Validating Safety Models with Fault Trees

Glenn Bruns and Stuart Anderson

Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, UK

Abstract. In verifying a safety-critical system, one usually begins by building a model of the basic system and of its safety mechanisms. If the basic system model does not reflect reality, the verification results are misleading. We show how a model of a system can be compared with the system's fault trees to help validate the failure behaviour of the model. To do this, the meaning of fault trees are formalised in temporal logic and a consistency relation between models and fault trees is defined. An important practical feature of the technique is that it allows models and fault trees to be compared even if some events in the fault tree are not found in the system model.

1 Introduction

Safety-critical systems often have mechanisms designed to prevent, detect, or tolerate system faults. To ensure that these mechanisms work as intended, a model of the system can be built from two parts: a model of the basic system and a model of the safety mechanisms (see Figure 1). Important properties of the system are then verified of the model. For example, if a component failure occurs, then it is detected.

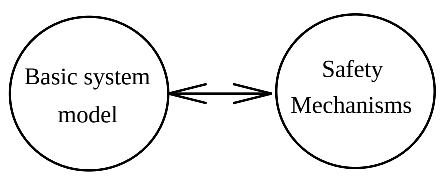

Fig. 1. A Model of a Safety-Critical System

For the verification results to be valid, the basic part of the model should reflect the true connection between component failures and system faults in the system. We are aware of a study of a rail interlocking system in which the preliminary system model allowed only one train per track section, thus making collisions impossible. Less obvious problems may be harder to discover, such as when a particular combination of failures leads to a system fault in the real system but not the system model.

We propose a validation technique in which a system model is compared to its fault trees. If a system model and its fault trees are not consistent in a sense that we will define, then the system model may not be valid. Fault trees are well suited for this purpose because they are specifically intended to capture the relationship between component failure and system faults.

The two main sections of the paper cover the precise meaning of fault trees and our proposed relationship between fault trees and system models. First, however, we present an example.

2 Example

To make discussion of the problem more concrete, we present a simple boiler system example (see Figure 2).

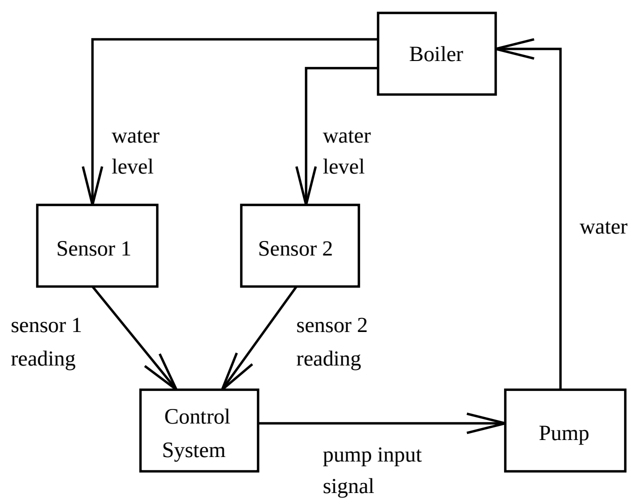

Fig. 2. A Simple Boiler System

Steam is produced by water contained in the boiler vessel. The water level in the vessel is read by two sensors, which pass their readings to a control system. If the readings are below a certain value, the pump is turned on, delivering water to the vessel. If the level readings are above a certain value, the pump is turned off.

One safety-critical fault of the system is a boiler level that is too high. A fault tree for this fault is given in Figure 3.

A fault tree represents how events in a system can lead to a particular system fault. The event symbols used here are either *basic* events (which are drawn as circles and represent component failures) or *intermediate* events (which are drawn as rectangles and represent events which occur because of lower-level

