# A Train Set as a Case Study for the Requirements Analysis of Safety-Critical Systems

R. DE LEMOS, A. SAEED AND T. ANDERSON
*Computing Laboratory, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU*

***Requirements analysis plays a vital role in the development of safety-critical systems since any faults in the requirements specification will corrupt the subsequent stages of system development. Experience in safety-critical systems has shown that faults in the requirements can and do cause disasters. The analysis of the safety requirements of a train set system is discussed in terms of a general framework for the requirements analysis of safety-critical systems. This framework is based on the clear separation of the mission and safety issues, and also on the separation of the analysis, into two phases, to be performed in terms of the properties of the real world, i.e. physical laws and rules of operation, and the properties of the system, i.e. the mapping of the real world properties in terms of the system sensors and actuators. Due to the different expressive needs of the two phases we propose the utilisation of different formal models, respectively, a logical formalism (Timed History Logic) and a net formalism (Predicate-Transition nets).***

*Received July 1991, revised September 1991*

## 1. INTRODUCTION

A safety-critical system is a system for which there exists at least one failure that can be adjudged to cause a disaster (e.g. loss of life). As the use of computers increases in critical applications and the level of criticality of the roles performed by the computers also increases, new methods for the development of such systems are urgently needed. One approach to improving the level of safety is to use formal specification and verification in conjunction with other methods of software development, such as testing and software fault-tolerance. The potential advantages of using formal methods, apart from contributing to improving the understanding of the whole specification, include unambiguity, refinement consistency, and the opportunity to check for completeness (with respect to a key set of questions and inferences based on the information specified).[5]

A further motivation for the work presented in this paper is the belief that a substantial improvement in the dependability of safety-critical systems can be achieved by performing a formal assessment on the results of the requirements analysis before proceeding to any subsequent phases. The aim of this approach is to locate and remove faults introduced during requirements analysis. (Although 'safety' is an attribute of the system rather than just software, in this paper the study is restricted to problems related to 'software safety'.)

The approach to be followed, in a framework for the software requirements analysis for safety-critical systems, is based on a clear separation of the mission and the safety requirements, and moreover, the subdivision of the analysis of the safety requirements into two distinct phases. On the one hand, the mission requirements focus on what the system is supposed to achieve in terms of function, timeliness and some dependability requirements – namely the attributes of reliability, availability and security. On the other hand, the safety requirements focus on the elimination and control of hazards, and the limitation of damage in the case of a disaster; in other words, they are related to the safety attribute of dependability.[8]

As far as the analysis of the safety requirements is concerned, we recognise two phases: the first deals with the identification of the real world properties, and the second deals with the mapping of these properties into the system.[1] Or alternatively, we are concerned with the laws or rules which dictate the behaviour of the system, and how these laws or rules are perceived and handled by the system. Because of the differing characteristics of each phase, instead of seeking a single formalism this paper discusses the approach of using a different formalism for each phase.[13] This has the advantage of allowing us to select formalisms in accordance with the properties that need to be expressed at each phase of development. The formal *specification* and *verification* of the safety requirements of a train set system presented in this paper aims to show that different formalisms can be integrated, exploiting the most appropriate features of each formalism.[15]

The rest of the paper is organised as follows. In the next section we present some justification for the separation of the mission from the safety requirements. Section 3 presents the framework for the formal specification and verification of the safety requirements, giving the characteristics and the formalism to be used in each phase. In Section 4, an example based on a train set crossing is discussed, in accordance with the framework of the previous section. Finally, Section 5 presents some concluding remarks.

## 2. SEPARATION BETWEEN THE MISSION AND SAFETY REQUIREMENTS

The adoption of a general structure for a safety-critical system will be a useful guide to the requirements analysis, since it allows the analysis to be split into different phases. For applications referred to as process control systems, which are the primary focus of this paper, a commonly accepted structure is to partition the system into three distinct components: the *operator*, the *controller* and the *physical process* (or plant). The *environment* is that part of the rest of the world which may affect, or be affected by, the system. This structure is further decomposed, to reflect the decision to separate

the mission and safety requirements, into mission and safety operators, and mission and safety controllers.

By analysing the progression of events that result in a disaster,[11] a justification for the separation between the mission and the safety requirements can be obtained. The *initiating event* is an event that can put the system in a hazardous state. A *hazard* is a physical situation, expressed as a system condition, that can lead to a disaster. A *disaster* is an unintended event or sequence of events that cause death, injury, environmental or material damage. The boundary between the safe and unsafe states of a system is related to the initiating events of the system. Thus, we define an *unsafe state* as a system state which follows some initiating event that can lead to a hazard, unless the safety controller and/or the safety operator takes corrective action. We apply the mission requirements to the behaviour of the system while it is in a safe state, and the safety requirements to the behaviour of the system when it enters an unsafe state.

There are some safety-critical systems for which the separation between mission and safety issues is realised at the level of the physical process, for example, the identification of shutdown systems of nuclear power plants. Other systems allow a separation to be made at the controller level, such as railway systems.[2] Whether it is feasible to separate the mission from the safety requirements for a particular safety-critical system depends on the ease with which a distinction can be drawn between safe and unsafe states. The mission controller is concerned with system behaviour while the system is in a safe state, and the safety controller when the system is in an unsafe state. Furthermore, the aims of the two controllers are distinct. The mission controller ensures that the mission is accomplished – this will also require that the system does not enter into an unsafe state. The safety controller is concerned with avoiding hazardous states by dealing with the unsafe states that precede these states; it is assumed that the system does not start in a hazardous state.

Some of the benefits of making this distinction during requirements analysis are: the resolution of potential conflicts, detection of omissions and inconsistencies between the mission and safety issues, the ability to focus on the safety-critical issues, and the simplification of safety certification.

## 3. ANALYSIS OF THE SAFETY REQUIREMENTS OF SAFETY-CRITICAL SYSTEMS

The basic concern in this paper is with the analysis of requirements and not with their elicitation (the process of acquisition of the relevant information from the user). We deal with techniques that can be used to reduce (or eliminate) the possibility of the occurrence of hazards due to faults introduced during the requirements analysis.

In the following, two phases of the analysis of the safety requirements are presented in terms of their main characteristics. These two phases are called the *Safety Requirements Analysis* and the *Safety System Analysis*. The separation of the analysis into these two distinct phases is intended to simplify the analysis task, permitting an easier understanding and reasoning about the real world properties, and how the system perceives and manipulates them.

Within the overall framework, we assume that there exists another phase – *Conceptual Analysis* – which should be conducted before the two mentioned above; this is intended to produce an initial informal statement of the aim and purpose of the system and to determine what is meant by safety for the system. As a product of this phase we obtain the Safety Requirements, enumerating the potential *disasters*, and the *hazards* related to these disasters.

### 3.1. Safety requirements analysis

In this phase, the real world properties, in terms of physical laws and rules of operation, are identified. As a product of this real world analysis, the *Safety Requirements Specification* is produced, containing the *safety constraints* and the *safety strategies*. A safety constraint is simply the negation of a hazard modified to incorporate safety margins, whereas a safety strategy is a means, defined as a set of conditions over the physical process, of maintaining the safety constraint. The general features which are required for a formalism to be appropriate for this phase are the following:[13] the specifications should have a conjunctive character, in the sense that new requirements can be added to the specification without the need to reconstruct the full specification, the behaviour of the system should be defined in terms of all of its possible runs (i.e. a linear sequence of events and/or states), and there is no need to explicitly specify concurrency and non-determinism. In the proposed framework each safety strategy is verified against the relevant safety constraint, and the strategies are checked for inconsistencies. The safety strategies are then validated against the mission requirements to ensure that these do not conflict while the system is in a safe state. The most appropriate formalisms for this phase are logical formalisms, such as Temporal Logic,[4] Real-Time Logic (RTL),[6] Real-Time Temporal Logic (RTTL),[12] and Timed History Logic (THL).[14]

In this paper we propose the use of THL during the Safety Requirements Analysis, primarily because the behaviour of systems expressed in THL is defined by imposing constraints over the set of all sequences of states that the system can exhibit. This property gives THL formulae a conjunctive nature. Therefore by using THL the Safety Requirements Specification can be constructed by considering each safety constraint and safety strategy separately.

