula H expresses that estimates can be made indirectly using estimators. H has the form $H_1 \wedge \ldots \wedge H_n$, where each H_i is based on estimator e_i of $Est = \{e_1, \ldots, e_n\}$. If e_i has arity $e : p_1 \times \cdots \times p_m \rightarrow p$, then for all parameters

p in *Par* and all sensor sets S included in *Sen*, H_i contains a conjunct of the form

$$E_1 \in ES_{p_1,S} \wedge \ldots \wedge E_m \in ES_{p_m,S} \Rightarrow e(E_1, \ldots, E_m) \in ES_{p,S}.$$

If e_i has arity $e : p'_1 \times \cdots \times p'_k \times p_{k+1} \times \cdots \times p_n \to p'$, then for all parameters p in *Par* and all sensor sets S included in *Sen*, H_i contains a conjunct of the form

$$E_1 \in ES'_{p_1,S} \wedge \ldots \wedge E_k \in ES'_{p_k,S} \Rightarrow e(E_1, \ldots, E_k, FE_{p_{k+1}}, \ldots, FE_{p_n}) \in ES'_{p,S}.$$

The final estimate variables FE_p are used for estimates of parameters in the previous state because the failure status of sensors in the set S may differ between the previous and next states.

The formula *Consistency* models estimate inconsistencies. Two estimates are taken to be inconsistent if they do not overlap. The set I_S contains all inconsistencies found between estimates possible from the sensor set S.

$$\mathit{Consistency} \stackrel{\text{def}}{=} \bigwedge_{S \subseteq Sen} I_S = \{(E_1, E_2) \mid \bigvee_{p \in Par} \bigvee_{E_1, E_2 \in ES_{p,S}} E_1 \cap E_2 \neq \emptyset\}$$

Notice that I_S is monotonic: $S_1 \subseteq S_2 \Rightarrow I_{S_1} \subseteq I_{S_2}$.

The formula *Report* models the criteria for sensor failure reporting. A sensor s is reported as failed if there are inconsistencies arising from a set S containing s, *and* if there are fewer inconsistencies arising from S itself.

$$\mathit{Report} \stackrel{\text{def}}{=} \bigwedge_{s \in Sen} r_s = \bigvee_{S \subseteq Sen} I_S \subset I_{S \cup \{s\}}$$

It is important that not all sensors found in every set S such that $I_S \neq \emptyset$ are reported because, since I is monotonic, this would mean that a single inconsistency would lead to the reporting of all sensors as failed.

The formula *FinalEst* models the final estimates of plant parameters. The final estimate of a parameter p is the intersection of all estimates that can be made from sensors not reported as failed.

$$\mathit{FinalEst} \stackrel{\text{def}}{=} \bigwedge_{p \in Par} FE_p = \bigcap ES_{p,\{s \in Sen \mid \neg r_s\}}$$

Figure 2 gives a graphical overview of the data fusion model. The boxes informally describe the types of the variables. The names to the right of the boxes give the variable names of the model. The arrows show how the values of one variable are determined by values of other variables by parts of the model.

4 Modelling the Boiler System

The boiler system model has two components. The first is an instantiation of the data fusion model; the second defines the shutdown condition from the