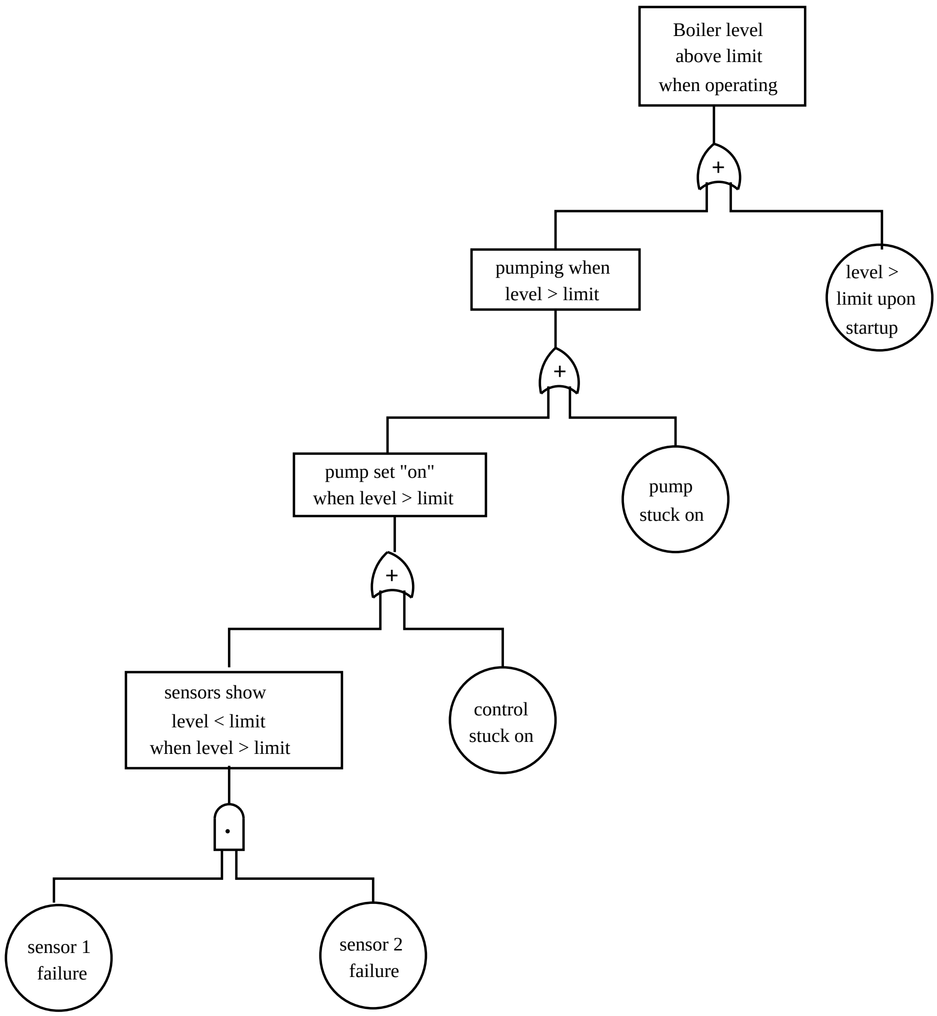

Fig. 3. A Fault Tree for the Boiler System

events). The system fault is shown as the event at the root of the tree. Event symbols are connected in the tree by gate symbols, which are either *and*-gates or *or*-gates.

The full fault tree notation has many more event and gate symbols, but if we do not consider the probabilistic meaning of fault trees then the symbols we have described are enough.

3 Fault Tree Semantics

If we are to compare fault trees and system models, we need to understand precisely what a fault tree means. Unfortunately, even the most definitive sources (e.g., the Fault Tree Handbook [5]) are vague on some critical points.

One issue is the *nature of events*. Are they to be regarded as conditions having duration or as instantaneous occurrences? The example event "contacts fail to open" from the Fault Tree Handbook suggests the former, but the example "timer reset" suggests the latter.

The second issue is the *gate condition*: does "and" mean that both input events happen at once, or only that one happens and then the other?

A third issue is the *nature of causality*. A gate models a *sufficient* cause if the output must occur if the gate condition is satisfied by the inputs. A gate models a *necessary* cause if the gate condition must be satisfied by the inputs if the output occurs. According to the Fault Tree Handbook, fault trees model sufficient and necessary causes. However, Figure IX-10 of the Handbook shows an event labelled "wire faults in K3 relay & comp. circuitry" as a cause of "K3 relay contacts fail to close", but one can imagine circumstances in which wire faults occur in such a way that the relay contacts do not fail to close. Therefore the cause as stated is not a sufficient one.

Causes of an event are also supposed to be *immediate*. This term seems related to the notion of flow, and may not be relevant in systems that cannot be captured easily with flow models. All examples in the Fault Tree Handbook are illustrated with flow diagrams. Immediacy also suggests time. For our purposes, a gate models an immediate cause if no time passes between a cause and its effect.

We now present a formal semantics for fault trees. Events are treated as conditions having duration, and the gate condition is taken to be that both inputs to an and-gate must occur at once. Three different formalisations of gates are given, corresponding to different stances on the issue of gate causality.