### 3.2. Safety system analysis

The activities to be performed during this phase include the identification of the interface between the safety controller and the physical process, and the specification of system behaviour that must be observed at the identified interface. Also in this phase a top level organisation of the system is realised in terms of the properties of the sensors and actuators of the system, and the effects of possible failures of these sensors and actuators. This phase leads to the production of the *Safety System Specification*, containing the *safety controller strategies*. A safety controller strategy is a refinement of a safety strategy incorporating the sensors and actuators, and their relationship with the real world. Our aim is to construct a Safety System Specification

that should be unaffected by failures in the mission subsystem. The general features which are required for a formalism to be appropriate for this phase are the following: the specification should be able to represent the architectural design of the system, modelling the interactions between the components that implement the safety requirements, which implies the need to explicitly specify concurrency and non-determinism. In the proposed framework, each safety controller strategy is verified against the relevant safety strategy, and the controller strategies are checked for inconsistencies. The type of formalisms most appropriate for this phase are graphical formalisms[9] such as Petri nets, Extended State Machines (ESM),[12] and Modecharts.[6] Also, program-like formalisms that have constructs supporting non-determinism and concurrency can be employed, such as CSP and CCS.

In this paper we propose the use of Predicate-Transition nets (PrT nets),[3] a form of high-level Petri net, during the Safety System Analysis. Petri nets are mainly used for the modelling and analysis of discrete-event systems which are concurrent, asynchronous, and non-deterministic. The use of PrT nets, instead of (Timed) Petri nets,[9] adds to the modelling power of the latter the formal treatment of *individuals* (i.e. the notion of token identity) and their changing properties and relations.

### 3.3. The overall framework and comparison with other approaches

The basic aim behind our framework for the requirements analysis of safety-critical systems is to subdivide the whole problem into smaller domains where the analysis of the requirements can be simplified, thereby leading to more accurate specifications. This is achieved by, first, splitting the requirements into mission and safety requirements, and second, subdividing the analysis of the safety requirements into the Safety Requirements Analysis and Safety System Analysis. The analysis of the safety requirements in terms of the system may highlight some shortcomings in the safety strategy; these shortcomings have to be addressed by modifying the safety strategy and subsequently the safety controller strategy.

If we adopt the approach of performing the analysis of the safety requirements in two phases, and representing the specifications produced at each phase in a different formalism, we must provide some means to relate the different specifications. At the end of the first phase the safety requirements are expressed as a set of safety strategies, which are predicates within the THL model of the system. In the second phase, a PrT net model specifying a safety controller strategy is constructed for each safety strategy. To verify a safety controller strategy against its safety strategy, an essential step is to express the predicates that describe the safety strategy in terms of the predicates of the PrT net model. These predicates are the logical invariants of the net which will be used to verify that the PrT net model accurately captures the safety strategy.

As far as the combining of the specifications of the mission and safety requirements are concerned, once the safe and unsafe states of the system are identified, we need only verify that they are consistent. Hence, the notation used to express the mission and safety requirements specifications need not be the same, provided some means to check whether the specifications are consistent is available. Furthermore, this consistency check can be restricted to those mission requirements which affect the safety requirements.

The basic approach suggested in this paper, of dividing the requirements analysis into distinct phases, and performing the analysis at each phase in the most appropriate formalism, has not previously been investigated. What has usually been presented is the utilisation of a single formalism such as Invariants,[1] Temporal Logic,[4] Petri nets,[9] and THL.[14] However, there are two approaches in the literature which use different formalisms for requirements analysis.[6,12] These papers are primarily concerned with the analysis of timeliness requirements; it is not their concern to establish a framework for the analysis of the safety requirements of safety-critical systems – the issue which is central to this paper. In Ref. 12, ESMs are used to model 'plant-controller processes'; from the paths of these ESMs, trajectories are obtained which can be used to provide a formal operational semantics. The specification of plant behaviour is then given by RTTL formulae over these trajectories, and verified by demonstrating that the trajectories defined by the ESMs do indeed comply with RTTL formulae. In Ref. 6, the specification of the system is realised in terms of RTL and Modecharts; Modecharts produce a decision procedure for classes of properties expressed as RTL formulae. The system properties are verified using Computation Graphs, obtained from the Modecharts, to check if the corresponding RTL formulae comply with the Modecharts.

## 4. THE TRAIN SET CROSSING

With the aim of exemplifying the proposed framework, an example of a train set crossing was selected. An obvious advantage of using a train set instead of a real railway system is that safety strategies can be studied and implemented without endangering the travelling public. The train set crossing described below raises safety-critical issues that are similar to those found at the traditional level crossing (i.e. road–rail).

The physical process consists of two track circuits *Cp* and *Cs*, and two types of trains – primary (*Trp*) and secondary (*Trs*). The circuits are divided into sections (numbered in a clockwise direction) and there are two separate crossing sections *CCa* and *CCb* at which they intersect. Trains of type *Trp* travel around circuit *Cp* and trains of type *Trs* travel around circuit *Cs*. Both types of train travel in one direction (clockwise) only, hence trains cannot reverse around the circuits. The longest train is shorter than the smallest section. A crossing section is that part of the track which consists of the sections (one from each circuit) at which the two circuits intersect. The circuits *Cp*, *Cs* and the crossing sections *CCa* and *CCb* are illustrated in figure 1.

The mission of the train set is to ensure that all trains complete their journeys, while allowing all concurrent movements of the trains. This mission is in fact a special case of the Merlin-Randell problem,[10] which is concerned with train journeys on arbitrary tracks. This problem has been heavily studied, and involves a complex analysis of synchronisation strategies.[7] A special case of synchronisation is that the primary trains must always take priority over the secondary trains at the crossing sections

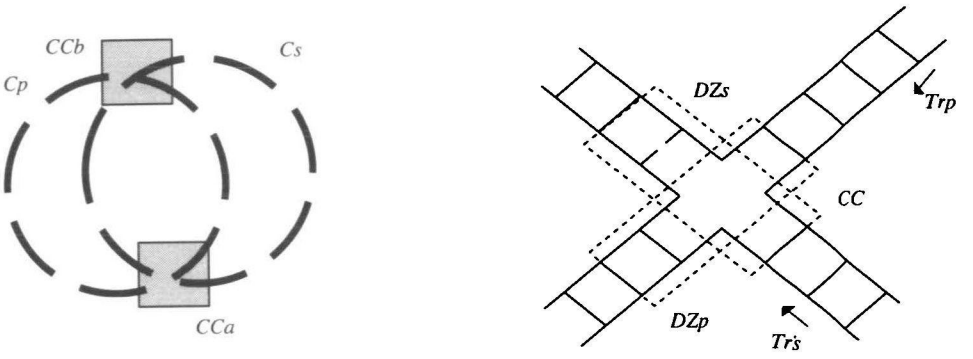


**Figure 1. The train set circuits and the crossing section.**

– that is, a primary train must not be made to wait for a secondary train (c.f. priority of trains over road vehicles at a level crossing). For the train set, one of the main advantages in separating the mission from the safety requirements is that synchronisation issues can be ignored in the analysis of the safety controller. As a result issues involving problems of optimal use of the sections of the tracks, routes of the trains, and moving stopped trains can be ignored during the analysis of the safety requirements. By focusing on the safety issues only, a thorough verification of the safety controller specification becomes practical. However, as will be shown later, the mission requirements must be considered to ensure that any proposed safety strategy does not lead to the implementation of a safety controller strategy that is inconsistent with the mission requirements.

*Conceptual analysis.* Several disasters are possible with this system, but we only consider the following disasters:

(*a*) trains of the same type collide;

(*b*) trains of different type collide.

