ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

**D Duke, University of York, UK and A S Evans, University of Bradford, UK
(Eds)**

BCS-FACS Northern Formal Methods Workshop

Proceedings of the BCS-FACS Northern Formal Methods Workshop,
Ilkley, UK. 23-24 September 1996

A Comparison of Formal Real-Time Specification Languages

A.S. Evans, D.R.W. Holton, L.M. Lai and P. Watson

Springer

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

A Comparison of Formal Real-Time Specification Languages

A.S. Evans

Department of Computing, University of Bradford
Bradford, BD7 1DP, UK

D.R.W. Holton

Department of Computing, University of Bradford
Bradford, BD7 1DP, UK

L.M. Lai

Department of Computing, University of Bradford
Bradford, BD7 1DP, UK

P. Watson

Department of Computing, University of Bradford
Bradford, BD7 1DP, UK

Abstract

In this paper we compare four languages for real time systems specification, namely Timed Z, Timed CSP, Timed CCS and TE-LOTOS, by applying them to the benchmark Railroad Crossing problem. We use slightly different sets of assumptions in each of our solutions in order to investigate how the presence or absence of such assumptions affects the resulting solution. We pay particular attention to the level of justification we may ascribe to each assumption; it may be explicit or implicit in the problem statement, implicit in our knowledge of real-world railroad crossings, or none of these, in which case it must be regarded as a simplifying assumption.

We compare and evaluate the resulting specifications in each of the four languages. Our solution in Timed Z is shown to be on a different level to the three process algebras, being much more abstract, closer to the English specification and further from an implementation. It is argued that the three process algebras have essentially equivalent expressive power over the domain of this problem.

We compare the proofs in each of the process algebra formalisms. Timed CSP has a well developed dedicated formal proof system, while the proof methods required by Timed CCS and TE-LOTOS are much more ad hoc. In these two cases we use proof techniques based on path and state analysis.

We briefly evaluate the Railroad Crossing case study itself. It is found to be a problem of great generality with hidden subtleties; we argue that this problem can teach us much about how to approach real-time specification tasks, and therefore must be considered a highly successful benchmark problem.

1 Introduction

In this paper a number of languages for specifying real time systems are compared. Each language has been used to specify the same benchmark problem, and the resulting specifications and proofs of properties have been evaluated and compared. First the problem to be specified is described, then any further assumptions made are documented. The specifications in Timed Z [5], Timed CSP [2], Timed CCS [1], and TE-LOTOS [11] are presented next, and indications are given as to how the proofs of the required properties proceed. The paper finishes with some conclusions about both the case study and the specification languages used.

2 The Generalised Railway Crossing

The generalised railway crossing problem has been promulgated as a benchmark for the use of formal methods in the specification, design and analysis of real time systems. It has been specified using a range of techniques, a selection of which can be found in [8]. The statement of the problem, taken from [6], is as follows:

The system to be developed operates a gate at a railroad crossing. The railroad crossing I lies in a region of interest R , i.e. $I \subseteq R$. A set of trains travel through R on multiple tracks in both directions. A sensor system determines when each train enters and exits region R . To describe the system formally, we define a gate function $g(t) \in [0, 90]$ where $g(t) = 0$ means the gate is down and $g(t) = 90$ means the gate is up. We also define a set $\{\lambda_i\}$ of *occupancy intervals*, where each occupancy interval is a time interval during which one or more trains are in I . The i th occupancy interval is represented as $\lambda_i = [\tau_i, \nu_i]$ where τ_i is the time of the i th entry of a train into the crossing when no other train is in the crossing and ν_i is the first time since τ_i that no train is in the crossing (i.e. the train that entered at τ_i has exited as have any trains that entered the crossing after τ_i).

Given two constants ξ_1 and ξ_2 , $\xi_1 > 0$, $\xi_2 > 0$, the problem is to develop a system to operate the crossing gate that satisfies the following two properties:

Safety Property: $t \in \bigcup_i \lambda_i \Rightarrow g(t) = 0$ (The gate is down during all occupancy intervals.)