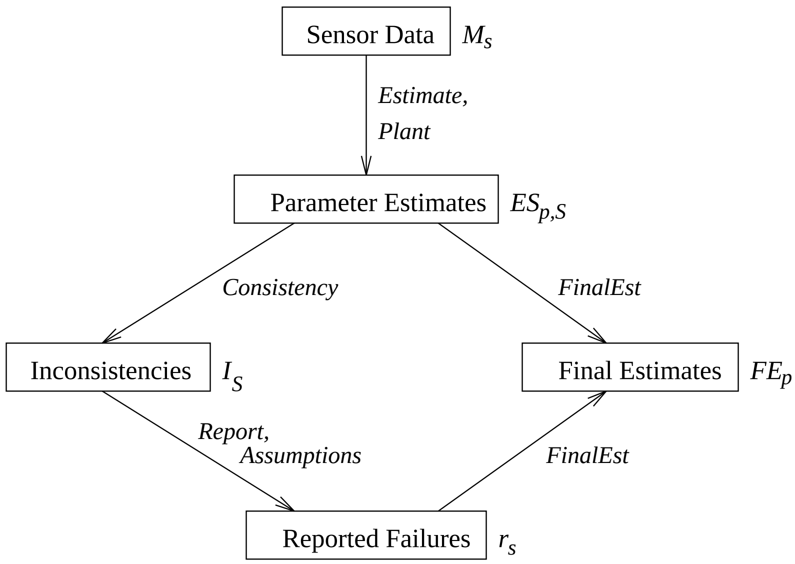

Figure 2: Variables of the Data Fusion Model

estimated boiler level. We first define the boiler-specific parameters of the data fusion component. The set *Par* contains the following plant parameter names:

l	boiler level
s	steam rate
$pump_i$	ith pump on/off ($1 \leq i \leq 4$)
p_{avg}	average pump rate
s_{avg}	average steam rate
np	number of operating pumps

The set *Sen* contains the following sensor names:

lm	level meter
sm	steam meter
pi_i	motor on/off indicator for pump i ($1 \leq i \leq 4$)
pm_i	pump monitor for pump i ($1 \leq i \leq 4$)

The function *param* maps sensors to the parameter they sense:

$$param(p) \stackrel{\text{def}}{=} \begin{cases} l & \text{if } p = lm \\ s & \text{if } p = sm \\ pump_i & \text{if } p = pi_i \quad (1 \leq i \leq 4) \\ pump_i & \text{if } p = pm_i \quad (1 \leq i \leq 4) \end{cases}$$

Notice that some parameters are not sensed directly, and that others are sensed by more than one sensor.

The set *Est* contains the following estimators:

$$
\begin{aligned}
e_1() : l &\stackrel{\text{def}}{=} L_l \\
e_2(x : l, y : pavg', z : savg') : l' &\stackrel{\text{def}}{=} x + (y - z) \\
e_3(x : np, y : np') : pavg' &\stackrel{\text{def}}{=} (x \cup y) \cdot K \\
e_4(x : s) : savg' &\stackrel{\text{def}}{=} x + L_{\delta s}/2 \\
e_5(x : \prod_{1 \le i \le 4} pump_i) : np &\stackrel{\text{def}}{=} \sharp i.(x_i = (1,1))
\end{aligned}
$$

Here the interval constant L_p is the physical limit of parameter p, K is the pump rate per pump, and $L_{\delta s}$ is the maximum possible change in steam rate during any sample period. To clarify the meaning of these estimators, we show the data fusion formula *Plant* derived from them:

$$
\begin{aligned}
Plant \stackrel{\text{def}}{=}\ & A_l \subseteq L_l \\
\wedge\ & A'_l \subseteq A_l + (A'_{pavg} - A'_{savg}) \\
\wedge\ & A'_{pavg} \subseteq (A_{np} \cup A'_{np}) \cdot K \\
\wedge\ & A'_{savg} \subseteq A_s + L_{\delta s}/2 \\
\wedge\ & A_{np} \subseteq \sharp i.A_{pump_i} = (1,1)
\end{aligned}
$$

The first conjunct simply states that the actual boiler level is always within the interval L_l. The second is derived from the mass balance equation for the boiler, which states that the water accumulated in the vessel is equal to the water pumped in less the water lost as steam. The third states that the average pump rate for a period must lie between the pump rates at the beginning and end of the period. This fact depends on the assumption that at most one change in pump state can occur during a single step. The fourth states that the average steam rate for a period can only differ from the actual steam rate at the beginning of the period by at most half the maximum change in steam rate. The final conjunct relates the number-of-pumps parameter to the individual pump parameters.