The hazardous states for the collision of trains of the same type are identified by considering the relative position of the trains on a circuit; a collision can occur only if some part of two trains are in the same section. Hence, since the longest train is smaller than the smallest section, a state is deemed to be hazardous if the front of one train is in the same, or adjacent, section as the front of another train.

The hazardous states for the collision of trains of different type are identified by considering the relative positions of the primary and secondary trains to a crossing section. A collision involving a primary train and a secondary train can occur only when both trains are in the same crossing section. Since the length of a train is less than a section, a train is in a crossing section only if the front of that train is in the crossing section or the section that follows the crossing section. To express the hazard associated with a crossing section, for each circuit we introduce the notion of danger zones (one for each crossing section) as the set of sections in which the front of a train can be while that train is in a crossing section. We conclude that the hazardous states for the collision of trains of different type are those in which the front of a primary train and the front of a secondary train are in the danger zones of the same crossing section. The danger zones *DZp* and *DZs* of a crossing section *CC* are illustrated in figure 1.

### 4.1. Safety requirements analysis

The safety requirements analysis is concerned with the identification of safety constraints and the definition of safety strategies. We consider each potential disaster separately.

*General model.* The type of circuit is denoted by $c \in L$, $L = \{p, s\}$, the crossing section by $r \in R, R = \{a, b\}$, the trains which run on $Cc$ are denoted by $x, y \in Trc = \{1, \ldots, Ntc\}$ and the sections of $Cc$ are denoted by $i, j, k, m, n \in Sc, Sc = \{0, \ldots, Nsc\}$. The lifetime of the system is denoted by the set $T$. Addition $\oplus$ and subtraction $\ominus$ on circuit section numbers are performed modulo the number of sections in the circuit. The behaviour of trains is captured by the state variable *P*train, described below in Table 1. *P*train$(c, x)$ denotes the state variable for the position of train $x$ on circuit $Cc$.

In the general model of the train set it is assumed that the position of the front of the trains is known at any time point during the system lifetime (captured by *P*train). Hence, at the start of the system lifetime the fronts of the trains are positioned in specified known sections.

#### *4.1.1. Collision of trains of same type*

*Circuit safety constraint.* The hazardous states for a circuit $Cc$ can be expressed, in the general model, by the system predicate: $\exists x, y \in Trc: \; x \neq y \wedge P\text{train}(c, x) \in \{P\text{train}(c, y) \ominus 1, P\text{train}(c, y)\}$.

Thus we deduce that a safety constraint of the form 'for any two trains there must be at least one section between the sections containing the fronts of the trains' will, if maintained on circuit $Cc$, prevent the occurrence

**Table 1**

| No. | Name | Range | Comments |
|---|---|---|---|
| $p_1$ | *P*train | $Sp^{Ntp} \times Ss^{Nts}$ | The position of each train expressed as a section number, that is, the section containing the front of a train. |

**Table 2.**

| No. | Name | Range | Comments |
|---|---|---|---|
| $p_2$ | *R*train | $Rp^{Ntp} \times Rs^{Nts}$ | The reservation sets of the trains, where $Rc = \mathcal{P}(Sc)$ for $c \in L$. |

of any hazardous state on *Cc*. This safety constraint $SC_1(c)$ is expressed in terms of a system predicate: $\forall x, y \in Trc: x \neq y \Rightarrow |Ptrain(c, x) - Ptrain(c, y)| > 1$. As a consequence of the safety constraint we can deduce a limit on the number of trains in terms of the sections: $Nsc \geqslant 2Trc$ (for each circuit).

*Circuit safety strategy*. The safety strategy for a circuit *Cc* is based on a reservation scheme; the basic rules of this scheme are:

(*a*) for any train, the current section (i.e. the position of the front of the train) and the section behind the current section must always be reserved;

(*b*) no section can be reserved by more than one train.

The notion of sections reserved by a train is captured by the state variable *R*train, described above in Table 2. *R*train$(c, x)$ denotes the reservation set of train $x$ on circuit *Cc*.

The circuit safety strategy $SS_1(c)$ can be formalised as two system predicates:

(*a*) $\forall x \in Trc$:
$\{Ptrain(c, x) \ominus 1, Ptrain(c, x)\} \subseteq Rtrain(c, x)$;

(*b*) $\forall x, y \in Trc: x \neq y \Rightarrow Rtrain(c, x) \cap Rtrain(c, y) = \varnothing$.

We define $\Gamma H$ as the set of all functions of the form $H: T \rightarrow Sp^{Ntp} \times Ss^{Nts} \times Rp^{Ntp} \times Rs^{Nts}$. We say that a history $H$ from $\Gamma H$ satisfies safety strategy $SS_1(c)$ if and only if rules *a* and *b* are invariant relations (i.e., they are satisfied at all time points in $T$) for that history.

**Lemma 4.1.** *A history that satisfies the safety strategy* $SS_1(c)$ *must satisfy safety constraint* $SC_1(c)$.

**Proof.** (By contradiction.) Assume
$\exists H \in \Gamma H: H \textbf{ sat } SS_1(c) \wedge$
$\exists t \in T: H \textbf{ sat}(\exists x, y \in Trc: x \neq y \wedge$
$|Ptrain(c, x) - Ptrain(c, y)| \leqslant 1)@\mathbf{t}$.
Then from rule *a* of $SS_1(c)$:
$H \textbf{ sat}(\{Ptrain(c, x) \ominus 1, Ptrain(c, x)\} \subseteq Rtrain(c, x) \wedge$
$\{Ptrain(c, y) \ominus 1, Ptrain(c, y)\} \subseteq Rtrain(c, y))@\mathbf{t}$,
hence $H \textbf{ sat}(Rtrain(c, x) \cap Rtrain(c, y) \neq \emptyset)@\mathbf{t}$. But this contradicts rule *b* of $SS_1(c)$. Therefore, $\forall H \in \Gamma H: H \textbf{ sat } SS_1(c) \Rightarrow H \textbf{ sat } SC_1(c)$.

**Corollary 4.2.** *A history that satisfies the safety strategies* $SS_1(p)$ *and* $SS_1(s)$ *must satisfy safety constraints* $SC_1(p)$ *and* $SC_1(s)$.

**Proof.** Immediately from lemma 4.1.

*Circuit safety strategy and mission requirements*. The aim of the framework presented in this paper is to restrict the requirements analysis of safety-critical software to the safety issues of the system. However, it is usually impossible to maintain a complete dichotomy between the mission and safety requirements, because it would be futile to impose safety requirements which were so stringent that the system could not satisfy its mission. Here we consider the impact of safety strategy $SS_1(c)$ on the mission requirements of the train set. More specifically, our aim is to ensure that $SS_1(c)$ does not lead to the definition of an over restrictive safety controller strategy; the safety controller must not impose restrictions on the movement of trains in addition to those imposed by the mission controller (except when the system enters an unsafe state). The restrictions imposed on the movement of trains can be stated in terms of the reserved sections; the upper bound of the reservation set of the safety controller for a train should be less than the lower bound of the reservation set of the mission controller.

To determine the upper bound of the reservation set of the safety controller we consider how the reservation set of a train $x$ is modified as the train travels around the circuit *Cc*. Firstly, we make the observation that before train $x$ enters a new section it must reserve that section, i.e. the sections $Ptrain(c, x) \ominus 1$, $Ptrain(c, x)$ and $Ptrain(c, x) \oplus 1$ must be reserved. Secondly, immediately after the front of train $x$ enters a new section the position of a train is updated, hence sections $Ptrain(c, x) \ominus 2$, $Ptrain(c, x) \ominus 1$ and $Ptrain(c, x)$ are reserved. Thirdly, while a train is travelling in a section, the current section and the previous section are reserved. Therefore an upper bound for the reservation set is: $\{Ptrain(c, x) \ominus 2, Ptrain(c, x) \ominus 1, Ptrain(c, x), Ptrain(c, x) \oplus 1\}$.