Utility Property: $t \notin \bigcup_i [\tau_i - \xi_1, \nu_i + \xi_2] \Rightarrow g(t) = 90$ (The gate is up when no train is in the crossing.)

This description, though seemingly comprehensive, contains a number of ambiguities and omissions which will be described in the following section.

3 Assumptions and Constraints

3.1 Assumptions

A number of assumptions had to be made about the problem as stated above before progress could be made. The first was related to the positioning of sensors in R . The problem formulation states that $I \subseteq R$, giving no hint as to where the sensors should be in relation to I . It was assumed that there is a sensor to detect when a train starts to enter R , and a sensor to detect when a train has completely left I .

The second major assumption is that the trains using the crossing behave like real world trains. In particular the following should be obeyed: all trains which enter R must leave R ; all trains which leave R must have entered R (these two requirements express the principle of Conservation of Trains); and only a finite number of trains may enter R in a finite time. The consequence of any of these assumptions being false is that there would be a loss of utility, safety or both.

The final assumption was that the crossing hardware works perfectly; for example if an *up* signal was sent to the hardware controlling the gate then it would accept that signal and start to move the gate up.

3.2 Constraints

When considering the constraints on the parameters to the problem it is useful to consider the time-line in Figure 1.

Various weak constraints on the parameters of the problem are given in [7]; these are summarised in Table 1.

The constraints presented in Table 1 may be strengthened using reasonable real-world assumptions as follows: it is assumed that useless raising and lowering of the gate, i.e. without allowing sufficient time (δ_{car}) for a car to pass through, is unacceptable. This follows the version of the

(Hatched periods denote intervals during which signals are to be sent to the gate controller)

Figure 1: Time-line for the railway crossing problem

problem presented in Appendix D of [7] and results in the constraint: *If a signal to raise the gate is sent at time t , then at no time between t and $t + \gamma_u + \delta_{car}$ is a signal to lower the gate sent.* This is called the *Common Sense Property*. It is interesting to reflect on an imaginary abstracted version of the English specification of the problem, in which the requirements remain the same but no reference is made to any real-world system. Then we would have no reason to derive the Common Sense Property; only our knowledge of real railway crossings drives us to do this.

$\epsilon_1 \leq \epsilon_2$	the time for the fastest train to reach I after entering R , ϵ_1 , is greater than or equal to the time for the slowest train to reach I
$\epsilon_1 > \gamma_d$	the time taken to close the gate, γ_d , is smaller than the time for the fastest train to reach I after entering R
$\epsilon_1 \geq \xi_1 \geq \gamma_d + \epsilon_2 - \epsilon_1$	ξ_1 must be chosen such that it is small enough to ensure that the gate is closed if a fast train has entered R , but large enough so that utility is not lost if a slow train has entered R
$\xi_2 \geq \gamma_u$	the time taken to raise the gate is small enough that the Utility Property is not violated

Table 1: Constraints on parameters

4 Model Based Specification

In this section, a ‘model-based’ approach is illustrated in the specification of the railway crossing problem. It is shown how the Z notation [15] can be used to model elegantly both the static (state and operations) and dynamic (real-time) behaviour of the system. The work is based the conventions outlined in [5] for applying Z to concurrent and real-time systems.

4.1 Approach

The classical Z style of specification provides a powerful and expressive means of describing the ‘static’ behaviour of a system. However, it is less clear how to describe the dynamic behaviour of systems in Z. This aspect is essential for the specification of concurrent and real-time systems. This limitation is overcome by adopting the following extended Z specification approach: (1) Specify the state and operations of the system in the conventional Z style (this is the *static* specification). Timing considerations are ignored at this stage. (2) Augment this with a specification of the system’s allowable real-time behaviour (this is the *dynamic* specification). Real-time behaviour is formalised in terms of an infinite sequence of states and times.

4.2 The Static Specification

The static specification of the GRC is divided into two components: *Trains*, which keeps track of the trains in R , and *Gate* which describes the state and operations of the gate mechanism.