The data fusion formula *Estimate* derived from *Est* is as follows, where the conjuncts are written according to the order of the cases described in Section 3:

$$
\begin{aligned}
Estimate \stackrel{\text{def}}{=}\ & \bigwedge_{S \subseteq Sen} \\
& (lm \in S \Rightarrow M_{lm} \in ES_{l,S} \\
\wedge\ & sm \in S \Rightarrow M_{sm} \in ES_{s,S} \\
\wedge\ & pi_i \in S \Rightarrow M_{pi_i} \in ES_{pump_i,S}
\end{aligned}
$$

$$
\begin{array}{ll}
\wedge & pm_i \in S \Rightarrow M_{pm_i} \in ES_{pump_i,S} \\
\wedge & e_1 \in ES_{l,S} \\
\wedge & (\bigwedge_{1 \le i \le 4} E_i \in ES_{pump_i,S}) \Rightarrow e_5(E_1, E_2, E_3, E_4) \in ES_{np,S} \\
\wedge & E_1 \in FE_{p_l} \wedge E_2 \in ES'_{pavg,S} \wedge E_3 \in ES'_{savg,S} \Rightarrow e_2(E_1, E_2, E_3) \in ES'_{l,S} \\
\wedge & E_1 \in FE_{np} \wedge E_2 \in ES'_{np,S} \Rightarrow e_3(E_1, E_2) \in ES'_{pavg,S} \\
\wedge & E \in ES'_{s,S} \Rightarrow e_4(E) \in ES'_{savg,S})
\end{array}
$$

For notational convenience, we collect the boiler parameters into a four-tuple:

$$Boiler \stackrel{\text{def}}{=} (Sen, Par, param, Est)$$

The *Shutdown* component of the boiler system model states that the shutdown variable *up* holds whenever the estimated boiler level is within the safe bounds:

$$Shutdown \stackrel{\text{def}}{=} up = FE_l \subseteq Safe$$

The constant *Safe* is the safe boiler level interval.

The top-level boiler system description:

$$BSys \stackrel{\text{def}}{=} \Box(Fusion(Boiler) \wedge Shutdown)$$

5 Properties of the Boiler System

We will now prove some properties of the boiler system model. For all but one of these properties the proofs depend only on the general data fusion model, not the boiler-specific parameters. The first property concerns failure reporting: if a sensor fails it should be reported. This property can be formalised in TLA as follows:

$$Failures\ Reported \stackrel{\text{def}}{=} \Box \bigwedge_{s \in Sen} (f_s \Rightarrow r_s)$$

Theorem 1 $BSys \Rightarrow Failures\ Reported$

Proof. Suppose f_s. Then by a failure assumption in *Assumptions* there exists an S such that

$$I_S \subset I_{S \cup \{s\}}.$$

This is just the definition of reporting in *Report*, so r_s. Therefore, by the deduction principle, $Fusion(Boiler) \wedge Shutdown \Rightarrow \bigwedge_{s \in Sen}(f_s \Rightarrow r_s)$. TLA rule STL4 then gives $\Box(Fusion(Boiler) \wedge Shutdown) \Rightarrow \Box \bigwedge_{s \in Sen}(f_s \Rightarrow r_s)$. $\Box$

The most important property of the boiler system is that if the shutdown variable *up* is true, then the boiler level is within its safe bounds. This property can be formalised by

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

We prove *Safety* by showing two general properties of estimation in the data fusion model. First, all estimates of a parameter from non-failed sensors contain the actual parameter value. Secondly, as a simple consequence, the final estimate of a parameter always contains the actual parameter value.