From the above analysis we can conclude that the safety strategy is not too restrictive on the movement of trains if the lower bound of the reservation set of the mission controller is at least $\{Ptrain(c, x) \ominus 2, Ptrain(c, x) \ominus 1, Ptrain(c, x), Ptrain(c, x) \oplus 1\}$. If this condition could not be satisfied then the mission requirements would have to be modified or a new safety strategy devised.

Here we construct a modified version of $SS_1(c)$, denoted by $SS_1(c)^*$, that includes the upper bound on the size of the reservation set; the two rules for $SS_1(c)^*$ are given below:

(*a*) $\forall c \in L$:
$\forall x \in Trc$: $\{Ptrain(c, x) \ominus 1, Ptrain(c, x)\} \subseteq Rtrain(c, x) \wedge Rtrain(c, x) \subseteq \{Ptrain(c, x) \ominus 2, Ptrain(c, x) \ominus 1, Ptrain(c, x), Ptrain(c, x) \oplus 1\}$;

(*b*) $\forall x, y \in Trc$:
$x \neq y \Rightarrow Rtrain(c, x) \cap Rtrain(c, y) = \emptyset$.

**Lemma 4.3.** *A history that satisfies the safety strategy* $SS_1(c)^*$ *must satisfy safety constraint* $SC_1(c)$.

**Proof.** Follows directly from lemma 4.1, and the fact that $SS_1(c)$ is a consequent of $SS_1(c)^*$.

**Corollary 4.4.** *A history that satisfies the safety strategies* $SS_1(p)^*$ *and* $SS_1(s)^*$ *must satisfy safety constraints* $SC_1(p)$ *and* $SC_1(s)$.

**Proof.** Immediately from lemma 4.3.

### *4.1.2. Collision of trains of different type*

The following analysis can be applied to both crossing sections provided they are sufficiently far apart. That is, on both circuits there must be at least one section

between the two crossing sections (which will ensure that the danger zones do not overlap).

*Crossing section safety constraint.* Let $CCp(r)$ (resp. $CCs(r)$) denote the number of the section of $Cp$ (resp. $Cs$) that is part of $CC(r)$. The danger zone of $Cc$ for $CC(r)$ is defined as: $DZc(r) = \{CCc(r),\ CCc(r) \oplus 1\}$. The hazardous states for the crossing section $CC(r)$ can be expressed by the system predicate: $\forall c \in L$: $\exists x \in Trc$: $Ptrain(c, x) \in DZc(r)$.

Thus we deduce that a safety constraint of the form 'either the front of no primary train is in the danger zone $DZp(r)$ or the front of no secondary train is in the danger zone $DZs(r)$' will, if maintained, prevent the occurrence of any hazardous state. This safety constraint ($SC_2(r)$) can be expressed in terms of a system predicate: $\exists c \in L$: $\forall x \in Trc$: $Ptrain(c, x) \notin DZc(r)$. Assuming that we wish to have at least one train on each circuit we can deduce that $Nsp \geqslant 3 \wedge Nss \geqslant 3$.

*Crossing Section Safety Strategy.* The crossing safety strategy is based on a modification of the reservation scheme. The two basic rules are:

(*a*) if any train $x$ on circuit $c$ is in a danger zone then the crossing section contained within that danger zone is reserved for circuit $c$;

(*b*) section $CCp(r)$ and section $CCs(r)$ cannot both be reserved.

The crossing section safety strategy ($SS_2(r)$) is formalised as two system predicates:

(*a*) $\forall c \in L$: $\forall x \in Trc$:
$Ptrain(c, x) \in DZc(r) \Rightarrow CCc(r) \in Rtrain(c, x)$;

(*b*) $\exists c \in L$: $(\forall x \in Trc$: $CCc(r) \notin Rtrain(c, x))$.

We will say that a history $H$ from $\Gamma H$ satisfies this safety strategy if and only if system predicates $a$ and $b$ are invariant relations for that history.

**Lemma 4.5.** *A history that satisfies the crossing safety strategy $SS_2(r)$ must satisfy safety constraint $SC_2(r)$.*

**Proof.** (By contradiction.) Assume
$\exists H \in \Gamma H$: $H$ **sat** $SS_2(r) \wedge$
$\exists t \in T$: $H$ **sat**$(\exists x \in Trp$: $Ptrain(p, x) \in DZp(r) \wedge$
$\exists y \in Trs$: $Ptrain(s, y) \in DZs(r))$ @ **t**.
Then from rule $a$ of $SS_2(r)$:
$H$ **sat**$(CCp(r) \in Rtrain(p, x) \wedge CCs(r) \in Rtrain(s, y))$ @ **t**.
But this contradicts rule $b$ of $SS_2$. Therefore $\forall H \in \Gamma H$: $H$ **sat** $SS_2(r) \Rightarrow H$ **sat** $SC_2(r)$.

**Corollary 4.6.** *A history that satisfies the safety strategies $SS_2(a)$ and $SS_2(b)$ must satisfy safety constraints $SC_1(a)$ and $SC_1(b)$.*

**Proof.** Immediately from lemma 4.5.

*Crossing section safety strategy and mission requirements.* At this stage, the safety requirements for the crossing sections take no account of the conflict with the mission requirement that trains on the primary circuit have priority; there is no mechanism which establishes that primary trains should pass first. To resolve this, we should not modify the safety controller specification, since this contradicts the philosophy of our framework, i.e. to keep the safety controller specification as simple as possible. A solution is to adapt the mission requirements to allow a secondary train to take the decision to stop before the crossing sections, in order to avoid a conflict.[15]

*Combination of safety strategies.* In terms of the two circuits and the two crossing sections the safety controller must implement the safety strategies for circuits $Cp$ and $Cs$ and for the crossing sections $CCc(a)$ and $CCc(b)$. Here we analyse the relationship between these safety strategies and the restrictions imposed on the size of the circuits and the number of trains.

By inspecting the rules of the safety strategies we can deduce that the following three rules are equivalent to $SS_1(p)^* \wedge SS_1(s)^* \wedge SS_2(a) \wedge SS_2(b)$:

(*a*) $\forall c \in L$: $\forall x \in Trc$:
$\{Ptrain(c, x) \ominus 1,\ \ Ptrain(c, x)\} \subseteq Rtrain(c, x) \wedge$
$Rtrain(c, x) \subseteq \{Ptrain(c, x) \ominus 2,\ \ Ptrain(c, x) \ominus 1,$
$Ptrain(c, x),\ Ptrain(c, x) \oplus 1\}$;

(*b*) $\forall c \in L$: $\forall x, y \in Trc$:
$x \neq y \Rightarrow Rtrain(c, x) \cap Rtrain(c, y) = \theta$;

(*c*) $\forall r \in R$: $\exists c \in L$: $(\forall x \in Trc$: $CCc(r) \notin Rtrain(c, x))$.

Rule $a$ is concerned with the reservation of sections (i.e. maintaining the reservation sets) and rules $b$ and $c$ impose mutual exclusion conditions over the reservation sets. We must confirm that the above three rules are not in conflict. The set of states which satisfy the three rules are characterised by the following conditions over the relative position of the trains: $\forall c \in L$: $\forall x, y \in Trc$: $|Ptrain(c, x) - Ptrain(c, y)| > 1 \wedge \forall r \in R$: $\exists c \in L$: $\forall x \in Trc$: $Ptrain(c, x) \notin DZc(r)$. These states are those that satisfy the circuit and crossing section safety constraints, hence the restrictions on the circuit size are obtained by examining the restrictions for the safety constraints: $Nsp \geqslant \max(2Trp, 3) \wedge Nss \geqslant \max(2Trs, 3)$.

The above restrictions are minimal constraints on the circuit size. For practical purposes, issues other than the satisfaction of the safety strategies must be considered, such as the ability of the safety controller to implement the safety strategies in the identified states and properties of the physical process. For example, the conditions given above do not preclude the possibility of a deadlock in the physical process. To prevent this situation, the conditions on the circuit sizes should be strengthened to strict inequalities.

