

COMPUTING SCIENCE

Validating Formal Verification using Safety Analysis Techniques

R. de Lemos, A. Saeed

TECHNICAL REPORT SERIES

No. 668
March, 1999

Contact:

r.delemos@newcastle.ac.uk, a.saeed@newcastle.ac.uk

<http://www.cs.ncl.ac.uk/people/r.delemos>

<http://www.cs.ncl.ac.uk/people/a.saeed>

Copyright © 1999 University of Newcastle upon Tyne
Published by the University of Newcastle upon Tyne,
Department of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK

Validating Formal Verification using Safety Analysis Techniques

Rogério de Lemos and Amer Saeed

Department of Computing Science
University of Newcastle upon Tyne, NE1 7RU, UK
{r.delemos, a.saeed}@newcastle.ac.uk

Abstract. The increased interest in the use of automated safety analysis is supported by the claim that manual safety analysis based on traditional techniques is error-prone, costly and not necessarily complete. It is also claimed that traditional techniques are not able to deal with the inherent complexities of software intensive systems. However, we show in this paper that a transition (from manual to automatic approaches) in the assessment process and technologies is accompanied by an inherent risk of obtaining false confidence, unless safeguards are provided. The safeguard presented in this paper integrates traditional deductive and inductive analysis techniques with model checking, a form of formal verification. The aim is to provide the safety analyst with a rigorous approach for the validation of formal models. The feasibility of the overall approach is illustrated in terms of a case study.

Keywords: model checking, formal verification, traditional safety analysis techniques, deductive and inductive analysis, formal refinement, requirements.

1. Introduction

The current trend has been to claim that manual traditional safety analysis techniques are error-prone, costly and not necessarily complete, moreover, it is also claimed that they are not able to deal with the inherent complexities of software intensive systems. In this paper, we claim that it is not sufficient to replace traditional manual techniques for automated counterparts which seem more effective in dealing with highly complex systems, unless the principles and the rationale behind the traditional safety analysis techniques are preserved. Moreover, the transition from manual to automatic in the practice of conducting safety analysis either in the technologies or in the process of assessment is vulnerable to the inherent risk of obtaining a false confidence.

Over the years traditional safety analysis techniques have provided an effective means by which analysts are able to generate, in a rigorous manner, failure scenarios that might affect safety, thus obtaining a deep understanding of the intricacies that affect the safety of complex systems. One merit of traditional safety analysis techniques is the methodological rigour that forces the analyst to focus on issues related with system safety. On the other hand, automatic safety analysis approaches, that aim to replace traditional ones, lack the methodological rigour which forces the safety analysts to have a comprehensive understanding of system safety (in terms of faults and their possible consequences). Furthermore, the formal models which provide the basis for the automatic approaches (the “raw material” of the automated tools) are often impenetrable to checks on their accuracy, hence are usually considered as “black boxes” by the safety analyst. In these circumstances, emphasis changes from completeness of the analysis to accuracy of the modelling, leading to the separation of two activities which traditionally have been combined for safety analysis.

Automated tools might be effective in dealing with some of the inherent complexities of software based systems, for example dependencies of many failure modes of system components. However if support is not provided to enable the analyst to understand the rationale behind key safety-related design choices, then the effectiveness of these tools is limited. In our opinion, to reduce the risk of obtaining false confidence, the application of automated tools should be supported by a deductive and inductive analysis of the formal models subject to automated analysis. In this paper we present an approach which uses safety analysis techniques for validating formal verification, based on model checking. The domain of application is safety-critical control systems, particularly those systems which are real-time and hybrid in their nature.

The paper is structured as follows. Section 2 provides an overview on model checking and the concept of abstractions, and describes the risks associated with model checking as a safety analysis technique. In Section 3 we describe how deductive and inductive analysis can be integrated with model checking in order to attain higher levels of confidence for the models being analysed. The case study which illustrates the

proposed approach is presented in Section 4. Finally, Section 5 concludes with a discussion evaluating our contribution and indicating directions for future work.

2. Model Checking

Model checking is a formal verification technique based on state exploration. Given a state transition system and a property, model checking algorithms exhaustively explore the state space to determine whether the system satisfies the property. The result is either a claim that the property is true (provision of *evidence*) or a counter-example in terms of a sequence of states that falsifies a property (guidance for *risk reduction*) /Chan 98/.

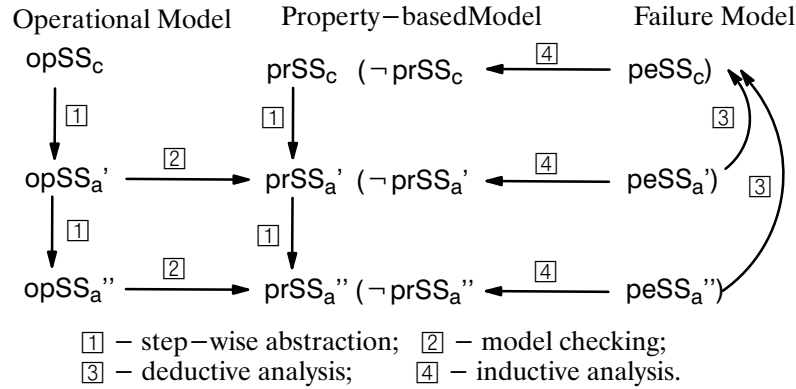


Figure 1. The basis for model checking.

The diagram of figure 1, schematically represents the basic steps of model checking. The property-based and operational representations of the concrete model of the system are: $prSS_c$ and $opSS_c$. Concrete models might include, for instance, non-linearities which the model checking algorithms are unable to analyse. Hence the need to translate into linear models, by applying abstractions, to obtain the property-based ($prSS_{a'}$) and operational ($opSS_{a'}$) descriptions of an abstract model.

Abstract interpretation is a technique for the systematic translation of a concrete model to an abstract one (using conservative approximations), and can be used either for improving the efficiency for analysing a discrete system by reducing the state explosion /Clarke 94, Heitmeyer 98/, or for approximating nonlinear hybrid systems /Henzinger 95a/. For both cases, existing approaches may rely on more than one method for obtaining efficient and precise representations, hence the multiple abstraction levels shown in figure 1. For example, for reducing the state space, *variable restriction* and *variable abstraction* can be used, respectively, to remove irrelevant variables and to replace detailed variables with more abstract ones /Heitmeyer 98/.

In model checking, a key relationship between an abstract model and the concrete model, from which it is derived, is preservation of properties (essentially invariants) across the models. Ideally an abstract model ($prSS_a$, $opSS_a$) should be both *sound* (i.e. if $prSS_a$ is a property of $opSS_a$, $prSS_c$ is also a property of $opSS_c$) and *complete* (i.e. if it is sound, and $prSS_c$ is a property of $opSS_c$, $prSS_a$ is also a property of $opSS_a$) for the concrete model ($prSS_c$, $opSS_c$) with respect to properties of interest. Methods have been proposed for abstraction that aim to preserve soundness and completeness.

One approach proposed for hybrid automaton /Henzinger 95a/ is to provide general algorithm's for the transformation of a concrete (nonlinear) model to an abstract (linear) equivalent. Two schemes are proposed. *Clock translation*, the resultant abstract model is both sound and complete, however the scheme can only be applied in the presence of strict conditions. *Phase-portrait approximation*, although this can be applied to any hybrid automaton the resultant model is sound but not complete. Another approach proposed for interpreted finite state machines /Heitmeyer 98/ is to provide methods for the derivation of an abstract model to support the analysis of a particular property, in the presence of certain conditions. This results in abstract models that are sound, but may not be complete. In summary, these methods enable an analyst to obtain confidence (directly from the process of translation) that the results obtained from model checking the abstract model are applicable to the concrete model.

2.2. Model Checking and Safety Analysis

A typical approach for the application of model checking to safety analysis is: to represent the safety property that the system has to satisfy, usually associated with the negation of the system hazard, as the the prop-

erty-based model, and the operational model would represent the system being designed, including the possible failures of the components of the system. For model checking to be effective as a safety analysis technique, it should support risk reduction and provide evidence for safety. In terms of risk reduction, model checking can identify the possible causes for the violation of the properties associated with the model; once causes are identified the model can be modified to eliminate or mitigate the risk (if both the property and model cannot be modified then risk remains unchanged). In terms of evidence, model checking can show that despite failures in the components of the system, the safety properties of the system are not affected (or if affected the risk associated with the failures is acceptable).