Lemma 1 $\mathit{BSys} \Rightarrow \Box\left(\bigwedge_{p \in Par} \bigwedge_{S \subseteq Sen} \bigwedge_{E \in ES_{p,S}} (\bigwedge_{s \in S} \neg r_s) \Rightarrow A_p \subseteq E\right)$

Proof. By induction over the length of inference of $ES_{p,S}$ from *Fusion*. We suppose, for an arbitrary state, that $E \in ES_{p,S}$ has been deduced from *Fusion*. We then prove that A_p is contained in E, i.e. that E is a good estimate, from the induction hypothesis that all estimates deduced from shorter inferences are also good.

For the base case we have

$$M_s \in ES_{p,S},$$

where $s \in S$ and $param(s) = p$. By assumption, $\neg r_s$. By Theorem 1 we know that $\neg r_s \Rightarrow \neg f_s$, and then from the assumption about sensors in *Assumptions* we have $A_p \subseteq M_s$.

For the first inductive case we have

$$e(E_1, \ldots, E_m) \in ES_{p,S}$$

where $E_i \in ES_{p_i,S}$, for $1 \leq i \leq m$. Since $E_i \in ES_{p_i,S}$ was proved by a shorter inference, by the induction hypothesis and $\bigwedge_{s \in S} \neg r_s$ we get $A_{p_i} \subseteq E_i$ for $1 \leq i \leq m$. Then

$$A_p \subseteq e(A_{p_1}, \ldots, A_{p_m}) \subseteq e(E_1, \ldots, E_m)$$

by an inequality in *Plant* and the monotonicity of estimators.

For the second inductive case we have

$$e(E_1, \ldots, E_k, FE_{p_{k+1}}, \ldots, FE_{p_n}) \in ES'_{p,S}$$

where $E_i \in ES'_{p_i,S}$, for $1 \leq i \leq k$. As in the last case we obtain $A'_{p_i} \subseteq E'_i$ for $1 \leq i \leq k$ by the induction hypothesis. For FE_{p_i} we have

$$FE_{p_i} = \bigcap ES_{p_i,\{s \in Sen | \neg r_s\}}.$$

Suppose $E \in ES_{p_i,\{s \in Sen | \neg r_s\}}$. Since this is proved by a shorter inference, by the induction hypothesis we obtain $A_{p_i} \subseteq E$. Therefore $A_{p_i} \subseteq FE_{p_i}$ for $k+1 \leq i \leq n$. Then

$$A'_p \subseteq e(A'_{p_1}, \ldots, A'_{p_k}, A_{p_{k+1}}, \ldots, A_{p_n}) \subseteq e(E_1, \ldots, E_k, FE_{p_{k+1}}, \ldots, FE_{p_n})$$

by an inequality in *Plant* and the monotonicity of estimators. $\square$

Lemma 2 $\mathit{BSys} \Rightarrow \square\left(\bigwedge_{p \in \mathit{Par}} A_p \subseteq \mathit{FE}_p\right)$

Proof. By definition, $\mathit{FE}_p = \bigcap \mathit{ES}_{p,\{s \in \mathit{Sen} \mid \neg r_s\}}$. Lemma 1 tells us, for each E in $\mathit{ES}_{p,\{s \in \mathit{Sen} \mid \neg r_s\}}$, that $A_p \subseteq E$. Therefore $A_p \subseteq \mathit{FE}_p$. $\square$

Having shown that final estimates always contain the actual parameter value, it is easy to see that the shutdown strategy is safe.

Theorem 2 $\mathit{BSys} \Rightarrow \mathit{Safety}$

Proof. By the definition of *Shutdown*, $\mathit{up} \Rightarrow \mathit{FE}_l \subseteq \mathit{Safe}$. By Lemma 2, $A_l \subseteq \mathit{FE}_l$. Therefore, by the transitivity of $\subseteq$ and TLA rule STL5 we have $\mathit{BSys} \Rightarrow \square(\mathit{up} \Rightarrow A_l \subseteq \mathit{Safe})$. $\square$

The *Failures Reported* and *Safety* properties would be trivially satisfied by a system that reports all sensors as failed. The final property we show of the boiler model is that there are no false alarms: at least one of the sensors reported as failed must have actually failed. Letting S_r be $\{s \in \mathit{Sen} \mid r_s\}$, this property can be formalised as follows:

$$\mathit{No\ False\ Alarms} \stackrel{\text{def}}{=} \square(S_r \neq \emptyset \Rightarrow \bigvee_{s \in S_r} f_s)$$

Before proving that the boiler has this property, we show that inconsistencies can only arise from a set of sensors if some sensor in the set has failed.