Inspection of the three rules above also gives insight into the implementation of the safety strategies. More specifically, the reservation sets must be maintained for each circuit, and mutual exclusion must be achieved for reserved sections for the sections of each circuit and the sections $CCp(r)$ and $CCs(r)$ (i.e. the sections that make $CC(r)$).

### 4.2. Safety system analysis

After establishing the safety strategies during the Safety Requirements Analysis, the Safety System Analysis phase investigates how these strategies are to be implemented by the safety controller as *safety controller strategies*; or alternatively; the safety strategies must be mapped onto a set of sensors and actuators. In the train set, at the start of every section there is a sensor which detects the presence of a train, and an actuator which allows the safety controller to stop the train within a section. To simplify the analysis we assume that neither the sensors nor the actuators fail.

The analysis to be performed entails modelling the relationship between the components of the safety controller (including sensors and actuators), the physical process, and the interface between them. The general approach followed in modelling the system is to maintain

a clear separation between models of the physical process and the safety controller, even though they must cooperate whenever an action is to be performed. The advantage of adopting this approach is that both models can be independently developed and modified, and the state of the physical process can be seen to correspond to the sequence of control commands issued by the safety controller.

The analysis will be divided into two parts, corresponding to the safety strategies established during the Safety Requirements Analysis. First, we model the safety strategy which prevents the collision of trains of the same type, and second, we model the safety strategy which prevents the collision of trains of different types.

### *4.2.1. Collision of trains of the same type*

To illustrate modelling the behaviour of trains in a circuit, we assume that each circuit contains seven sections ($Ns = 6$) and two trains ($Nt = 2$). For simplicity, the modelling and analysis is performed for just one circuit. The PrT model is shown in figure 2 (because of the flexibility of the model it is possible to modify the number of sections and trains without affecting the structure of the model of the safety controller). An outline definition of PrT nets is presented in the Appendix.

The predicates of the PrT net model of a train set circuit are the following:
$S0x$ to $S6x$ – train $x$ occupies a section of the circuit;
$ICPxj$ – train $x$ is allowed to enter section $j$;
$IPCxj$ – train $x$ has entered section $j$;
$FSn$ – section $n$ is not reserved by the safety controller;
$RS1xm$ – train $x$ has reserved section $m$;
$RS2xk + xj$ – train $x$ has (temporarily) reserved sections $k$ and $j$.

To obtain the structural properties of the PrT net model of the circuit, we derive the $S$-invariants of the net; these are integer equation invariants which are obtained from the projection of the predicates.[3] In order to simplify the calculation of the $S$-invariants of the PrT net model of figure 2, the predicates representing the circuit sections of the model of the physical process were folded into one predicate ($Sxj$ – train $x$ occupies section $j$).

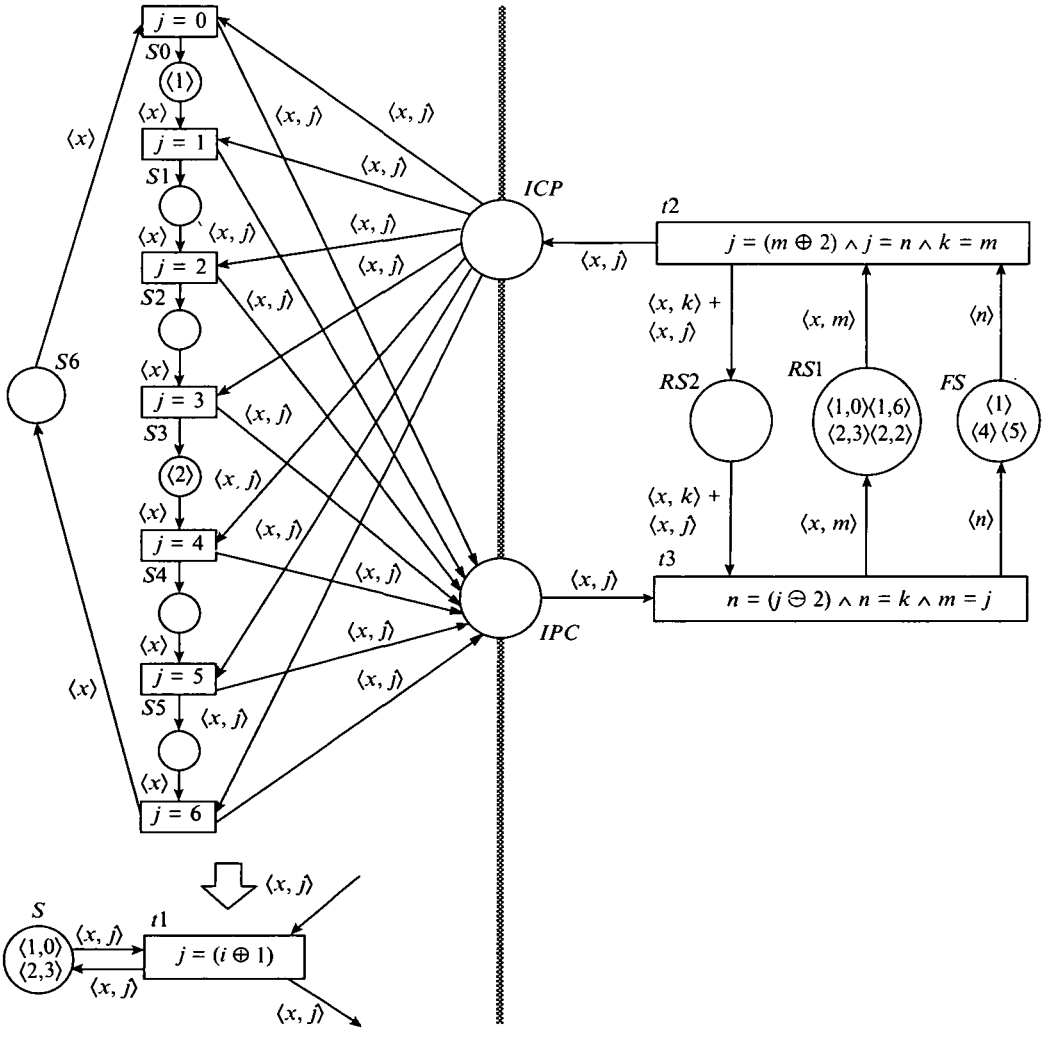


**Figure 2. The PrT net model of the train set circuit.**

The $S$-invariants of the PrT net model of the safety controller strategies of the circuit are:

$|RS2|_2 + 2|RS1|_2 = 8;$ (*si1*)

$|IPC| + |ICP| + |FS| = 3;$ (*si2*)

$|RS1| - |FS| = 1;$ (*si3*)

$|RS2| + 2|FS| = 6.$ (*si4*)

**Lemma 4.7.** *The* PrT *net model of the train set circuit shown in figure 2 is contact-free and deadlock-free.*

To confirm that a PrT net is contact-free, we must show that there is no marking that puts the same tuple at the same predicate more than once. For the PrT net of figure 2, this follows immediately from equations (*si1*) and (*si2*).

To confirm that a PrT net is deadlock-free, we must show that there is no forward reachable marking at which all transitions are disabled. By inspecting the PrT net of figure 2, we show that if *t2* and *t3* are disabled then *t1* must be enabled. Firstly, we consider the case when *t2* is disabled, and secondly when *t3* is disabled.

(a) If *t2* is disabled, this implies that the tuples $\langle x, m\rangle$ in *RS*1 and $\langle n\rangle$ in *FS* are not sufficient to satisfy the transition selector of *t2*. This condition can be confirmed from (*si3*) from which we obtain that $|RS1| = 2$ and $|FS| = 1$. Hence from (*si1*) we have $|RS2|_2 = 4$ and from (*si2*) we have $|IPC| + |ICP| = 2$, which leads to two possible cases: if $|IPC| \neq 0$ then *t3* would be enabled, however $|IPC| = 0$ implies that *t1* is enabled.