Formally, fault trees are interpreted as formulas of temporal logic. We use the modal mu-calculus (see Appendix A), but nearly all temporal logics are expressive enough for our purposes. Similarly, the kinds of structures that temporal logics are interpreted over are very general. We assume only that a system model can be represented as a transition system or as a set of sequences of states.

Events are formalised as atomic propositions, which are interpreted as sets of states. For example, the event "sensor failure" could be modelled as the atomic proposition SF, which is interpreted as all states in which the sensor has failed. This formalisation of events fits with most of the examples of the Fault Tree Handbook, and is consistent with the meaning of the term "event" in probability theory. Since fault tree are subject to probabilistic analysis, a consistent view of events is desirable.

Next we will formalise the meaning of gates. We will let $+(in_1, in_2, out)$ stand for an or-gate with inputs in_1 and in_2 and output out. Similarly, $\bullet(in_1, in_2, out)$ stands for an and-gate. The semantics of a gate g, denoted $[\![g]\!]$, gives the logical relationship between the input and output events of g.

3.1 A Propositional Semantics for Gates

Formalising gates with propositional logic is a simple approach that is reasonably close to the informal description of gates in the Fault Tree Handbook. In terms of the issues just discussed, this interpretation requires and-gate inputs to occur at the same time for the gate condition to be satisfied, and takes causality to be necessary, sufficient, and immediate. The subscript p on the semantic function stands for "propositional".

$$[\![+(in_1, in_2, out)]\!]_p \stackrel{\text{def}}{=} out \Leftrightarrow in_1 \vee in_2$$

$$[\![\bullet(in_1, in_2, out)]\!]_p \stackrel{\text{def}}{=} out \Leftrightarrow in_1 \wedge in_2$$

Informally, the first statement says that the output of an and-gate is true whenever both inputs are true. Remembering that events are treated as sets of states, the statement alternatively says that the set of states denoted by *out* is the intersection of the sets denoted by in_1 and in_2. The concept of causality here is truly immediate: whenever both causes are present the effect is also present.

3.2 Two Temporal Semantics for Gates

The greatest weakness of the propositional interpretation of fault trees is the assumption that no time can pass between cause and effect. This assumption violates a common intuition about causality. Since the examples in the Fault Tree Handbook mostly concern examples in which flow is virtually instantaneous (as in an electric circuit), the problem rarely arises there. In cases where flow is not instantaneous, events are modelled so that causes can be made immediate, albeit somewhat unnaturally. For example, in the pressure tank analysis of Chapter VII continuous pump operation can lead to a pump failure. This cause is modelled as the event "tank ruptures due to internal over-pressure caused by continuous pump operation for t > 60 sec". Since the idea of a cause leading to an event is natural, it is worthwhile to try to view fault trees in this way.

Our first temporal semantics requires that and-gate inputs occur at the same time to satisfy the gate condition, and takes causality to be only sufficient, not necessary or immediate. This means that once the gate condition is satisfied, the gate output must eventually occur. The temporal logic operator **even** is used to express the temporal condition of eventuality. Thus $\mathbf{even}(\phi)$ means that the property expressed by formula ϕ will hold in the future.

The temporal relation between input and output events for gates can be defined as

$$[\![+(in_1, in_2, out)]\!]_{t1} \stackrel{\text{def}}{=} (in_1 \vee in_2) \Rightarrow \mathbf{even}(out)$$

$$[\![\bullet(in_1, in_2, out)]\!]_{t1} \stackrel{\text{def}}{=} (in_1 \wedge in_2) \Rightarrow \mathbf{even}(out)$$

The first definition says that it is always the case that if input events in_1 and in_2 occur together, then eventually output event *out* will occur.

Our second temporal semantics treats causality as only necessary. The temporal operator $\mathbf{prev}(\phi)$ means that the property expressed by formula ϕ held in the past.

$$[\![+(in_1, in_2, out)]\!]_{t2} \stackrel{\text{def}}{=} out \Rightarrow \mathbf{prev}(in_1 \vee in_2)$$
$$[\![\bullet(in_1, in_2, out)]\!]_{t2} \stackrel{\text{def}}{=} out \Rightarrow \mathbf{prev}(in_1 \wedge in_2)$$

However, these definitions allows the gate output *out* to occur many times for a single occurrence of $in_1 \wedge in_2$. A better interpretation might require that if *out* happens, then $in_1 \wedge in_2$ must have happened at least as recently as the previous occurrence of *out*.

There are other possible interpretations based on other choices about the basic semantic issues. For example, combining the two temporal semantics we have presented would give one modelling sufficient and necessary causality.