As with any modelling technique, the confidence that can be attributed to the results obtained from model checking is dependent on the accuracy of the models, hence property-based and operational models should be validated to confirm they are accurate representations of the actual system. Although it is relatively easy to check whether the operational model satisfies the specified properties, there are several error sources in the process of modelling. For example, either the property-based or operational models might have a misrepresentation (inappropriate parameter which defines states/transitions of the automata, or flawed initial conditions) that allow a property to be confirmed for the model despite it being inappropriate for the real system. In particular, an analogy can be made with testing when applying model checking as a safety analysis technique: model checking is able to confirm the presence of faults in the model, but not their absence. Moreover, while testing is able to probe the actual product being developed, model checking is only restricted to probe a representation of the actual product. Hence, additional assurance should be provided that either the model being checked is an accurate representation of the system, or that all the exposed inaccuracies between the model and the actual system do not impact system safety.

In terms of abstract interpretation, although the actual methods for either reducing the state explosion or for approximating nonlinear hybrid systems might be sound and complete, the process of applying these methods is still error prone. During the transformation of a concrete specification into a more abstract one certain details of the models are discarded, these could eliminate some of the causes for a hazardous state. Although the abstract model might be proven to be safe, there are cases in which it cannot be claimed that the concrete model is also safe – when the abstract model $(prSS_a, opSS_a)$ is not a sound abstraction of the concrete model $(prSS_c, opSS_c)$.

In this paper we do not claim that model checking should be avoided as a safety analysis technique. On the contrary, our claim is that model checking is an effective technique when used with caution, as investigated for the model checker SMV (Symbolic Model Verification /McMillan 92/) for conducting fault analysis of chemical plants. Over confidence on the capabilities of model checking without balancing its inherent risks might lead to some unexpected and undesired situations. An approach for the validation of abstract interpretations using safety analysis techniques is presented. Instead of validating both the property-based and the operational models, the focus is on the property-based model. Otherwise the complexity of the proposed approach would be proportional with the the number of states of the operational model.

3. Vulnerability Analysis for Abstraction

The aim of vulnerability analysis (VA), when applied to model checking, is to provide evidence to support the claim that safety properties of the abstract model are reflected in the concrete model, and to complement the results obtained from model checking. The mechanics for checking for potential vulnerabilities are based on the application of traditional deductive and inductive analysis of the property-based representations.

- *Deductive analysis*, in terms of safety analysis, starts with a hazard and proceeds backwards, seeking possible failures that can lead to the hazard. An example of a method which supports deductive analysis is Fault Tree Analysis (FTA).
- *Inductive analysis*, in terms of safety analysis, starts with a set of failure events and proceeds forward, seeking possible consequences (i.e. hazards) resulting from the events. Typical examples of methods which support inductive analysis are Failure Modes and Effect Analysis (FMEA) and Event Tree Analysis (ETA).

The approach being proposed is a safeguard against obtaining false confidence from model checking, or following inappropriate guidance for risk reduction in the system. Although the transformation methods can be sound/complete, it is possible that these methods are not applied in accordance with their guidelines. Moreover, there are situations in which existing methods for the abstract interpretation of concrete models

are not suitable for a particular application, the analyst is forced to employ ad-hoc approaches to derive an abstract model. In these cases, although it might not be possible to find an alternative transformation which is sound/complete, it might nevertheless be effective to conduct the model checking of a correspondent abstract model, to obtain evidence that the risk is acceptable.

To incorporate deductive and inductive analysis into model checking, the original dual language (property-based and operational) description of systems has to include a third description related with the set of primitive events that lead to the violation of safety properties. The model of the system is then represented as a triple $(opSS, prSS, peSS)$, which also includes the failure model ($peSS$) a predicate that characterises a set of primitive events (denoted by $peSS$) that refer to failures in the system components, or violations in the system behavioural assumptions.

The complete framework for model checking, including the representation of failure behaviours, is shown in figure 1. From this framework we can infer the following logical relations. The first two relations establish the cause-consequence relationship between the occurrence of primitive events and the violation of a safety property. The occurrence of a primitive event in the concrete model $peSS_c$ (abstract model $peSS_a$), is a sufficient condition for the violation of the safety property $prSS_c$ (resp. $prSS_a$):

$$peSS_c \Rightarrow \neg prSS_c, \quad peSS_a \Rightarrow \neg prSS_a$$

The following relation is a term of a refinement relation which also includes safety assumptions and unsafe states /de Lemos 98/, and states that the violation of an abstract safety property implies the violation of the correspondent concrete safety property:

$$\neg prSS_a \Rightarrow \neg prSS_c$$

During the deductive analysis the aim is to identify and compare the set of primitive events of the concrete and abstract models which can lead to the violation of the safety property. This enables a comparison between the failure properties of two different models of the same system /Cepin 97/, and can lead to several possibilities. In terms of safety, the most relevant cases are those in which model checking of the abstract model leads to inconclusive results: when there is a subset of excluded primitive events ($peSS_c^* = peSS_c - peSS_a$) which are part of the concrete model, but not part of the abstract model. In these cases, model checking of the abstract model is not able to identify counter-examples which demonstrate that the safety property can be violated on the real system. Once these excluded primitive events are identified, the role of the inductive analysis is to establish the sequence of events that can lead to a system hazard. The outcome of this analysis will provide the basis to mitigate or eliminate the risk associated with a primitive event. This provides a complementing means for conducting safety analysis when model checking the concrete model is not possible. In terms of the diagram of figure 1, the deductive analysis is conducted by comparing the primitive events of a concrete model with its correspondent abstract interpretation, and the inductive analysis is conducted by identifying the sequence of events that lead to the violation of a safety property, once an excluded primitive event occurs.

The integration of safety analysis techniques with the abstract interpretation of concrete models is based on similar work which integrated deductive and inductive analysis with the process of decomposing and refining safety specifications /de Lemos 98/. For refinement, the main concern is to check what is the impact in terms of system safety when detailing a specification: new hazards can be introduced, and new assumptions have to be identified. Although the process of abstraction is quite different from the refinement of specifications (in the sense that detail is actually removed from the specifications rather being added), the safety analysis techniques which have shown to be effective for refinement can also be applied with minor modifications to abstraction.

3.1. Deductive Analysis

During the deductive analysis the aim is to confirm that the failure properties of the abstract model are consistent with those of the concrete model. The process to achieve this consists of two stages:

- *a causal analysis* of the concrete (and abstract models) to determine the causes for the violation of a safety property $prSS_c$ (respectively $prSS_a$) in the context of its operational specification $opSS_c$ ($opSS_a$);

- a comparison between the causes for the abstract and concrete models.

The starting point for the causal analysis is to identify the hazard characterized by the negation of the safety property (i.e. $\neg \text{prSS}_c$). The next step is to identify sources of failures which define the causes, known as primitive events, that lead to the hazard. In this paper, FTA is advocated for the deductive analysis, since it constrains the analysis by identifying only those behaviours that lead to the violation of specification prSS_c .

The comparison of peSS_c and peSS_a can result in five cases. Two extremes are: *identical* sets (i.e. $\text{peSS}_c = \text{peSS}_a$), the results of model checking are applicable to risk reduction and provision of evidence; *disjoint* sets (i.e. $\text{peSS}_a \cap \text{peSS}_c = \{\}$), nothing can be inferred about the safety properties of the concrete model. The other three (more typical) cases are discussed in detail.

- *Event inclusion* – the primitive events of the concrete model are a subset of those obtained from the abstract model (i.e. $\text{peSS}_c \subset \text{peSS}_a$). Evidence is obtained from model checking, however guidance for risk reduction (i.e. counter-examples) for the abstract model may not be applicable to the concrete model.
- *Event exclusion* – the primitive events of the concrete model are a superset of those obtained from the abstract model (i.e. $\text{peSS}_c \supset \text{peSS}_a$). Guidance provided for risk reduction by model checking the abstract model can be used to modify the concrete model, however evidence that a safety property holds for the abstract model may not be applicable to confirmation for the concrete model. Two strategies can be adopted, to overcome the deficiencies in model checking for this case. In a first strategy, the impact of the excluded primitive events peSS_c^* on system safety can be determined by inductive analysis. An alternative strategy is to define an additional property prSS_a^* , for which the corresponding peSS_a^* will include peSS_c^* . In this case confirmation (by model checking) of both prSS_a and prSS_a^* can be used to confirm prSS_c .
- *Event overlap* – the intersection of the primitive events is a strict subset of the events of the concrete and abstract models (i.e. $\text{pe} = \text{peSS}_c \cap \text{peSS}_a$, $\text{pe} \subset \text{peSS}_c$ and $\text{pe} \subset \text{peSS}_a$). In this case both the evidence and guidance for risk-reduction, obtained from model checking, may not be applicable for the concrete model. The two strategies proposed in the event exclusion case can also be applied here. For event overlap these strategies enable the results to be treated as for event inclusion (within an identified level of risk), or identify other properties over the abstract model to mimic event inclusion.