Lemma 3 $\mathit{BSys} \Rightarrow \bigwedge_{S \subseteq \mathit{Sen}} \left(I_S \neq \emptyset \Rightarrow \bigvee_{s \in S} f_s\right)$

Proof. By contradiction. Assume that $I_S \neq \emptyset$ and $\bigwedge_{s \in S} \neg f_s$. By the first assumption and *Consistency* there exists a p in *Par* and E_1,E_2 in $\mathit{ES}_{p,S}$ such that $E_1 \cap E_2 = \emptyset$. By the second assumption and Theorem 1, which states that all failures are reported, $\bigwedge_{s \in S} \neg f_s$. Then by Lemma 1 we have that $A_p \subseteq E_1$ and $A_p \subseteq E_2$, so $E_1 \cap E_2 \neq \emptyset$, a contradiction. $\square$

Theorem 3 $\mathit{BSys} \Rightarrow \mathit{No\ False\ Alarms}$

Proof. By contradiction. Assume that $S_r \neq \emptyset$ and $\bigwedge_{s \in S_r} \neg f_s$. By the first assumption and *Reporting* there exists an S contained in *Sen* such that $I_S \neq \emptyset$, and therefore by Lemma 3 there exists an s in *Sen* such that f_s. By the second assumption this sensor s must not be in S_r. But this contradicts Theorem 1, which states that if f_s, then r_s and hence $s \in S_r$. $\square$

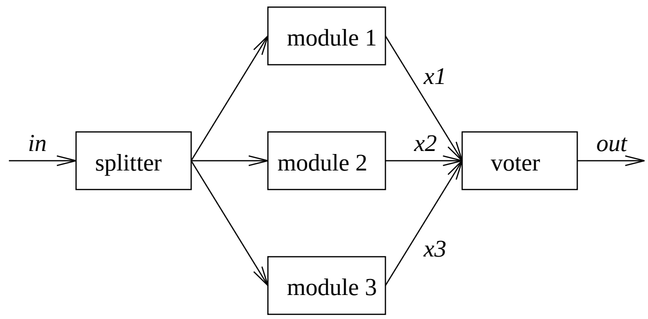

Figure 3: Triple-Modular Redundancy

6 Modelling Triple-Modular Redundancy

The data fusion model of Section 3 uses a pessimistic policy for reporting: if an inconsistency is ever introduced into a set of estimates by taking into account an additional sensor s, then s is reported as failed. This policy is compatible with the weak failure assumptions of the model. The data fusion model can be modified when stronger failure assumptions are made.

Consider the technique of triple-modular redundancy (TMR) [4]. In TMR (see Figure 3), an input (in) is sent to three modules (m_1,m_2, and m_3), which each compute a result (x_1,x_2, and x_3) and send it to a voter. The voter produces the majority value as output (out). If the voter detects a difference between the majority output and the input of module i, then a failure (r_{m_i}) is reported. Note that TMR provides both failure detection and fault tolerance.