4.2.1 The Trains' State

Consider a set of trains, *trains*, which are in the domain of interest. Each train's position is recorded as being in one of three states: *R* - in the region *R*; *I* - in the gate; *not_here* - not in either *R* or *I*:

$$[TRAIN]$$

$$POSITION ::= R \mid I \mid not_here$$

$\frac{Trains}{\begin{array}{l} trains : \mathbb{P} TRAIN \\ position : TRAIN \rightarrow POSITION \end{array}}$
$dom\ position = trains$

Initially, there are no trains in *R*:

$$TrainsInit \hat{=} [Trains \mid \forall t : trains \bullet position(t) = not_here]$$

4.2.2 The Gate's State

The state space of the gate is specified as follows:

$$GATE_POS ::= up \mid down \mid going_up \mid going_down$$

$\frac{Gate}{gate : GATE_POS}$

The gate can be *up*, *down*, *going_up*, or *going_down*.

Initially, the gate is raised:

$$GateInit \hat{=} [Gate \mid gate = up]$$

4.2.3 Operations

There are three operations used to specify the behaviour of the trains. They describe what happens when a train enters each particular region or leaves the gate:

$\frac{EnterR \quad \Delta Trains \quad t? : TRAIN}{\begin{array}{l} position(t?) = not_here \\ position' = position \oplus \{t? \mapsto R\} \end{array}}$

A train may enter *R* provided that it is not already in the region.

Once in the region, a train may enter the gate:

$\frac{EnterI \quad \Delta Trains \quad t? : TRAIN}{\begin{array}{l} position(t?) = R \\ position' = position \oplus \{t? \mapsto I\} \end{array}}$

Once in the gate, a train may leave:

<i>LeaveI</i>
$\Delta Trains$
$t? : TRAIN$
$position(t?) = I$
$position' = position \oplus \{t? \mapsto not_here\}$

The gate has four operations, which describe the lowering and raising of the gate:

<i>Lower</i>
$\Delta Gate$
$gate \in \{up, going_up\}$
$gate' = going_down$

<i>Raise</i>
$\Delta Gate$
$gate \in \{down, going_down\}$
$gate' = going_up$

<i>Down</i>
$\Delta Gate$
$gate = going_down$
$gate' = down$

<i>Up</i>
$\Delta Gate$
$gate = going_up$
$gate' = up$

This completes the ‘static’ specification of the system. However, the specification as it stands says nothing about the behaviour of the system over time. This is the objective of constructing the dynamic specification.

4.3 The Dynamic Specification

The first step in writing the dynamic specification is to construct a *next-state schema*. This is the disjunction of the GRC’s operations. It represents the fact that an atomic step in the GRC’s behaviour may be caused by any of its operations:

$$GRCNS \hat{=} \\ EnterR \vee EnterI \vee LeaveI \vee \\ Lower \vee Raise \vee Down \vee Up$$

The timed requirements of the system can now be specified. First, we introduce the global constants and their constraints:

$$\begin{array}{|l} \epsilon_1, \epsilon_2, \xi_1, \xi_2, \gamma_d, \gamma_u : \mathbb{N} \\ \hline .. constraints .. \end{array}$$

Here, the constraints on the global constants are the same as those adopted in section 3.2.

The combined system state and initial state schemas are obtained by the conjunction of the trains' and gate's state schemas and initial state schemas:

$$\begin{aligned} GRC &\triangleq Trains \wedge Gate \\ GRCInit &\triangleq TrainsInit \wedge GateInit \end{aligned}$$

The system's behaviour can now be specified. A behaviour σ (represented as an infinite sequence of GRC states and times) is valid if: (a) it is a valid computation of the system, and (b) the execution of the *EnterI*, *Down* and *Up* operations are restricted by certain lower and upper time bounds:

<i>TimedBehaviour</i>
$\sigma : seq_{\infty} (GRC \times \mathbb{N})$
$\sigma \text{ validcomp}_t (\{GRCInit \bullet \theta GRC\}, \{GRCNS \bullet \theta GRC \mapsto \theta GRC'\})$
$\forall t : TRAIN \bullet$
$\sigma \text{ bounds} (\{EnterI \bullet \theta GRC \mapsto \theta GRC'\}, \epsilon_1, \epsilon_2)$
$\sigma \text{ bounds} (\{Down \bullet \theta GRC \mapsto \theta GRC'\}, 0, \gamma_d)$
$\sigma \text{ bounds} (\{Up \bullet \theta GRC \mapsto \theta GRC'\}, 0, \gamma_u)$