3.2. Inductive Analysis

Starting from the occurrence of a primitive event the role of inductive analysis of safety specifications is to determine the sequence of events which can lead to the violation of a safety property. The aim of the inductive analysis is to complement the safety analysis conducted using model checking by probing the primitive events of the concrete model which are not part of the set of primitive events of the abstract model. This analysis is necessary for the cases of event exclusion and event overlap, to analyse the impact of excluded primitive events (peSS_c^*). The identification of the sequence of events that leads to the violation of a safety property establishes potential weakness that the concrete model might have. These weakness need to be rectified to mitigate or eliminate the risk associated with a particular primitive event, unless the risk associated with the weakness is acceptable. In this paper, the use of ETA is advocated for conducting the inductive analysis.

4. Case Study: Industrial Press

4.1. Description of the Case Study

The case study involves establishing design parameters for the safe operation of an industrial press /de Lemos 96, Holcombe 96/. The press comprises a plunger, an operator and a programmable logic controller (PLC). At its maximum height the plunger is held by a latch. Once the latch is removed the plunger moves downwards. Its upward motion is controlled by a motor. The operator controls the movement of the plunger via two buttons placed one meter apart. The states of the buttons are interpreted by the PLC which controls the latch and motor. A diagram of the industrial press, and the forces associated with the plunger are shown in figure 2.

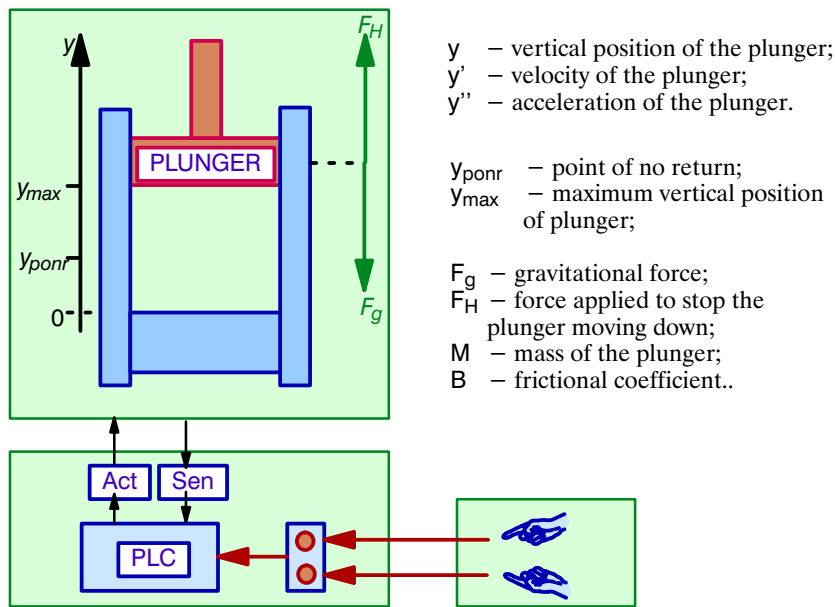


Figure 2. Diagram of the industrial press ensemble.

The informal requirements for the design of the press aim to ensure its safe operation:

- if the operator releases either button while the press is closing, the plunger should stop before it closes.

As the plunger is a very heavy object, closing the press is dangerous. Indeed, it should be prevented from closing fully if the operator can be injured. However, there will be positions in the plunger downward motion at which the motor will not be able to prevent the plunger from closing fully. At these positions, any danger will be prolonged if we attempt to prevent the press from closing as such an action can only slow the plunger without stopping it. The point of no return provides the basis for a control strategy which distinguishes two basic modes of operation associated with the press:

- the press closes without interruption until it is closed;
- if the press is closing and the operator releases either one of the two buttons while the plunger has not yet passed the point of no return, then closing is interrupted.

In the following, we will focus on the strategy of stopping the plunger when the operator releases one of the buttons. In this scenario, a force is applied to the plunger which hinders the press to close. The system is intrinsically nonlinear, described by the concrete model. Whether the press will close depends on the parameters which describe the press, and the height of the plunger when an external force is applied. However, in the following we show that model checking can produce different outcomes depending on the ad hoc abstractions which are employed over the concrete model.

4.2. Property-Based Model of the Industrial Press

The concrete model of the industrial press was originally specified using Extended Real-Time Logic /de Lemos 96, Hall 96/ (an outline of ERTL is presented in Appendix A). The abstract model was specified using Hybrid Automata and the model checker HYTECH /Henzinger 95b/ was used to check the safety properties of the abstract model (although HYTECH does not support the implementation of ICTL model checking, many real-time requirements can be reduced to reachability analysis).

In the following subsections, for the sake of brevity, we focus on the property-based representations of the industrial press. The correspondent operational models of both the concrete and abstract representations are presented in Appendix B. To represent the concrete model in a concise form, we have employed the notation of Hybrid Automata, as a graphical representation for the ERTL model /de Lemos 96/. The system is defined in terms of four state machines: **plunger** – to model the states of the plunger; **operator** – models the position of the operator in relation to the plunger; **opInterface** – models the state of the two buttons; and **controller** – a specification for the PLC. Common event labels are used to represent synchronisation between the component state machines.

In the concrete model, the states of the **plunger** are defined in terms of position (y), velocity (y') and acceleration (y''). The nonlinearities of the concrete model do not permit model checking. Hence abstract interpretations that eliminate the constraints over acceleration, need to be applied to obtain a linear model. Unfortunately methods based on *variable restriction* cannot be directly applied to y'' , without adjusting the value of other parameters, since the property of interest is dependent on y'' . The method of *clock translation* is not applicable and though *phase-portrait approximation* can be applied, its application for this system is not straightforward (several options were studied). The **plunger** machine for the abstract model, presented in figure B.4, was obtained by eliminating constraints on y'' and where deemed appropriate adjusting the constraints on y' to take into account the affects of y'' .

4.2.1. ERTL Model of the Industrial Press

The safety property is defined in the context of the ERTL operational model, in particular the states of the **plunger** and **operator** machines. The initial step for defining the safety property is to identify the system hazard **S_Hazard** which occurs whenever the operator enters a hazard state **O_Hazard** (i.e. places his hands in the target area of the plunger) while the plunger also enters a hazard state **P_Closed_Hazard**.

$$\forall t \bullet \Phi(\text{S_Hazard}, t) \Leftrightarrow \Phi(\text{P_Closed_Hazard}, t) \wedge \Phi(\text{O_Hazard}, t).$$

The plunger can only enter into a hazard state when the press closes while attempting to stop the plunger (**P_Stopping**). The following relation states that for the plunger to be in a hazard state, there exists a previous time point at which the plunger was stopping.

$$\forall t \bullet \Phi(\text{P_Closed_Hazard}, t) \Rightarrow \exists t_1 < t : \Phi(\text{P_Stopping}, t_1).$$

This relation can be rewritten by replacing the system predicate **P_Stopping** with the variables that characterize the state.

$$\forall t \bullet \Phi(\text{P_Closed_Hazard}, t) \Rightarrow \exists t_1 < t : \Phi(0 < y < y_{\max} \wedge -v_c < y' < 0 \wedge y'' = (F_H - F_g - B y')/M, t_1).$$

4.2.2. HYTECH Model of the Industrial Press

In HYTECH the safety property previously can be reduced into a reachability analysis problem. Two regions are defined an *initial region* (a composite of initial conditions for the state machines), and a *final region* (typically the negation of the property to be confirmed). HYTECH then aims to compute a path from the initial region to the final region, if one exists a counter-example is provided otherwise the property is confirmed.