(b) If *t3* is disabled, this implies that there are no tuples in the predicates *RS*2 and *IPC*, or there is neither $\langle x,j\rangle$ in *IPC* nor $\langle x,k\rangle + \langle x,j\rangle$ in *RS*2. If there are no tuples in *RS*2 and *IPC* then it follows from (*si1*) and (*si3*) that the number of tuples in *RS*1 and *FS* are identical to those in the initial marking, and it can be easily shown that *t2* is enabled. If there are no tuples in *RS*2 then it follows from (*si2*) and (*si4*) that $|IPC| + |ICP| = 0$ implying that there are also no tuples in *IPC*, in which case the number of tuples in *RS*1 and *FS* are identical to the previous case. If there are no tuples in *IPC* then it follows from (*si2*) that $|ICP| + |FS| = 3$ which leads to two possible cases: if $|FS| = 3$ then $|ICP| = |RS2| = 0$ in which case the number of tuples in *RS*1 and *FS* are identical to the initial marking, but if $|FS| \neq 3$ then $|ICP| \neq 0$ implies that *t1* must be enabled.

*Verification of circuit safety controller strategy.* The safety controller strategy defined by the PrT net model of figure 2 is verified by proving that the safety strategy $SS_1(c)$ is a property of the PrT net model. To verify this, a link must be identified between THL and PrT net model of the train set system. This link can be established in terms of system predicates of the THL model and the predicates of the PrT net model, as follows:

$P\text{train}(c,x) = i \Leftrightarrow Sxi;$

$i \in R\text{train}(c,x) \Leftrightarrow RS1xi \vee RS2xi.$

The two rules of $SS_1(c)$ can be expressed in terms of the PrT net model as logical formulae (*lf1* and *lf2*) over the predicates of the PrT net model, by substituting the equivalent predicates of the PrT net model for the THL predicates.

*lf1.* If any train $x$ is in section $i$ then sections $i$ and $i \ominus 1$ are reserved by train $x$.

$$(\forall x)(\forall i)[Sxi \Rightarrow (RS1xi \vee RS2xi) \wedge (RS1xi \ominus 1 \vee RS2xi \ominus 1)].$$

*lf2.* Any section $i$ is reserved by at most one train.

$$(\forall x)(\forall y)(\forall i)[x \neq y \Rightarrow (\neg RS1xi \wedge \neg RS2xi) \vee (\neg RS1yi \wedge \neg RS2yi)].$$

**Lemma 4.8.** *The formulae lf1 and lf2 are logical invariants (i.e. they hold on all reachable markings) of the PrT net model.*

**Proof.** A sketch of a proof is given below by analysing the transitions of the PrT net model. In the initial marking *lf1* holds; both trains have reserved the current and previous sections:

$\langle 1,0\rangle \in S \Rightarrow \{\langle 1,0\rangle, \langle 1,6\rangle\} \subset RS1 \cup RS2;$

$\langle 2,3\rangle \in S \Rightarrow \{\langle 2,3\rangle, \langle 2,2\rangle\} \subset RS1 \cup RS2.$

We show that the firing of the three transitions cannot violate *lf1*.

Transition *t1* can fire with $\langle x,i\rangle$ only when $\langle x,j\rangle$ is in *ICP* ($j = i \oplus 1$), therefore only after *t2* fires producing $\langle x,j\rangle$ in *ICP* and *RS*2. Prior to the firing of *t1*, $\{\langle x,i\rangle, \langle x, i \ominus 1\rangle\} \subset RS1 \cup RS2$, after *t1*, $\{\langle x,j\rangle, \langle x,i\rangle\} \subset RS1 \cup RS2$ (since $\langle x,j\rangle$ must remain in *RS*2 until *IPC* fires producing $\langle x,j\rangle$).

Firing *t2* adds $\langle x,j\rangle$ to $RS1 \cup RS2$, therefore *t2* cannot violate *lf1*.

Firing *t3* removes $\langle x,k\rangle$ from $RS1 \cup RS2$ only when $\langle x,j\rangle$ is in *S* and from the transition selector of *t3* ($k = j \ominus 2$), therefore *t3* cannot violate *lf1*.

In the initial marking *lf2* holds; no section is reserved by more than one train:

$RS1 \cup RS2 = \{\langle 1,0\rangle, \langle 1,6\rangle, \langle 2,3\rangle, \langle 2,2\rangle\}.$

Firing *t1* does not change the markings on *RS*1 or *RS*2.

Firing *t2* removes $\langle x,m\rangle$ from *RS*1 and adds $\langle x,k\rangle + \langle x,j\rangle$ to *RS*2, where $k = m$ and $j = m \oplus 2$. The only change in $RS1 \cup RS2$ is the addition of $\langle x,j\rangle$. From the transition selector of *t2*, $j$ ($= n$) is a free section, hence prior to *t2* being fired $\neg(\exists y)\ \langle y,j\rangle \in (RS1 \cup RS2)$. Therefore *t2* cannot violate *lf2*.

Firing *t3* removes $\langle x,k\rangle$ from $RS1 \cup RS2$, therefore *t3* cannot violate *lf2*.

### *4.2.2. Collision of trains of different type*

For simplicity, the modelling and analysis of the behaviour of trains in a crossing section is performed for just one crossing section, and we assume that each circuit contains four sections ($Ns = 3$) and one train ($Nt = 1$), as shown in figure 3. In the PrT net model of the physical process we have folded the models of the primary and the secondary circuits into one net; the danger zone of each circuit is represented by the predicates *CC* and *S*0, and the predicate *S*2 represents the section which immediately precedes *CC*.

The predicates of the PrT net model of the crossing section are the following:

*S*0*cx*, *S*1*cx*, *S*2*cx* – a section of the circuit $c$ is occupied by train $x$;

*CCcx* – the crossing section of circuit $c$ is occupied by train $x$;

*ICPcxj* – train $x$ in circuit $c$ is allowed to enter section $j$;

*IPCcxj* – train $x$ in circuit $c$ has entered section $j$;

*APcxj* – train $x$ in circuit $c$ is in section $j = 2$, (i.e. $x$ is about to enter *DZ*);

*ZDcxj* – train $x$ in circuit $c$ has access to section $j$ ($= 3$, the crossing section);

*ME* – either primary or secondary trains allowed to enter the crossing section.

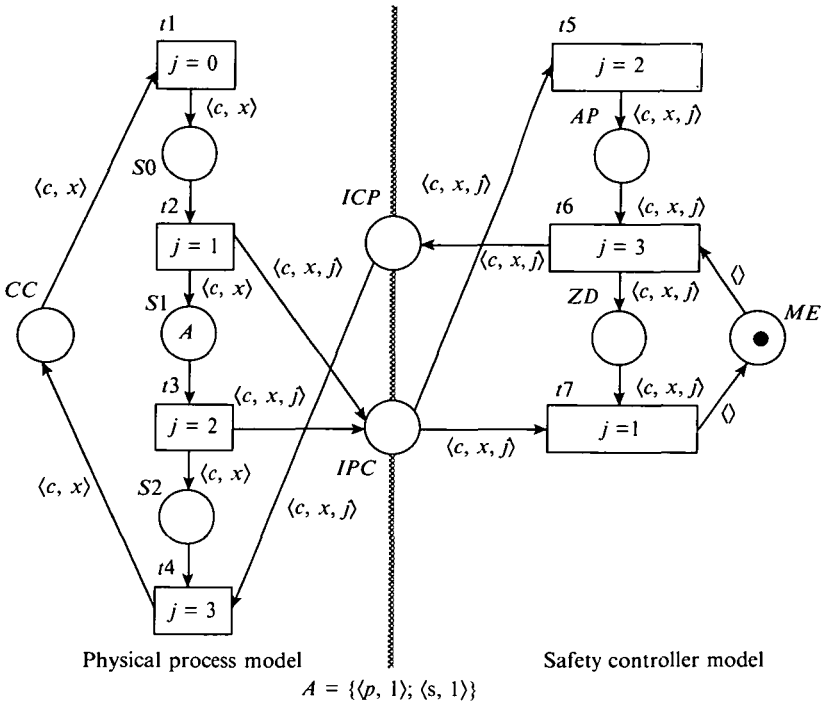


**Figure 3. The PrT net model of the crossing section.**

To obtain the structural properties of the PrT net model of the crossing section, we derive the $S$-invariants of the net, which are:

$S0+S1+S2+CC = 2;$ (*si1*)

$ME+|ZD| = 1;$ (*si2*)

$S1+2S2-2|ICP|_3-|IPC|_3-|AP|_3+|ZD|_3 = 2;$ (*si3*)

$S0-S2+CC+2|ICP|_3+|IPC|_3+|AP|_3-|ZD|_3 = 0;$ (*si4*)

$2S0+S1+2CC+2|ICP|_3+|IPC|_3+|AP|_3-|ZD|_3 = 2.$ (*si5*)

**Lemma 4.9.** *The PrT net model of the train set crossing section shown in figure 3 is contact-free and deadlock-free.*

**Proof.** The contact-freeness of the PrT net model of the crossing section is shown through inspection of (*si1*), (*si2*), (*si4*) and (*si5*). From (*si1*) we know that the number of tuples in the net model of the physical process remains constant and equal to the initial marking. From (*si2*) we know that there is no increasing number of tokens in the predicates *ME* and *ZD*. Finally, from (*si4*) and (*si5*) we know that *ICP*, *IPC* and *AP* are contact-free, implying that the number of tuples in the whole PrT net model remains constant.

To confirm that the PrT net model of the crossing section, figure 3, is deadlock-free we show that if *t1*, *t2*, *t3*, *t5*, *t6* and *t7* are disabled then *t4* must be enabled. The approach to be followed here is to consider each of the transitions which we assume to be disabled individually and show that *t4* is enabled implying deadlock-freeness.

If *t1* is disabled, it means that there are no tuples in *CC*. From (*si1*) it follows that if *S*0 or *S*1 have tuples, then transitions *t2* or *t3* must be enabled. However, if the two tuples are in *S*2 then (*si4*) is reduced to $2|ICP|_3+|IPC|_3+|AP|_3-|ZD|_3 = 2$. From this equation, if $|ZD| = 0$ then there are no tuples in *ICP* (reasoning from the PrT net model) the two tuples can either be in *IPC* or *AP*, implying that at least *t5* or *t6* are enabled; if $|ZD| \neq 0$ then *t6* is disabled, one of the tuples is in either *IPC* or *AP*, implying that *t5* or *t6*, respectively, are enabled, and the other tuple is in *ICP*, implying that *t4* is enabled.

If we apply a similar reasoning to the other transitions that we assume to be disabled, we will be able to show that *t4* will be enabled.

*Verification of crossing section safety controller strategy*. The safety controller strategy defined by the PrT net model of figure 3 is verified by proving that the safety strategy $SS_2(r)$ is a property of the PrT net model.

The THL predicates and the predicates of the PrT net model correspond as follows:

$Ptrain(c, x) \in DZc(r) \Leftrightarrow CCcx \vee S0cx;$

$CCc(r) \in Rtrain(c, x) \Leftrightarrow ZDcxj.$

The two rules of $SS_2(r)$ can be expressed in terms of the PrT net model as logical formulae (*lf1* and *lf2*) over the predicates of the PrT net model, by substituting the equivalent predicates of the PrT net model for the THL predicates.

*lf1*. If any train is in a danger zone then the crossing section is reserved by that train.

$(\forall c)(\forall x)[CCcx \vee S0cx \Rightarrow ZDcxj].$

*lf2*. A crossing section cannot be reserved for both the primary circuit and the secondary circuit.

$(\exists c)(\forall x)[\neg ZDcxj].$

**Lemma 4.10.** *The formulae lf1 and lf2 are logical invariants (i.e. they hold on all reachable markings) of the PrT net model.*

**Proof.** A sketch of a proof is given by analysing the transitions and $S$-invariants of the PrT net model.

In the initial marking *lf1* holds, since the sections *CC* and *S*0 are empty: $(CC \cup S0) = \emptyset$.

Firstly, we make the observation that tuples can be added to the set $CC \cup S0$ only by transition *t4* and removed only by transition *t2*. In the following we argue that $\langle c, x, j\rangle \in ZD$ before *t4* adds $\langle c, x\rangle$ to $CC \cup S0$ and until *t2* removes $\langle c, x\rangle$ from $CC \cup S0$.

Transition *t4* can fire with $\langle c, x\rangle$ only if $\langle c, x, j\rangle$ is in *ICP*, therefore after *t6* fires with $\langle c, x, j\rangle$. Hence before *t4* fires $\langle c, x, j\rangle \in ZD$. Now $\langle c, x, j\rangle$ can be removed from *ZD* only when *t7* fires with $\langle c, x, j\rangle$; this transition can fire only when $\langle c, x, j\rangle \in ICP$. Therefore *t7* can fire with $\langle c, x, j\rangle$ only after *t2* has fired with $\langle c, x, j\rangle$. Hence $\langle c, x, j\rangle \in ZD$, at least until *t2* fires.

The fact that *lf2* is a logical invariant for the PrT net model follows from *S*-invariant (*si2*).

## 5. CONCLUSIONS

The basic aim of this paper is to present a general framework for the specification and verification of safety requirements in the software development of safety-critical systems. The framework consists of the following steps. From the system conception, and after the identification of the system disasters and hazards, the first phase is the description of the real world properties in the form of safety constraints and safety strategies; these notions are specified in terms of a logical formalism. The interrelationships between the components of the safety controller are expressed as the safety controller strategies; these strategies are specified in terms of a net formalism. The general approach was shown to be feasible by applying it to the example of the train set crossing.

In the paper we have identified just two distinct phases for the analysis of the safety requirements, but a larger number of phases could be utilised if this was desirable, depending on the type of application and its level of criticality. The choice of the type of formalism to be used, at each phase, is influenced by the issues which are most significant at that phase, and the potential that certain formalisms have in representing these issues. This approach enables a fuller exploitation of the appropriate features of the different formalisms than can be achieved with a single-formalism approach.

**Acknowledgements**

The authors would like to acknowledge the financial support of BAe (DCSC), CAPES/Brazil and ESPRIT Basic Research Action PDCS.



## REFERENCES

1. P. Bishop, Invariants as an alternative to Petri nets for safety design. *EWICS TC7, WP* 498. (1986).
2. V. Chandra and M. R. Verma, A fail safe interlocking system for railways. *IEEE Design & Test of Computers,* **8** (1), 58–66 (1991).
3. H. Genrich, Predicate/transition nets. *Petri Nets: Central Models and their Properties*. Edited W. Brauer, W. Reisig and G. Rozemberg. Lectures Notes in Computer Science **254**, 206–47 (1987).
4. J. Gorski, Design for safety using temporal logic. *SAFECOMP'86*. Sarlat, France. 149–155 (1986).
5. M. S. Jaffe, N. G. Leveson, M. P. E. Heimdahl and B. E. Melhart, Software requirements analysis for real-time process-control systems. *IEEE Transactions on Software Engineering*, **SE-17** (3), 241–258 (1991).
6. F. Jahanian and A. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, **SE-12** (9), 890–904 (1986).
7. M. Koutny, The Merlin-Randell problem of train journeys. *Acta Informatica*, **23** (4), 429–463 (1986).
8. J. C. Laprie, Dependability: basic concepts and associated terminology. *ESPRIT PDCS Report No 31* (1990).
9. N. G. Leveson and J. Stolzy, Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, **SE-13** (3), 386–397 (1987).
10. P. Merlin and B. Randell, Notes on deadlock avoidance on the train set. *Report MRM/144*. Comp. Lab. University of Newcastle upon Tyne (1978).
11. Draft Interim Defence Standard 00–56. *Hazards Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment*. UK Ministry of Defence. London (1991).
12. J. S. Ostroff and W. M. Wonham, Modelling, specifying and verifying real-time embedded computer systems. *Proc. of the Real-Time Systems Symposium 1987*. San Jose, CA. 124–132 (1987).
13. A. Pnueli, Specification and development of reactive systems. *Information Processing 86*. 845–858 (Edited H. J. Kugler, 1986).
14. A. Saeed, T. Anderson and M. Koutny, A Formal model for safety-critical computing systems. *SAFECOMP'90* London, UK 1–6 (1990).
15. A. Saeed, R. de Lemos and T. Anderson, The role of formal methods in the requirements analysis of safety-critical systems: a train set example. *Proceedings of the 21st Symposium on Fault-Tolerant Computing*. Montreal, Canada 478–485 (1991).