Suppose we try to model the voter with our data fusion model. The voter input ports are taken as sensors, and the input is taken as the sole plant parameter. The data fusion parameters are then:

$$
\begin{aligned}
Sen &\stackrel{\text{def}}{=} \{s_1, s_2, s_3\} \\
Par &\stackrel{\text{def}}{=} \{in\} \\
param(s_i) &\stackrel{\text{def}}{=} in \quad \text{for } 1 \leq i \leq 3 \\
Est &\stackrel{\text{def}}{=} \emptyset
\end{aligned}
$$

Imagine that $in = 10$, that sensors 1 and 2 have not failed and therefore output 10, and that sensor 3 has failed and outputs 20. According to our data fusion model, *all* sensors are reported as failed, because

$$
\begin{aligned}
I_{s_1,s_2,s_3} &= \{10, 20\} \\
I_{s_1,s_2} &= \emptyset \\
I_{s_1,s_3} &= \{10, 20\}
\end{aligned}
$$

$$\begin{aligned} I_{s_2,s_3} &= \{10,20\} \\ I_{s_1} &= \emptyset \\ I_{s_2} &= \emptyset \\ I_{s_3} &= \emptyset \end{aligned}$$

Sensor 1 is reported because $I_{s_3} = \emptyset$ but $I_{s_1,s_3} = \{10,20\}$, suggesting that s_1 has caused a failure. It is easy to find the sets that lead to the reporting of sensors 2 and 3 as also failed.

The erroneous reporting of sensors 1 and 2 occurs because our data fusion model makes only weak failure assumptions. We can make the data fusion model work for TMR by strengthening the failure assumptions and failure-reporting conditions. We add to *Assumptions* the following conjunct, which expresses a single-failure assumption:

$$f_{s_i} \Rightarrow \bigwedge_{j\in\{1,2,3\}} \left(j \neq i \Rightarrow \neg f_{s_j}\right)$$

The formula *Report* is changed to express that a sensor s is reported failed if inconsistencies arise from all pairs of sensors containing s:

$$r_s = \bigwedge_{S\supseteq\{s\}} (|S| = 2 \Rightarrow I_S \neq \emptyset)$$

The properties *Failures Reported* and *No False Alarms* hold of this revised model. However, they hold because each estimate comes from exactly one sensor, so that every inconsistency involves exactly two sensors. Thus, these properties depend on the TMR-specific parameters of the revised model. The property $\Box(A_{in} \subseteq FE_{in})$ also holds of the TMR model. Informally, it holds because if one module fails, then by the single failure assumption the others do not, and their values must be the same correct value. Then the failed module is reported, and the others are not, causing the correct value to be output.

7 Related Work

The data fusion problem appears to be known as *analytical redundancy* in control theory [8, 9]. The main difference between the two approaches is that analytical redundancy is a probabilistic decision process, while data fusion is a logical one. In the data fusion model presented here, only sensor failure is handled, while some formulations of analytical redundancy handle sensor, component, and actuator failure. A more technical distinction is found in the notion of consistency in the two approaches. In our data fusion model, consistency is absolute: two estimates, represented by intervals, are consistent if they overlap. In analytical redundancy consistency is statistical: two estimates, represented as points in state space, are relatively consistent if the distance between them is small.

Our boiler system example is a *hybrid system* [10], as it contains interacting digital and analog devices. The only problem we faced in modelling the continuous boiler behaviour within our discrete boiler system model was to approximate the average pump and steam rates with sampled pump and steam rates. To approximate these rates we made two assumptions about the sampling period. First, that a pump can be changed at most once during any sampling period, and secondly, that the maximum change in steam rate during a sampling period is bounded by the constant $L_{\delta S}$. We made no other assumptions about the maximum time between samples and did not assume a uniform sampling rate.

The data fusion model can be compared to software fault-tolerance schemes, such as recovery blocks [11, 12]. In a recovery block, alternative computations are tried in turn until an acceptable result is obtained. Here, all alternative computations are tried, and the intersection of the acceptable results are combined to give the narrowest possible estimate. Since data fusion is also used for fault detection, it is important that all computations are performed even after an acceptable result is found.

We have mentioned that the data fusion model provides for the combination of statistical and logical methods. The likelihood that the properties of a data fusion model hold is bounded from below by the likelihood that the failure assumptions hold. The analysis of a storage management system in [13] also combines logical and probabilistic reasoning, but there failures are able to make data unavailable, but not corrupted. For example, if a processor crashes during a disk write then the half-written block is assumed to be unreadable. The correctness of the system is proved from the assumption that at least one disk is always error-free.