For the abstract model of the industrial press, the initial region is defined as:

$$\begin{aligned} \text{init_reg} := & \text{loc[plunger]} = \text{P_Opened} \ \& \ y = 20 \ \& \\ & \text{loc[operator]} = \text{O_Init} \ \& \ \text{loc[opInterface]} = \text{OI_NoB} \ \& \\ & \text{loc[controller]} = \text{C_Opened_NoBOneB}. \end{aligned}$$

The final region is simply the definition of **S_HAZARD**:

$$\text{final_reg} := \text{loc[plunger]} = \text{P_Closed_Hazard} \ \& \ \text{loc[operator]} = \text{O_Hazard}.$$

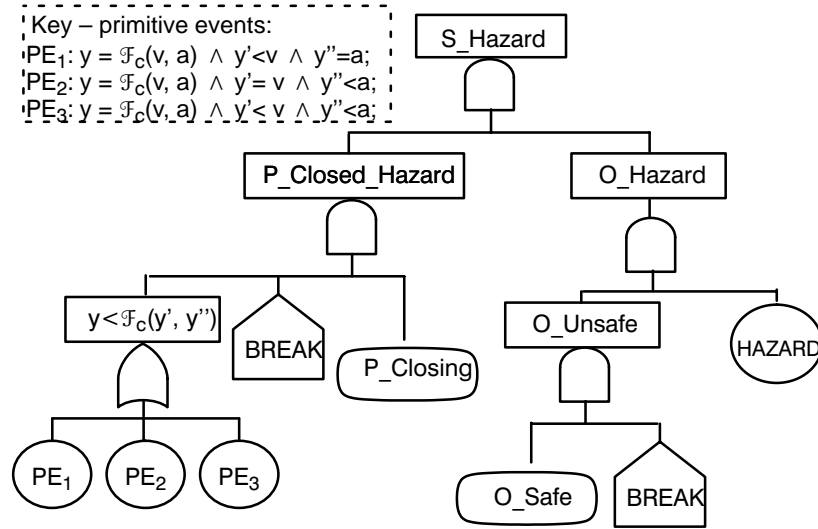
If the reachability analysis confirms that **final_reg** is not reachable, we can conclude the safety property is an invariant for the abstract (HYTECH) model.

4.3. Vulnerability Analysis

In the following we analyse the impact on safety of the elimination of acceleration from the dynamic model of the plunger. Intuitively, we can summarise the risk posed by this scenario as follows. If the acceleration is eliminated from the abstract model then the plunger in the concrete model may take longer to close from the same specified position. Hence, even if the reachability analysis confirms for the abstract model that the plunger will close before the operator can enter the hazard state, the same cannot be claimed for the concrete model unless further analysis is conducted. This scenario corresponds to *event exclusion* mentioned previously.

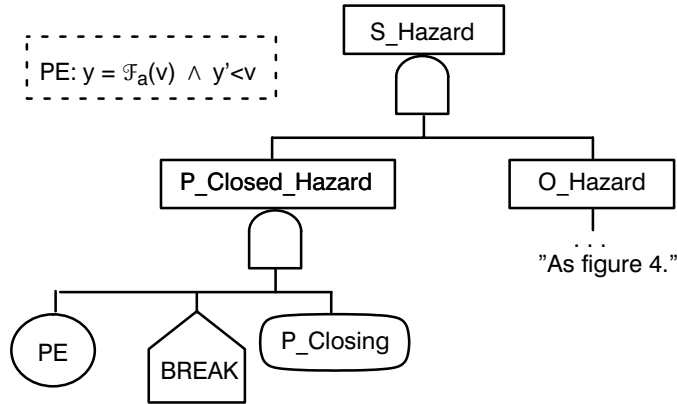
4.3.1. Deductive Analysis of the Abstract and Concrete Models

Step 1: Causal Analysis of Concrete model. The causes for the top event S_Hazard are the plunger being in P_Closed_Hazard and the operator being in O_Hazard. The fault tree summarising the causal analysis is shown in figure 3, in the following we focus on the causes for P_Closed_Hazard. This is caused by the plunger being in state P_Closed a BREAK occurring (this event synchronises with the operator) and the position of y being too low for the given velocity and acceleration. This analysis leads to three primitive events for the plunger: PE1 – downward velocity too high, PE2– upward acceleration too low and PE3 – both velocity and acceleration.



$\mathcal{F}_c(v, a)$ is a monotonic decreasing function that returns the lowest position at which a BREAK can occur and at which the plunger will stop for given v and a ;
Figure 3. Fault tree for concrete model.

Step 2: Causal Analysis of Abstract model. The fault tree for the abstract model (figure 4) differs in the causes for P_Closed_Hazard, now these are: state P_Closing, occurrence of BREAK and the downward velocity is too high (PE).



$\mathcal{F}_a(v)$ is a monotonic decreasing function that returns the lowest position at which a BREAK can occur and the plunger will stop for given v ;

Figure 4. Extract of fault tree for abstract model.

Step 3: Comparative Analysis of Causes. The predicate that characterises the primitive event sets for the concrete model $peSS_c = PE1 \wedge PE2, \wedge PE3$ and for the abstract model $peSS_a = PE$. The predicate for the excluded events is:

$$peSS_c^* \equiv y = \mathcal{F}_a(v) \wedge y = \mathcal{F}_c(v, a) \wedge y' = v \wedge y'' < a$$

The set $peSS_c^*$ corresponds to the situations in which a hazard occurs due to the acceleration being too low for the given velocity.

4.3.2. Inductive Analysis of the Concrete Models

The inductive analysis starts with the an event in the set **peSSc*** (represented as: $y' = v \wedge y'' < a$). To analyse the impact of this event on **S_HAZARD**, the state of the plunger (i.e if it is **P_Closing**), the occurrence of **BREAK** and the state of the operator (i.e. if it is **O_Hazard**) are analysed by the event tree in figure 5.

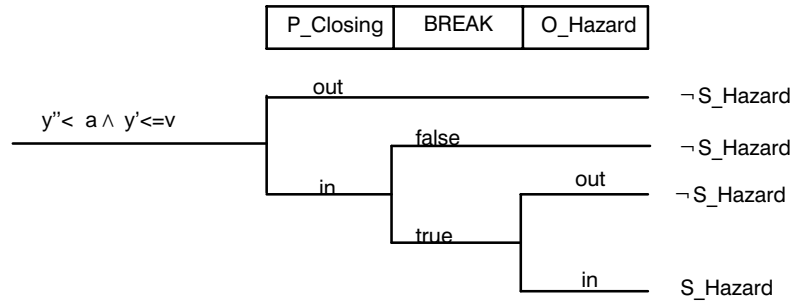


Figure 5. Event tree for the concrete model.

From the event tree, we can infer that the excluded events are hazardous if the plunger is in **P_Closing** and the event **BREAK** occurs and the operator is in **O_Hazard**. The risk posed by the excluded events, would be mitigated if **BREAK** never occurs during **P_Closing**. However, since these are represented as expected events and states (figure 3) this has a minimal affect on risk reduction. A more substantial mitigating factor, is the state **O_Hazard**. The event **BREAK** acts as a synchronisatation point that marks the start of a race between the plunger entering **P_Closed_Hazard** (assuming it does not succeed to stop) and the operator entering **O_Hazard**. For those circumstances in which the plunger enters **P_Closed_Hazard** before the operator enters **O_Hazard**, the risk posed by the excluded events is mitigated.

4.4. Remarks

The above VA shows that selecting a particular value for the velocity of the plunger in the state **P_Stopping** is problematic. One alternative is define a range e.g. $[-3/2, 0]$, however this leads to an inherently unsafe abstract model. Since a hazard can occur regardless of the delay assumed in the operator moving from **O_Unsafe** to **O_Hazard**.

In addition to providing the raw results the VA increases the understanding of the circumstances that lead to **S_Hazard**, these circumstances can viewed as the result of two races started by **BREAK**:

1. In the plunger the race is between closing (i.e y reaching 0) and stopping (i.e. y' reaching 0), closing wins if $y < \mathcal{F}_c(y', y'')$.
2. Between the plunger and the operator the race is between closing and the operator entering **O_Hazard**, the operator can win if the *minimum time* to enter **O_Hazard** from **O_Unsafe** is less than the *maximum time* to enter **P_Closed_Hazard**.

The value of the acceleration affects the result of race 1, if elimination of acceleration decreases the *maximum time* the plunger takes to close the (safe) *minimum time* for the operator to enter **O_Hazard** would be computed incorrectly. Viewing the causes in the time domain, suggests that a translation of the concrete model into a model based on representing the maximum time between a **BREAK** and entering **P_Closed_Hazard** could lead to a sound abstract model for **S_HAZARD**. Such an approach can be viewed as a combination of *clock translation* and *directed variable restriction*.