Fault tree gates have been interpreted temporally before (see [1]), but the use of temporal logic here allows much simpler semantics. This simplicity makes comparison between alternative interpretations easier.

3.3 Putting Gates Together

We now present the semantics of a fault tree t based on the set of gates contained in the tree (written as $gates(t)$) . We use the temporal operator $\mathbf{always}(\phi)$, which means that the property expressed by ϕ holds in every state.

$$[\![t]\!] \stackrel{\text{def}}{=} \mathbf{always}(\bigwedge_{g \in gates(t)} [\![g]\!])$$

In English, this definition says that it is always the case that every gate condition is satisfied. Note that the meaning of a fault tree is given in terms of the meaning of its gates.

The propositional semantics has some great advantages over the temporal ones. Because a gate output is defined in the propositional case to be logically equivalent to the disjunction or conjunction of its inputs, the fault tree can be manipulated according to the laws of propositional logic. This property allows internal events of a fault tree to be removed by simplification, giving a relation between only the primary failures and the system fault (as is found in minimal cut set interpretations of fault trees [5]).

A further advantage of the propositional interpretation of fault trees is that the meaning is given as an *invariant* property – a property that can be checked by looking at states in isolation. Invariant properties are an easy class of temporal logic formulas to prove.

The main advantage of the temporal semantics is their ability to model richer notions of causality. Unfortunately, it is no longer possible to eliminate internal events by simplification, and thus minimal cut sets cannot generally be obtained. Furthermore, this formalisation of fault trees uses the temporal property of eventuality, and is therefore a *liveness* property. This class of temporal logic formulas are generally more difficult to prove than invariant formulas.

The best interpretation of a fault tree probably depends on the system being studied. In some cases one might want to choose different interpretations for different kinds of gates. For example, or-gates could be interpreted propositionally and and-gates interpreted temporally. Alternatively, a wider variety of gate types could be defined, and their use mixed in a single fault tree.

The material in the next section can be applied independently of choice of semantics for fault trees.

4 Relating Fault Trees to System Models

The last section showed that a fault tree expresses a property of failure events in a system. We might therefore expect a model of the system to have the properties expressed by its fault trees. We will attempt to make this relationship precise.

Let $\mathcal{F}$ stand for the set of system faults for which fault trees have been developed, and let $ft(F)$ be the fault trees for fault F. Given a system model $\mathcal{M}$ with an initial state s_0, we write $s_0 \models_{\mathcal{M}} \phi$ if the system model has the property expressed by formula ϕ. The condition expressing that a model $\mathcal{M}$ of a system is consistent with the set of fault trees for the system is

$$s_0 \models_{\mathcal{M}} \bigwedge_{F \in \mathcal{F}} [\![ft(F)]\!]$$

This condition is too strong, however, because usually a system model will capture only certain aspects of a system. One way to weaken the relation above is to require a system model to satisfy the property expressed by a fault tree only if the system fault of the tree is found in the system model. Letting $faults(\mathcal{M})$ stand for the system faults in a model $\mathcal{M}$, the new consistency condition is

$$s_0 \models_{\mathcal{M}} \bigwedge_{F \in \mathcal{F} \cap faults(\mathcal{M})} [\![ft(F)]\!]$$

This relation is still quite strong, however. If a system model only captures certain failures, then it probably would not satisfy this condition. It would be useful to know the weakest relation that should definitely be expected to hold between a model of a system and the fault trees of a system. Our approach is to assume that we know nothing about events not given in a system model. As an example, suppose that we have a single or-gate, $+(B, C, A)$, which by the propositional interpretation gives the relation $A \Leftrightarrow B \vee C$ between events A, B, and C. Also suppose that we know nothing about event B. Then we will still expect that $C \Rightarrow A$. Logically this amounts to the projection of the relation $A \Leftrightarrow B \vee C$ onto the atomic propositions A and C. The projected relation is arrived at by taking the disjunction of the cases where B is true and B is false. In other words, the disjunction of $A \Leftrightarrow true \vee C$ and $A \Leftrightarrow false \vee C$ is equivalent to the formula $C \Rightarrow A$. In the general case, where more than one event might be missing, we need to consider all combinations of possibilities for the missing events.