## APPENDIX

### Predicate-transition nets (PrT nets)

In the following we present an informal definition of PrT nets; a formal definition is given elsewhere.[3]

Let *S*, *T*, *F* be finite sets. The triple $N = (S, T, F)$ is called a *directed net* iff the following conditions hold: $S \cap T = \emptyset, S \cup T \neq \emptyset, F \subseteq (S \times T) \cup (T \times S)$, and domain $(F) \cup$ codomain $(F) = S \cup T$.

For a given net $N = (S, T, F)$, *S* is the set of *places* of *N*, *T* is the set of *transitions* of *N*, and *F* is the *flow* relation containing the *arcs* of *N*. For $x \in S \cup T, I(x) = \{y \in S \cup T | (y, x) \in F\}$ is called the preset of $x$, and $O(x) = \{y \in S \cup T | (x, y) \in F\}$ is called the postset of $x$.

A PrT net consists of the following constituents:

(1) a directed net $(S, T, F)$ where *S* is the set of predicates, and *T* is the set of transitions;
(2) predicates are variable relations amongst individuals ('first-order' places);
(3) the transitions are schemes of elementary changes of markings representing the actions carried out by the system;

(4) an arc label specifies a variable extension of a predicate to which the arc is connected;
(5) a marking is a mapping that assigns to each predicate formal sums of $n$-tuples of individual symbols, also called tuples.

The graphical representation of a PrT net is obtained by representing a predicate by a circle, a transition by a box, an element of $F \cap (S \times T)$ by a directed arc from a circle to a box, and an element of $F \cap (T \times S)$ by a directed arc from a box to a circle.

For the analysis of the PrT net model we have employed the $S$-invariant method.[3] The $S$-invariants are obtained from the projection of the entries of the incidence matrix $C$ of the net. The projection along the $j$th position of the tuple ($|\,|_j$) introduces a kind of partial cardinal number, by ignoring information at position $j$ in the tuple. As a consequence every solution of $|C|^T|i| = 0$ whose entries do not contain individual variables determines a family of $S$-invariants.

# Book Reviews

CHARLES F. GOLDFARB
*The SGML Handbook*
Oxford University Press, 1990. 664 pp.
ISBN 0 19 853737 9. £50.00.

This book is the result of some twenty years work by the author and others on the development of the Standard Generalized Markup Language, which had led to the publication of ISO 8879, the SGML standard. The main elements of the book are: the up-to-date amended full-text of ISO 8879, a detailed structured overview of SGML and additional tutorial and reference material on SGML. The text of the standard has been extensively annotated by the author.

Charles Goldfarb is regarded as the inventor of the SGML language and was also the technical leader of the team which developed it into an international standard. He was also the leader of the initial research project at IBM which developed GML, on which SGML is based. He is the leading authority on SGML, and this book gives the reader some of his thoughts on the standard.

The book is divided at the highest level into four parts and four appendices. Part one, 'Tutorials', consists of the three tutorial annexes from ISO 8879 and a new tutorial based on the LINK feature. Part two, 'A Structured Overview of SGML', comprises the definitions contained in ISO 8879, explained and ordered to give the reader the key ideas. Part three, 'ISO 8879 Annotated', is the full ISO text with extensive annotations. Part four, 'ISO 8879 Annexes', is purely the full ISO annexes. Appendix A, 'A Brief History of the Development of SGML', covers mark-up concepts, GML, and the progression through to the publication of the standard. Appendix B, 'Recommendations for a Possible Revision of ISO 8879', is the SGML committee document N1035, which details the changes agreed by the developers, should the standard be reviewed. Appendix C, 'About the ISO 8879 Text', describes how this document was used in preparing the book. Appendix D gives details of relevant contacts where additional information on SGML and ISO 8879 may be obtained.

The book has a section at the beginning entitled 'How to Use This Book', which acts in the same way as a READ.ME file which is received with a piece of software. This explains the typographical conventions and also the linking system which allows the book to function as a paper hypertext. This is becoming commonplace with books on this type of subject. In this case it is justified due to the fact that the text of the standard is presented alongside the annotations. The links are necessarily complicated, directing the reader to both page and section number, but moving between pages soon becomes routine. The two ribbon page markers are a very welcome addition.

It is suggested that the book is for those people who wish to understand, use and implement the standard. The book is really too detailed for those people who wish to just learn about the standard, such as students and researchers – there are other publications which are more suitable for this purpose. It certainly, however, fulfils its role as an essential aid for practitioners wishing to develop applications. Its cover price reflects the fact this is an important book in this field. It is a substantial volume which is well produced. Although of some interest to others, practitioners alone will be able to justify its purchase.

CHRISTOPHER HANKINS
*London*

TIMOTHY WEGNER AND MARK PETERSON
*Fractal Creations*
Pitman, London, 1991.
£31.50.

Today, many publishers are seeking authors who can supply an interesting manuscript, and also provide some form of add-on value to convince potential purchasers that the package is value for money. *Fractal Creations* certainly has a useful component of add-on value in the form of a colour poster, a pair of anaglyph 3D glasses, and a 5.25″ disk of fractal programs. The poster is a minor feature and shows 10 examples of fractals rendered with false colours, together with some advertising for the Waite Group Press.

But what about the book? Well, this is a cross between a traditional book and a reference manual. The first chapter is a primer on fractals; the second chapter provides advice on how to control the FRACTINT software environment; and the third chapter supplies extra detail on the individual fractal programs.

The literary style of the first chapter is in a popular vein, with the authors attempting to convince the reader that fractals are a fundamental feature of our universe. Personally, I found some of their descriptions rather loose, for according to Wegner and Peterson virtually everything is a fractal from an atom to a galaxy. I wish that they had given a formal description of a fractal set and then suggested that features of our universe exhibit fractal-like properties. Perhaps this is the price that must be paid for popular writing versus a rigid technical style.

Nevertheless, within the space of 41 pages the reader is rapidly introduced to fractals, chaos, dynamic systems, complex numbers, orbits, the Mandelbrot and Julia sets and attractors. The second chapter, which is just over 100 pages, describes the FRACTINT software environment, and really one should be seated at a computer when reading this, as it provides a thorough guided tour of the software environment. In the last chapter a further 100 pages or so provide useful background information on over 70 fractals with names like Newton's Basin and Popcorn Fractals.

Unfortunately, I was on holiday in France when I started reading *Fractal Creations* and had to wait for my return to the UK before I could investigate the software. I must admit that I was highly sceptical of getting any of the software to work as it claimed to work with CGA, EGA, VGA, SVGA and Hercules graphics adaptors. Turning to the back of the book I broke the envelope's seal storing the disk and accepted the printed warning that I agreed to abide by the accompanying software licence conditions. I also noted the warning that 'colour cycling' can induce hypnotic states.

The entire package was quickly loaded on to my humble Amstrad computer and within minutes I was exploring the Mandelbrot and Julia sets. The book's claim of using high-speed display algorithms is certainly true; several minutes is all that is needed to obtain some excellent pictures, with some in 3D. Although I have not had time to investigate every fractal type supplied, those that I have tested work as claimed, and so far the software has not failed me.

I am sure that I will return to the FRACTINT software in the future and explore other features; in the meantime, I can recommend the book to anyone who is still fascinated by fractals and would like to replace their home-grown algorithms by a well-written integrated software environment.

JOHN VINCE
*Sussex*