5. Conclusions

Although the model checking literature provides techniques which are sound and complete for obtaining abstract interpretations from concrete models, nevertheless the application of these techniques are still error prone (when not applied appropriately). Means to validate these abstract interpretations are necessary in order to obtain the required confidence that the abstract models obtained are indeed sufficiently accurate. In this paper we have presented an approach based on deductive and inductive analysis for the validation of the abstract models. We recognize the fact that the application of these techniques in complex systems is limited, however if the usage of these techniques is restricted to the property-based representations then we are able to scope their complexity. Furthermore, the traditional safety analysis techniques could have the same role as the dependency graph browser of the Software Cost Reduction (SCR) method, in automatically removing irrelevant variables from the model.

Another aspect that might hinder the utilization of model checking as a safety analysis technique is the lack of a methodological support for guiding its use when conducting safety analysis. Testing, for example, is a validation technique which relies on a wide range of strategies, such as black box testing, and random testing, to ensure that the risk associated with a particular system is acceptable. Once similar methodological support is defined, confidence can be placed on model checking as a safety analysis technique.

Acknowledgements

These authors would like to acknowledge the financial support of EPSRC/UK ADAPT and SafeGames projects.

References

- /Cepin 97/ M. Cepin, R. de Lemos, B. Mavko, S. Riddle, A. Saeed. "An Object-Based Approach to Modelling and Analysis of Failure Properties". *Proceedings of the 16th International Conference on Computer Safety, Reliability and Security (SAFECOMP'97)*. York, UK. September 1997. Ed. P. Daniel. Springer-Verlag. Berlin, Germany. pp. 281–294.
- /Clarke 94/ E. Clarke, O. Grumberg, and D. Long. "Model Checking and Abstraction". *ACM Transactions on Programming Languages and Systems (TOPLAS) Vol. 15(5)*. ACM. September, 1994.
- /Chan 98/ W. Chan, et. al. "Model Checking Large Software Specifications". *IEEE Transactions on Software Engineering Vol. 27(7)*. IEEE Computer Society. July 1998. pp. 498–520.
- /de Lemos 96/ R. de Lemos, and J. Hall. "Extended RTL in the Specification and Verification of an Industrial Press". *Hybrid Systems III*. LNCS 1066. Eds. R. Alur, T. A. Henzinger, and E. Sontag. Springer-Verlag. Berlin, Germany. 1996. pp. 114–125.
- /de Lemos 98/ R. de Lemos, A. Saeed, and T. Anderson. *On the Integration of Requirements Analysis and Safety Analysis for Safety-Critical Software*. Technical Report Series No. 630. Department of Computing Science, University of Newcastle upon Tyne, UK. May, 1998.
- /Hall 96/ J. G. Hall, and R. de Lemos. "ERTL: an Extension to RTL for the Specification, Analysis and Verification of Hybrid Systems". *Proceedings of the 8th EUROMICRO Workshop on Real-Time Systems 96*. L'Aquila, Italy. June 1996. pp. 3–8.
- /Heitmeyer 98/ C. Heitmeyer, et. al. "Using Abstraction and Model Checking to Detect Safety Violations in Requirements Specifications". *IEEE Transactions on Software Engineering Vol. 27(11)*. IEEE Computer Society. November 1998. pp. 927–948.
- /Henzinger 95a/ T. A. Henzinger, and P.-H. Ho. "Algorithmic Analysis of Non-Linear Hybrid Systems". *Proceedings of the Computer-Aided Verification (CAV'95)*. LNCS 939. Springer-Verlag. Berlin, Germany. pp. 225–238.
- /Henzinger 95b/ T. A. Henzinger, and P.-H. Ho. "HYTECH: The Cornell HYbrid TECHnology Tool". *Hybrid Systems II*. Lecture Notes in Computer Science 999. Eds. P. Antsaklis, et. al. Springer-Verlag, Germany. 1995. pp. 264–293.
- /Holcombe 96/ M. Holcombe, F. Ipate, and A. Groundoudis. "Complete Functional Testing of Safety Critical Systems". *Proceedings of the IFAC Workshop on Safety Reliability in Emerging Control Technologies*. November 1995. Daytona Beach, FL. Ed. T. Hilburn, G. Suski, and J. Zalewski. Pergamon Press. 1996. pp. 199–204.
- /Stauner 97/ T. Stauner, O. Müller, and M. Fuchs. "Using HYTECH to Verify an Automotive Control System". *Hybrid and Real-Time Systems*. Ed. O. Maler. LNCS 1201. Springer-Verlag. Berlin, Germany. 1997. pp. 139–153.
- /Turk 97/ A. L. Turk, S. T. Probst, and G. J. Powers. "Verification of Real Time Chemical Processing Systems". *Proceedings of the International Workshop on Hybrid and Real-Time Systems*. Lecture Notes in Computer Science 1201. Ed. O. Maler. Grenoble, France. March 1997. pp. 257–272.

Appendix A. Extended Real Time Logic (ERTL)

Extended Real Time Logic (ERTL) /de Lemos 96, Hall 96/ is a first order predicate logic for the modelling and analysis of hybrid systems, taking as a basis Jahanian & Mok's Real Time Logic (RTL) /Jahanian 86, Jahanian 88/. RTL uses uninterpreted predicates to relate events of a system to the time of their occurrence, thereby providing the means for reasoning about absolute timing properties of real-time systems. The ex-

tensions provided by ERTL allow reasoning about system behaviour in both value and time domains through predicates defined in terms of system variables.

The *occurrence relation* (Θ) captures the notion of real time by assigning a time value to each occurrence of an event. $\Theta(e, i, t)$ defines that the i th occurrence of event e occurs at time t .

$$\forall t \bullet \forall i \in P: \Theta(\text{MotorOn}, i, t)$$

The i th occurrence of event **MotorOn** has occurred at time t .

A *transition event* is defined by the transition of a system predicate from false to true, or from true to false, at a particular time point. For a system predicate P , the respective transition events are $\nearrow P$ and $\searrow P$.

$$\forall t \bullet \forall i \in P: \Theta(\searrow(\text{plateOnBeg} \wedge \text{beltOn}), i, t) \Leftrightarrow \Theta(\nearrow(\neg \text{plateOnEnd} \vee \neg \text{beltOn}), i, t)$$

The transition event which captures the instant which the conjunction of the predicates **plateOnBeg** and **beltOn** becomes false is equivalent to the transition event which captures the instant that the negation of either **plateOnBeg** or **beltOn** becomes true.

The *holding relation* (Φ) captures whether a system predicate holds true at a time point. $\Phi(f, i, t)$ defines that a formula f holds for the i th time, at time t .

$$\forall t \bullet \forall i \in P: \Phi(\text{moveDown}, i, t) \Leftrightarrow \Phi(\text{bottom} \wedge \text{plateOn}, i, t)$$

The predicate **moveDown** holds true iff the conjunction of the predicates **bottom** and **plateOn** also holds true.

Appendix B. Operational Model of the Industrial Press

B.1. ERTL Model of the Industrial Press

The concrete model of the industrial press was originally specified using Extended Real-Time Logic (ERTL) /de Lemos 96, Hall 96/. In the following, we have also employed the notation of Hybrid Automata in order to obtain a graphical representation for the ERTL model.