8 Conclusions

We have presented a general model of data fusion and have used it to prove safety properties of a generic boiler system. There are other desirable properties of the data fusion model that we have neither stated nor proved. For example, we would like the set of estimates for every parameter and sensor set to be finite and unique. We believe this property holds of our model. We would also like to show that our final estimates are optimal in the sense of being as accurate as possible in light of actual sensor failure.

In data fusion models a key issue is the choice of compatible failure assumptions and reporting conditions. For example, it was easy to prove that failures are reported in our data fusion model because the assumptions and reporting-conditions are similar. In the TMR example of Section 6 the assumptions and reporting conditions were quite different. Ideally, a data fusion model would have failure assumptions that are easy to assess, and reporting conditions that are easy to compute. Another key issue concerns the allowed failure modes of sensors. We have assumed that arbitrary failures are possible. What could be shown if we were to assume that sensors have fail-stop [14] behaviour?

We claim that our approach to data fusion allows logical and probabilistic reasoning to be combined. However, we have not attempted to estimate the likelihood that failure assumptions of our model hold for the boiler system. While it might be easy to estimate that sensors report values within their specified accuracy when they have not failed, it would probably be difficult to estimate that failed sensors lead to inconsistencies, as a detailed knowledge of the likelihood and behaviour of failure modes is needed. Furthermore, the system context is relevant. For example, a sensor that fails to a 0 reading will produce a consistent failure in contexts where a 0 reading is expected.

Acknowledgements

We would like to thank Peter Bishop of Adelard for his helpful comments. This work was supported by SERC project GR/J 58619 “Communication in Safety Cases – A Semantic Approach”.

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

References

[1] A. Hopkins, T. Smith, and J. Lala, “FTMP – a highly reliable fault-tolerant multiprocessor for aircraft,” *Proceedings of the IEEE*, pp. 1221–1239, October 1978.

[2] J. Wensley, L. Lamport, J. Goldberg, M. Green, K. Levitt, P. Melliar-Smith, R. Shostak, and C. Weinstock, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proceedings of the IEEE*, vol. 60, pp. 1240–1255, October 1978.

[3] C. Goring, "Safety first – applying computer-based safety systems in the off-shore environment," *Computing and Control Engineering Journal*, pp. 245–250, December 1993.

[4] J. von Neumann, "Probabilistic logics and synthesis of reliable organisms from unreliable components," *Annals of Mathematical Studies*, vol. 34, pp. 43–98, 1956.

[5] L. Lamport, "The temporal logic of actions," Tech. Rep. 79, Digital Systems Research Center, 1991. To be published in ACM Transactions on Programming Languages and Systems.

[6] "Generic problem competition." Institute for Risk Research, University of Waterloo, 1992.

[7] "Specification for a software program for a boiler water content monitor and control system." Institute for Risk Research, University of Waterloo, 1992. A component of the International Symposium on the Design and Review of Software Controlled Safety-Related Systems.

[8] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[9] E. Y. Chow and A. S. Willsky, "Analytical redundancy and the design of robust failure detection systems," *IEEE Transactions on Automatic Control*, vol. AC-29, pp. 603–614, July 1984.

[10] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems*, vol. 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[11] B. Randall, "System structure for software fault tolerance," *IEEE Transactions on Software Engineering*, vol. SE-1, pp. 220–232, June 1975.

[12] T. Anderson and P. Lee, eds., *Fault Tolerance: Principles and Practice*. Prentice Hall, 1981.

[13] F. Cristian, "A rigorous approach to fault-tolerant programming," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 23–31, January 1985.

[14] F. B. Schneider, "Byzantine generals in action: Implementing fail-stop processors," Tech. Rep. 83-569, Department of Computer Science, Cornell University, 1983.