Briefly, the relation validcomp_t is true for all behaviours in which the first step in the sequence (σ) at time 0 belongs to the set of initial states and each subsequent state is related to the previous one by the next-state schema or the ticking of the clock. An operation in the behaviour is bounded with lower or upper time limits l and u if once enabled at t it executes within $t + l$ and $t + u$ or it is disabled^{1,2}.

Here the bounds of *EnterI* are the earliest and latest times that a train can enter the gate after entering *R*. The bounds of *Down* and *Up* are the upper bounds on the time required to lower and raise the gate. Note, the bounds of *EnterI* have been quantified over all trains. In effect this associates an implicit timer with each train, ensuring that they will all meet their deadlines for entering *I*.

As an example, a possible behaviour of the gate might be:

<i>gate</i>	<i>trains</i>	<i>time</i>
<i>up</i>	TrainsInit	0
<i>up</i>	?	1
<i>going_down</i>	?	2
<i>going_down</i>	?	2+
<i>down</i>	?	$\leq \gamma_d + 2$

..

In this behaviour the gate first starts going down at time $t = 2$; therefore the gate must be lowered by the time $\gamma_d + 2$ (the bound is asserted from the first point in time that the operation *Down* is enabled).

4.4 Adding the Safety and Utility Properties

Unfortunately, the above specification as it stands does not guarantee the required Safety and Utility Properties. This is because no constraints have been specified on the interaction between the trains and gate. Therefore, the specification must be strengthened to rule out invalid behaviours.

¹Full definitions of the validcomp_t and bounds relations can be found in [4].

²Note the use of schema binding θ to project out the required state components of the initial and next-state schemas for substitution into validcomp_t and bounds .

4.4.1 Safety Property

In order ensure the Safety Property is preserved the following invariant is added to the system state:

$$(\exists t : \text{trains} \bullet \text{position}(t) = I) \Rightarrow \text{gate} = \text{down}$$

The gate is always down when there are trains in the gate.

4.4.2 Utility Property

The time bound restriction expressed by the Utility Property is a more complicated temporal property. Additional operations are therefore added to strengthen the specification with the additional properties. There are two cases to be considered:

1. *The time from when the gate starts going down until some train enters I is bounded by ξ_1 .* This behaviour can be specified by a new operation *EnterIU*:

$\frac{\text{EnterIU} \quad \text{EnterI} \quad \Delta \text{Gate}}{\text{gate} = \text{going_down}}$
--

A new time bound must also be added to assert that the operation must occur within ξ_1 seconds.

2. *From when the crossing becomes empty, either the time until the gate is up is bounded by ξ_2 or else the time until a train is in I is bounded by $\xi_1 + \delta_{car} + \xi_2$.*

Again, this can be specified by the addition of further operations to the system:

The first operation includes the components of *Up* but also has the additional pre-condition that the crossing is empty. It has the upper time bound of ξ_2 :

$\frac{\text{UpU} \quad \text{Up} \quad \Xi \text{Trains}}{\forall t : \text{TRAIN} \bullet \text{position}(t) \neq I}$

The second operation includes the components of *EnterI* but also has the additional pre-condition that the crossing is empty. It has the upper time bound $\xi_1 + \delta_{car} + \xi_2$:

$\frac{\text{EnterIU2} \quad \text{EnterI}}{\forall t : \text{TRAIN} \bullet \text{position}(t) \neq I}$
--

The requirement that either time bound must hold is specified as the disjunction of the two possible time bounds:

$$(\sigma \text{ bounds } (\{ \text{UpU} \bullet \theta \text{GRC} \mapsto \theta \text{GRC}' \}, 0, \xi_2) \vee \sigma \text{ bounds } (\{ \text{EnterIU2} \bullet \theta \text{GRC} \mapsto \theta \text{GRC}' \}, 0, \xi_1 + \delta_{car} + \xi_2))$$