B.1.1. Model of the Plunger

Variables

The set of variables is $V = \{y, y', y''\}$:

Variables	Range	Comments	Units
y	R	The vertical position of the plunger, $\forall t \bullet \Phi(0 \leq y \leq y_{max}, t)$.	m
y'	R	The velocity of the plunger, $\forall t \bullet \Phi(v_{min} \leq y' \leq v_{max}, t)$.	m/s
y''	R	The acceleration of the plunger, $\forall t \bullet \Phi(a_{min} \leq y'' \leq a_{max}, t)$.	m/s^2

Control Locations

The set of control locations is $A = \{P_Opened, P_Closing, P_Closed_Safe, P_Closed_Hazard, P_Stopping, P_Stopped, P_Opening\}$:

P_Opened – the press is opened, and the plunger is up:

$$\forall t \bullet \Phi(P_Opened, t) \Leftrightarrow \Phi(y = y_{max} \wedge y' = 0 \wedge y'' = 0, t);$$

$P_Closing$ – the press is closing, and the plunger is moving down:

$$\forall t \bullet \Phi(P_Closing, t) \Leftrightarrow \Phi(0 < y < y_{max} \wedge -v_c < y' < 0 \wedge y'' = (-F_g - B y')/M, t);$$

P_Closed_Safe – the plunger is down, and the press has closed safely:

$$\forall t \bullet \Phi(P_Closed_Safe, t) \Leftrightarrow \Phi(y=0 \wedge y'=0 \wedge y''=0, t) \wedge \\ \forall t' \bullet \Theta(\nearrow P_Closed_Safe, t') \Rightarrow \Theta(\searrow P_Closing_TwoB, t');$$

P_Closed_Hazard – the press has closed, while the plunger was stopping:

$$\forall t \bullet \Phi(P_Closed_Hazard, t) \Leftrightarrow \Phi(y=0 \wedge y'=0 \wedge y''=0, t) \wedge \\ \forall t' \bullet \Theta(\nearrow P_Closed_Hazard, t') \Rightarrow \\ (\Theta(\searrow P_Stopping, t') \vee \Theta(\searrow P_Closing_One/NoB, t'));$$

$P_Stopping$ – the plunger is stopping:

$$\forall t \bullet \Phi(P_Stopping, t) \Leftrightarrow \Phi(0 < y < y_{max} \wedge -v_c < y' < 0 \wedge y'' = (F_H - F_g - B y')/M, t);$$

$P_Stopped$ – the plunger has stopped:

$$\forall t \bullet \Phi(P_Stopped, t) \Leftrightarrow \Phi(0 < y < y_{max} \wedge y' = 0 \wedge y'' = 0, t).$$

$P_Opening$ – the press is opening, and the plunger is moving up:

$$\forall t \bullet \Phi(P_Opening, t) \Leftrightarrow \Phi(0 < y < y_{max} \wedge y' = v_o \wedge y'' = 0, t).$$

Transition Events

The set of transition events is $T = \{ \nearrow P_Opened, \nearrow P_Closing, \nearrow P_Closed_Safe, \nearrow P_Closed_Hazard, \nearrow P_Stopping, \nearrow P_Stopped, \nearrow P_Opening \}$:

$$\forall t \bullet \Theta(\nearrow P_Opened, t) \Leftrightarrow \Theta(\searrow P_Opening, t);$$

$$\forall t \bullet \Theta(\nearrow P_Closing, t) \Leftrightarrow \Theta(\searrow P_Opened, t);$$

$$\forall t \bullet \Theta(\nearrow P_Closed_Safe, t) \Leftrightarrow \Theta(\searrow P_Closing, t);$$

$$\forall t \bullet \Theta(\nearrow P_Closed_Hazard, t) \Leftrightarrow \Theta(\searrow P_Stopping, t);$$

$$\forall t \bullet \Theta(\nearrow P_Stopping, t) \Leftrightarrow \Theta(\searrow P_Closing, t);$$

$$\forall t \bullet \Theta(\nearrow P_Stopped, t) \Leftrightarrow \Theta(\searrow P_Stopping, t);$$

$$\forall t \bullet \Theta(\nearrow P_Opening, t) \Leftrightarrow \Theta(\searrow P_Closed_Safe, t) \vee \\ \Theta(\searrow P_Closed_Hazard, t) \vee \Theta(\searrow P_Closed_Stopped, t);$$

Events

The set of events is $E = \{ CLOSE, BREAK, BOTTOM, STOPPED, OPEN, TOP \}$:

$CLOSE$ – the plunger starts to move down:

$$\forall t \bullet \Theta(CLOSE, t) \Leftrightarrow \Theta(\nearrow P_Closing, t);$$

$BREAK$ – the plunger starts to stop:

$$\forall t \bullet \Theta(BREAK, t) \Leftrightarrow \Theta(\nearrow P_Stopping, t);$$

$BOTTOM$ – the plunger reaches the bottom of the press:

$$\forall t \bullet \Theta(BOTTOM, t) \Leftrightarrow \Theta(\nearrow P_Closed_Safe, t) \vee \Theta(\nearrow P_Closed_Hazard, t);$$

$STOPPED$ – the plunger stops:

$$\forall t \bullet \Theta(STOPPED, t) \Leftrightarrow \Theta(\nearrow P_Stopped, t);$$

$OPEN$ – the plunger starts to move up:

$$\forall t \bullet \Theta(OPEN, t) \Leftrightarrow \Theta(\nearrow P_Opening, t);$$

TOP – the plunger reaches the top of the press:

$$\forall t \bullet \Theta(TOP, t) \Leftrightarrow \Theta(\nearrow P_Opened, t);$$

Initial Condition

As initial condition we assume that the plunger is the press is opened, and the plunger is up:

$$\Phi(P_Opened, t).$$

Hybrid Automata

The model consists of seven states (Figure B.1): **P_Opened**, plunger is fully opened; **P_Closing**, plunger is closing; **P_Stopping**, plunger is stopping; **P_Closed_Safe**, plunger closes safely; **P_Closed_Hazard**, a BOTTOM has occurred after **BREAK** and plunger closes in a hazardous situation; **P_Stopped**, plunger has stopped before closing; and **P_Opening** plunger is open.

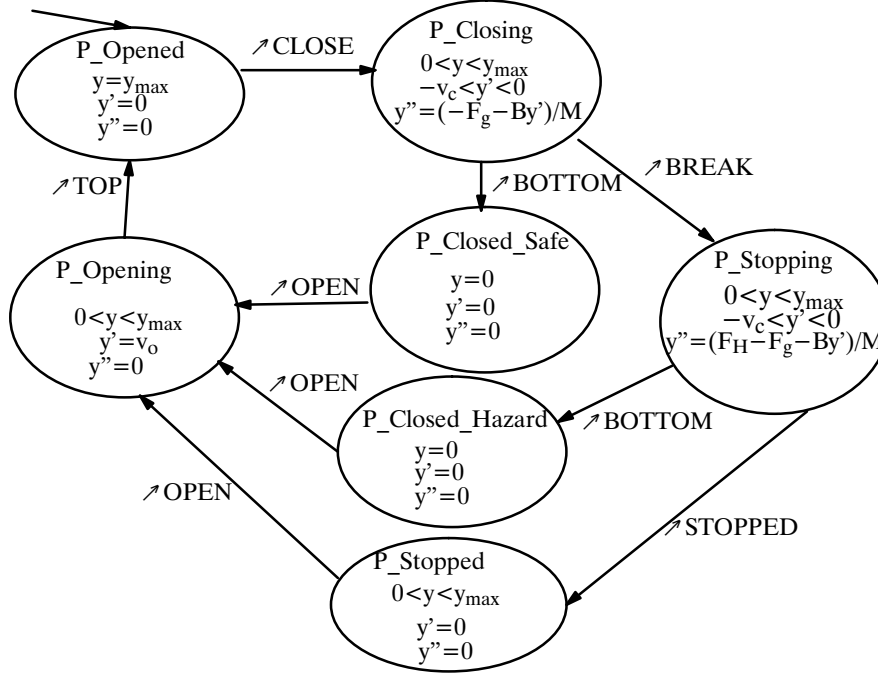


Figure B.1. Abstract Model of the plunger

B.1.2. Model of the Operator

Control Locations

The set of control locations is $\mathcal{A} = \{O_Init, O_Safe, O_Unsafe, O_Hazard\}$:

O_Init – the press is open;

O_Safe – the operator presses the two buttons;

O_Unsafe – the press is closing, the operator releases a button, and the hands are in the unsafe region;

O_Hazard – the press is either closing or closed, one of the buttons is released, and the hands are in the hazard region;

Transition Events

The set of transition events is $\mathcal{T} = \{\nearrow O_Init, \nearrow O_Safe, \nearrow O_Unsafe, \nearrow O_Hazard\}$:

$$\forall t \bullet \Theta(\nearrow O_Init, t) \Leftrightarrow \Theta(\searrow O_Init, t);$$

$$\forall t \bullet \Theta(\nearrow O_Safe, t) \Leftrightarrow \Theta(\searrow O_Init, t);$$

$$\forall t \bullet \Theta(\nearrow O_Unsafe, t) \Leftrightarrow \Theta(\searrow O_Safe, t);$$

$$\forall t \bullet \Theta(\nearrow O_Hazard, t) \Leftrightarrow \Theta(\searrow O_Unsafe, t) \vee \Theta(\searrow O_Hazard, t);$$

Events

The set of events is $E = \{OPOB, OPTB, OROB, ORTB\}$:

$OPOB$ – the operator presses one button:

$$\forall t \bullet \Theta(OPOB, t) \Leftrightarrow \Theta(\nearrow O_Init, t);$$

$OPTB$ – the operator presses the two buttons:

$$\forall t \bullet \Theta(OPTB, t) \Leftrightarrow \Theta(\nearrow O_Safe, t);$$

$OROB$ – the operator releases one button:

$$\forall t \bullet \Theta(OROB, t) \Leftrightarrow \Theta(\nearrow O_Unsafe, t);$$

Initial Condition

As initial condition we assume that the plunger is the press is opened, and the plunger is up:

$$\Phi(O_Init, t).$$

Hybrid Automata

The model of the operator consists of four states (Figure B.2): O_Init , the initial state of the operator; O_Safe , this corresponds to the situation when the operator presses two buttons on the plunger; O_Unsafe this corresponds to the situation when one of the Buttons is released; and O_Hazard when the operators hands are in the target area of the plunger.