To formalise this idea, let $\phi[\phi'/Q]$ be the formula ϕ with every occurrence of atomic proposition Q within ϕ replaced by ϕ'. For example, $A \vee B[false/B]$ gives $A \vee false$. For multiple substitutions, let $\phi[\phi_1/Q_1, \ldots, \phi_n/Q_n]$ be the formula ϕ with occurrences of $Q_1, \ldots, Q_n$ in ϕ simultaneously replaced by $\phi_1, \ldots, \phi_n$ (no atomic proposition is allowed to occur twice in the substitution list). For example, $A \vee B \Leftrightarrow C[true/A, false/C]$ gives $true \vee B \Leftrightarrow false$. We will write $S_1 \times S_2$ for the cross product of sets S_1 and S_2, i.e., $S_1 \times S_2 \stackrel{\text{def}}{=} \{(x, y) \mid x \in S_1 \text{ and } y \in S_2\}$.

Let $Bool$ be the set $\{true, false\}$ of boolean constants, and let $Bool^n$ be the n-fold product of $Bool$. The interpretation of a fault tree in the absence of a set of events $\mathcal{E} = \{a_1, \ldots, a_n\}$ is defined to be:

$$[\![t - \mathcal{E}]\!] \stackrel{\text{def}}{=} \bigvee_{\{b_1,\ldots,b_n\} \in Bool^n} [\![t]\!][b_1/a_1, \ldots, b_n/a_n]$$

Let $events(\mathcal{M})$ be the set of events in the system model $\mathcal{M}$, and let $events(t)$ be the set of events in fault tree t. Then $events(t) \setminus events(\mathcal{M})$ is the set of events found in the fault tree t but not the model $\mathcal{M}$. The condition expressing that a model $\mathcal{M}$ of a system is consistent with the set of fault trees for the system is now

$$s_0 \models_{\mathcal{M}} \bigwedge_{F \in \mathcal{F} \cap faults(\mathcal{M})} [\![ft(F) - (events(ft(F)) \setminus events(\mathcal{M}))]\!]$$

5 Conclusions

This paper contains three contributions to the study of safety-critical systems. First, it presents the idea that fault trees can be used to check the validity of safety-critical system models. Second, it contains three formal semantics for fault trees. These semantics are an improvement on earlier work by expressing the meaning of fault trees with temporal logic, by expressing events as sets of states, and by identifying four elements of the meaning of gates: gate condition, sufficiency, necessity, and immediacy. Finally, the paper defines a consistency condition between a model of a system and the system's fault trees that works even for models that contain only some of the failure events in the their fault trees.

Tool support for checking the consistency condition exists in the form of *model checkers*, which automatically show whether a finite-state model satisfies a temporal logic formula [3]. Proof tools (such as [2]) are available in case the model is not finite-state.

The work described here should be regarded as a first step towards a complete understanding of fault trees and their relation to system models. As mentioned in the section on the semantics of gates, the formalisation here of necessary causes may be too simplistic. The consistency condition given might need to be strengthened to ensure that an event representing a component failure can always occur provided it has not already occurred. Our consistency condition also cannot handle the case in which a single failure event in a system model represents several failure events in a fault tree.

A A Temporal Logic

We use an extended form of the modal mu-calculus [4, 6] as a temporal logic to express behavioural properties. The syntax of the extended mu-calculus is as follows, where L ranges over sets of actions, Q ranges over atomic sentences, and Z ranges over variables:

$$\phi ::= Q \;\big|\; \neg\phi \;\big|\; \phi_1 \wedge \phi_2 \;\big|\; [L]\phi \;\big|\; Z \;\big|\; \nu Z.\phi$$

The operator νZ binds free occurrences of Z in ϕ, with the syntactic restriction that free occurrences of Z in a formula ϕ lie within an even number of negations.

Let $\mathcal{S}$ be a set of states and *Act* a set of actions. A formula ϕ is interpreted as the set $\|\phi\|^{\mathcal{T}}_{\mathcal{V}}$ of states, defined relative to a a fixed transition system $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathit{Act}\})$ and a valuation $\mathcal{V}$, which maps variables to sets of states. The notation $\mathcal{V}[\mathcal{S}'/Z]$ stands for the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ except that $\mathcal{V}'(Z) = \mathcal{S}'$. Since the transition system is fixed we usually drop the state set and write simply $\|\phi\|_{\mathcal{V}}$. The definition of $\|\phi\|_{\mathcal{V}}$ is as follows:

$$\begin{aligned}
\|Q\|_{\mathcal{V}} &= \mathcal{V}(Q)\\
\|\neg\phi\|_{\mathcal{V}} &= \mathcal{S} - \|\phi\|_{\mathcal{V}}\\
\|\phi_1 \wedge \phi_2\|_{\mathcal{V}} &= \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}}\\
\|[L]\phi\|_{\mathcal{V}} &= \{s \in \mathcal{S} \mid \text{if } s \xrightarrow{a} s' \text{ and } a \in L \text{ then } s' \in \|\phi\|_{\mathcal{V}}\}\\
\|Z\|_{\mathcal{V}} &= \mathcal{V}(Z)\\
\|\nu Z.\phi\|_{\mathcal{V}} &= \bigcup\{\mathcal{S}' \subseteq \mathcal{S} \mid \mathcal{S}' \subseteq \|\phi\|_{\mathcal{V}[\mathcal{S}'/Z]}\}
\end{aligned}$$

A state s satisfies a formula relative to a model $\mathcal{M} = (\mathcal{T}, \mathcal{V})$, written $s \models_{\mathcal{M}} \phi$, if $s \in \|\phi\|^{\mathcal{T}}_{\mathcal{V}}$.

Informally, $[L]\phi$ holds of a state s if ϕ holds for all states s' that can be reached from s through an action a in L. A fixed point formula can be understood by keeping in mind that $\nu Z.\phi$ can be replaced by its "unfolding": the formula ϕ with Z replaced by $\nu Z.\phi$ itself. Thus, $\nu Z.\psi \wedge [\{a\}]Z = \psi \wedge [\{a\}](\nu Z.\psi \wedge [\{a\}]Z) = \psi \wedge [\{a\}](\psi \wedge [\{a\}](\nu Z.\psi \wedge [\{a\}]Z)) = \ldots$ holds of any process for which ψ holds along any execution path of a actions.

The operators $\vee$, $\langle a\rangle$, and μZ are defined as duals to existing operators (where $\phi[\psi/Z]$ is the property obtained by substituting ψ for free occurrences of Z in ϕ):

$$\begin{aligned}
\phi_1 \vee \phi_2 &\stackrel{\text{def}}{=} \neg(\neg\phi_1 \wedge \neg\phi_2)\\
\langle L\rangle\phi &\stackrel{\text{def}}{=} \neg[L]\neg\phi\\
\mu Z.\phi &\stackrel{\text{def}}{=} \neg\nu Z.\neg\phi[\neg Z/Z]
\end{aligned}$$

These additional basic abbreviations are also convenient:

$$[a_1, \ldots, a_n]\phi \stackrel{\text{def}}{=} [\{a_1, \ldots, a_n\}]\phi$$
$$[-]\phi \stackrel{\text{def}}{=} [Act]\phi$$
$$true \stackrel{\text{def}}{=} \nu Z.Z$$
$$false \stackrel{\text{def}}{=} \neg true$$

Common operators of temporal logic can also be defined as abbreviations:

$$\textbf{always}(\phi) \stackrel{\text{def}}{=} \nu Z.\phi \wedge [-]Z$$
$$\textbf{even}(\phi) \stackrel{\text{def}}{=} \mu Z.\phi \vee (\langle - \rangle true \wedge [-]Z)$$

To define a *previously* operator, a reverse modal operator $\overline{[L]}$ must be added to the logic.

$$\|\overline{[L]}\phi\|_{\mathcal{V}} = \{s \in \mathcal{S} \mid \text{if } s' \xrightarrow{a} s \text{ and } a \in L \text{ then } s' \in \|\phi\|_{\mathcal{V}}\}$$

The previously operator is just the reverse version of **even**:

$$\textbf{prev}(\phi) \stackrel{\text{def}}{=} \mu Z.\phi \vee (\overline{\langle - \rangle} true \wedge \overline{[-]}Z)$$

References

1. R.E. Bloomfield, J.H. Cheng, and J. Gorski. Towards a common safety description model. In J.F. Lindeberg, editor, *SAFECOMP '91*, 1991.
2. J.C. Bradfield. A proof assistand for symbolic model checking. In *Proceedings of CAV '92*, 1992.
3. Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The concurrency workbench: A semantics based tool for the verification of concurrent systems. Technical Report ECS-LFCS-89-83, Laboratory for Foundations of Computer Science, University of Edinburgh, 1989.
4. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
5. N.H. Roberts, W.E. Vesely, D.F. Haasl, and F.F. Goldberg. *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission, 1981.
6. C. Stirling. Temporal logics for CCS. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models*. Springer Verlag, 1989. Lecture Notes in Computer Science, 354.

This article was processed using the LaTeX macro package with LLNCS style