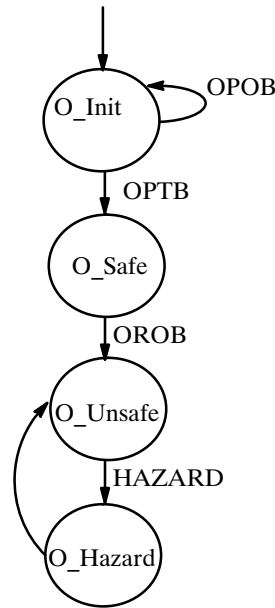


Figure B.2. Model of the operator

B.1.3. Model of the Operator Interface

Control Locations

The set of control locations is $\mathcal{A} = \{OI_NoB, OI_OneB, OI_TwoB\}$:

OI_NoB – no buttons are pressed;

OI_OneB – one of the buttons is pressed:

OI_TwoB – the two buttons are pressed.

Transition Events

The set of transition events is $T = \{ \nearrow OI_NoB, \nearrow OI_OneB, \nearrow OI_TwoB \}$:

$$\forall t \bullet \Theta(\nearrow OI_OneB, t) \Leftrightarrow \Theta(\searrow OI_NoB, t) \vee \Theta(\searrow OI_TwoB, t);$$

$$\forall t \bullet \Theta(\nearrow OI_TwoB, t) \Leftrightarrow \Theta(\searrow OI_OneB, t);$$

$$\forall t \bullet \Theta(\nearrow OI_NoB, t) \Leftrightarrow \Theta(\searrow OI_OneB, t);$$

Events

The set of event constants is $E = \{ OPOB, OPTB, OROB, ORTB \}$:

OPOB – operator presses one button;

$$\forall t \bullet \Theta(OPOB, t) \Leftrightarrow \Theta(\nearrow OI_OneB, t);$$

OPTB – operator presses two buttons;

$$\forall t \bullet \Theta(OPTB, t) \Leftrightarrow \Theta(\nearrow OI_TwoB, t);$$

OROB – operator releases one button;

$$\forall t \bullet \Theta(OROB, t) \Leftrightarrow \Theta(\nearrow OI_OneB, t);$$

ORTB – operator releases two buttons.

$$\forall t \bullet \Theta(ORTB, t) \Leftrightarrow \Theta(\nearrow OI_NoB, t);$$

Initial Condition

As initial condition we assume that the operator has not pressed any buttons:

$$\Phi(OI_NoB, 0).$$

Hybrid Automata

The model consists of three states (Figure B.3): *OI_NoB*, none of the buttons are pressed; *OI_OneB*, one of the buttons is pressed; and *OI_TwoB* both of the buttons are pressed. The events of this model synchronise with the operator.

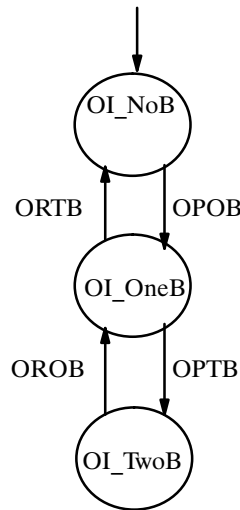


Figure B.3. Model of the operator interface

B.1.4. Model of the Controller

Control Locations

The set of control locations is $\mathcal{A} = \{ C_Opened_NoB/OneB, C_Opened_TwoB, C_Closing_TwoB, C_Closing_OneB/NoB, C_Closed_Safe, C_Closed_Hazard, C_Stopping, C_Stopped, C_Opening \}$:

$C_Opened_NoB/OneB$ – the press is opened, and the operator has not pressed the two buttons:

$$\forall t \bullet \Phi(C_Opened_NoB/OneB, t) \Leftrightarrow \Phi(P_Opened, t) \wedge (\Phi(OI_NoB, t) \vee \Phi(OI_OneB, t));$$

C_Opened_TwoB – the press is opened, and the operator is pressing the two buttons:

$$\forall t \bullet \Phi(C_Opened_TwoB, t) \Leftrightarrow \Phi(P_Opened, t) \wedge \Phi(OI_TwoB, t);$$

$C_Closing_TwoB$ – the plunger is moving down, and the operator is pressing the two buttons:

$$\forall t \bullet \Phi(C_Closing_TwoB, t) \Leftrightarrow \Phi(P_Closing, t) \wedge \Phi(OI_TwoB, t);$$

$C_Closing_NoB/OneB$ – the plunger is moving down, and the operator is either pressing only one button or none at all:

$$\forall t \bullet \Phi(C_Closing_OneB/NoB, t) \Leftrightarrow \Phi(P_Closing, t) \wedge (\Phi(OI_NoB, t) \vee \Phi(OI_OneB, t));$$

C_Closed_Safe – the plunger is down, and the press has closed safely:

$$\forall t \bullet \Phi(C_Closed_Safe, t) \Leftrightarrow \Phi(P_Closed_Safe, t);$$

C_Closed_Hazard – the press has closed while the plunger was stopping:

$$\forall t \bullet \Phi(C_Closed_Hazard, t) \Leftrightarrow \Phi(P_Closed_Hazard, t);$$

$C_Stopping$ – the plunger is stopping:

$$\forall t \bullet \Phi(C_Stopping, t) \Leftrightarrow \Phi(P_Stopping, t);$$

$C_Stopped$ – the plunger has stopped:

$$\forall t \bullet \Phi(C_Stopped, t) \Leftrightarrow \Phi(P_Stopped, t).$$

$C_Opening$ – the press is opening, and the plunger is moving up:

$$\forall t \bullet \Phi(C_Opening, t) \Leftrightarrow \Phi(P_Opening, t).$$

Transition Events

The set of transition events is $T = \{ \nearrow C_Opened_NoB/OneB, \nearrow C_Opened_TwoB, \nearrow C_Closing_TwoB, \nearrow C_Closing_OneB/NoB, \nearrow C_Closed_Safe, \nearrow C_Closed_Hazard, \nearrow C_Stopping, \nearrow C_Stopped, \nearrow C_Opening \}$:

$$\forall t \bullet \Theta(\nearrow C_Opened_NoB/OneB, t) \Leftrightarrow \Theta(\searrow C_Opening, t);$$

$$\forall t \bullet \Theta(\nearrow C_Opened_TwoB, t) \Leftrightarrow \Theta(\searrow C_Opened_NoB/OneB, t);$$

$$\forall t \bullet \Theta(\nearrow C_Closing_TwoB, t) \Leftrightarrow \Theta(\searrow C_Opened_TwoB, t);$$

$$\forall t \bullet \Theta(\nearrow C_Closing_OneB/NoB, t) \Leftrightarrow \Theta(\searrow C_Closing_TwoB, t);$$

$$\forall t \bullet \Theta(\nearrow C_Closed_Safe, t) \Leftrightarrow \Theta(\searrow C_Closing_TwoB, t);$$

$$\forall t \bullet \Theta(\nearrow C_Closed_Hazard, t) \Leftrightarrow \Theta(\searrow C_Closing_OneB/NoB, t) \vee \Theta(\searrow C_Stopping, t);$$

$$\forall t \bullet \Theta(\nearrow C_Stopping, t) \Leftrightarrow \Theta(\searrow C_Closing_OneB/NoB, t);$$

$$\forall t \bullet \Theta(\nearrow C_Stopped, t) \Leftrightarrow \Theta(\searrow C_Stopping, t);$$

$$\forall t \bullet \Theta(\nearrow C_Opening, t) \Leftrightarrow \Theta(\searrow C_Closed_Safe, t) \vee \Theta(\searrow C_Closed_Hazard, t) \vee \Theta(\searrow C_Closed_Stopped, t);$$

Events

The set of events is $E = \{CLOSE, BREAK, BOTTOM, STOPPED, OPEN, TOP, OPTB, OROB\}$:

$$\forall t \bullet \Theta(OPTB, t) \Leftrightarrow \Theta(\nearrow C_Opened_TwoB, t);$$

$$\forall t \bullet \Theta(OROB, t) \Leftrightarrow \Theta(\nearrow C_Closing_OneB/NoB, t);$$

$$\forall t \bullet \Theta(CLOSE, t) \Leftrightarrow \Theta(\nearrow C_ClosingTwoB, t);$$

$$\forall t \bullet \Theta(BREAK, t) \Leftrightarrow \Theta(\nearrow C_Stopping, t);$$

$$\forall t \bullet \Theta(BOTTOM, t) \Leftrightarrow \Theta(\nearrow C_Closed_Safe, t) \vee \Theta(\nearrow C_Closed_Hazard, t);$$

$$\forall t \bullet \Theta(STOPPED, t) \Leftrightarrow \Theta(\nearrow C_Stopped, t);$$

$$\forall t \bullet \Theta(OPEN, t) \Leftrightarrow \Theta(\nearrow C_Opening, t);$$

$$\forall t \bullet \Theta(TOP, t) \Leftrightarrow \Theta(\nearrow C_Opened_NoB/OneB, t);$$

Initial Condition

As initial condition we assume that the press is opened, and the operator has not pressed any buttons:

$$\Phi(C_Opened_NoB/OneB, 0).$$

Hybrid Automata

The model consists of nine states (Figure B.4) and can be viewed as a composite of the plunger model and operator interface model. The states are described as: **C_Opened_NoB/OneB**, corresponds to the situation when none or one button is pressed; **C_Opened_TwoB**, the situation when both buttons are pressed; **C_Closing_TwoB**, both buttons are pressed and the plunger is closing; **C_Closed_Safe**, the plunger closed safely; **C_Closed_Hazard**, the plunger closed in a hazardous situation; **C_Closing_OneB/NoB**, a button has been released while the plunger is closing; **C_Stopping**, the plunger is stopping; and **C_Opening**, the plunger is opening.

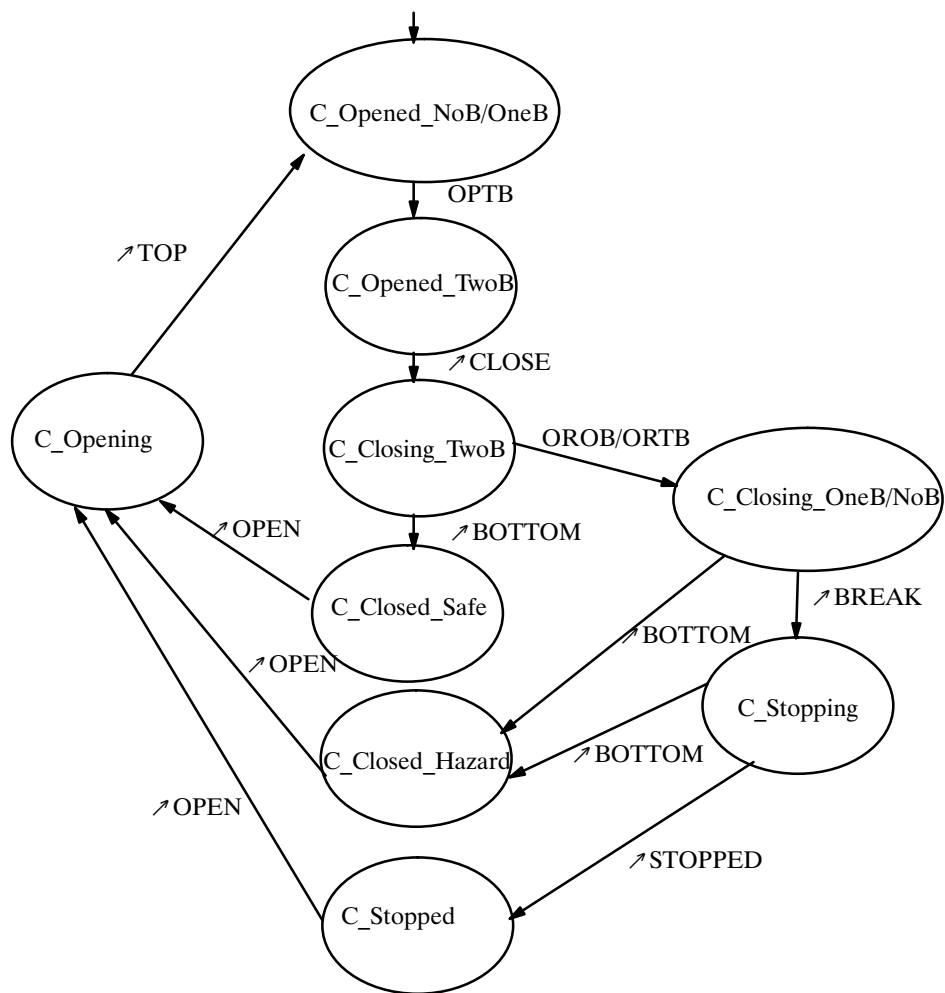


Figure B.4. Model of the controller

B.2. HYTECH Model of the Industrial Press

The HYTECH model also consists of four Hybrid Automata. The descriptions of the machines for the operator, operator interface and controller are not affected by the abstract interpretation that removes acceleration. Only the model of the plunger differs between the ERTL and HYTECH models.

B.2.1. Model of the Plunger

This model (figure B.5) consists of the same seven states as the ERTL model. The main differences are the elimination of the constraints over acceleration, and adjusting the constraint over velocity in $P_Stopping$ to be $-3/2$, in order to take into account acceleration. Also some of the strict inequalities over velocity are replaced by strict inequalities, and y_{max} is defined as a constant 20.

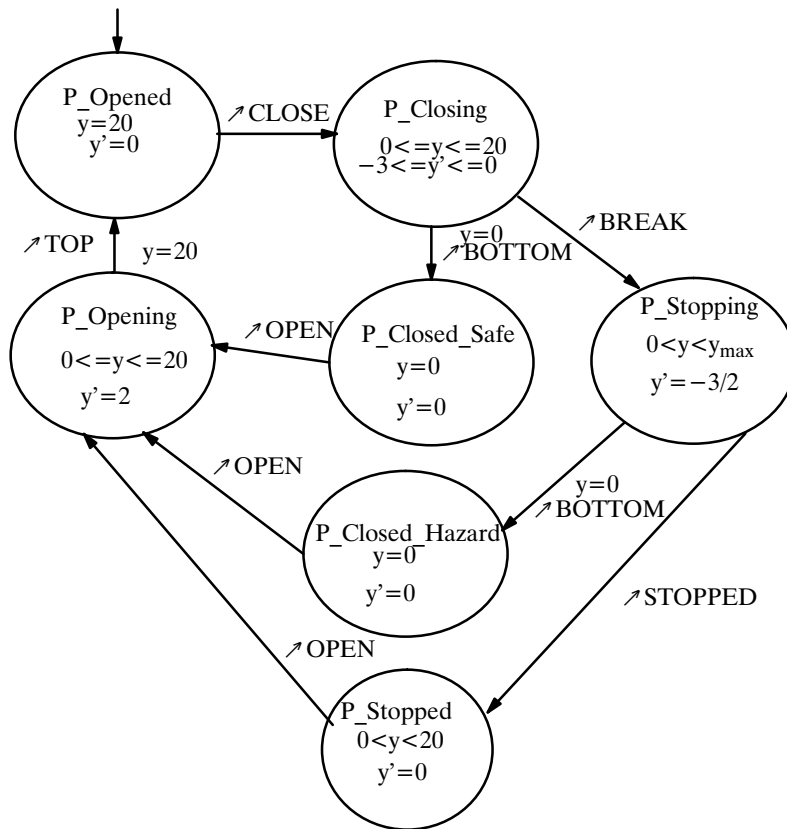


Figure B.5. Abstract Model of the plunger