4.5 Discussion

This section has illustrated an alternative model based approach to the specification of the generalised rail-road crossing problem using Z. The specification that resulted was similar to that obtained by Heitmeyer and Lynch using timed automata. However, rather than incorporating time within the state itself, we have incorporated it within an additional specification of the system's behaviour. This has the great advantage of providing a separation of concern between static

and timed behaviour, which is missing from Heitmeyer's and Lynch's specification. This approach also resulted in much simpler specifications. The main disadvantage of the approach however is in the verification of the system's properties. Because Heitmeyer and Lynch treat time as just another state variable, they are able to use traditional assertional proof techniques to verify required properties with little difficulty. In the approach outlined above, it will be necessary to develop new proof techniques for inferring properties of timed computations from operations and their bounds. This area of work is currently under investigation.

5 Process Algebra Specifications

5.1 Timed CSP

Timed CSP is an extension of CSP. The first model for Timed CSP was proposed by Jones [9] which proved unsatisfactory for a number of technical reasons. It was suggested that a better model could be obtained by recording the events refused *during* the observation of a trace; this is a feature of the later and more successful model proposed by Reed and Roscoe [13]. Then, Davies and Schneider in [2, 14] extended Reed and Roscoe's model to include specifications and a proof system.

In this section we use Timed CSP to describe the railroad crossing problem. We prove that the Timed CSP implementation satisfies the Safety and Utility Properties required, but do not consider the Common Sense Property. The application of the proof system for Timed CSP is simple and efficient.

To introduce timing information into CSP, several assumptions are required:

Real Time. The non-negative real numbers are used for the time domain.

Global Clock. All observations are recorded with reference to an imaginary global clock.

Instantaneous Events. All events have zero duration.

Finite Speed. No process can engage in infinitely many events within a finite time interval.

Hiding and Control. Observable events cannot occur without the cooperation of the environment. Hidden events do not require the cooperation of the environment, and occur as soon as they become available.

Delay Constant. To preserve causality, a positive delay constant δ is chosen as a lower bound between consecutive events in a sequential process. This ensures that if the occurrence of event a makes another event b possible, then b cannot occur at the same time as a . For simplicity, when the inference rules for Timed CSP are applied in the following, the constant time delay δ is omitted, assuming that it is too small to affect our processes.

5.1.1 The Timed CSP Specification of the Railroad Crossing

The main process for the railroad crossing in Timed CSP consists of two component processes, *Counter* and *Gate*, which are composed in parallel and communicate via the hidden actions k and m :

$$Main \stackrel{\text{def}}{=} (Counter \parallel \{k, m\} \parallel Gate) \setminus \{k, m\}$$

The process *Counter* monitors the arrival and departure of trains and records the number of trains in R . It also sends signals k , representing the arrival of a train when R is empty, and m , representing the departure of the last train in R , to process *Gate* which closes (*down*) and opens (*up*) the gate accordingly.

$$\begin{aligned} Counter &\stackrel{\text{def}}{=} a \rightarrow k \rightarrow Counter_1 \\ Counter_i &\stackrel{\text{def}}{=} a \rightarrow Counter_{i+1} \end{aligned}$$

$$\begin{aligned} \Box l \rightarrow Counter_{i-1} & \quad \text{for } i \geq 1 \\ Counter_0 & \stackrel{\text{def}}{=} m \rightarrow Counter. \end{aligned}$$

Counter counts the number of trains arriving at *R*, signified by the *a* action, and the number of departures, signified by *l*. When a train arrives and *R* is empty, *Counter* sends a signal *k* to *Gate* and counts this train by moving to *Counter*₁, indicating that there is one train in *R*. After the arrival of the first train, other trains can come and go in any order. This is modelled by an internal choice between *a* and *l* in *Counter*_{*i*}, for *i* ≥ 1, indicating that the arrival and departure of trains are not controlled by its environment. The maximum number of trains in *R* at any one time is unbounded. When the last train in *R* has departed, it is safe to open the gate. *Counter*₀ sends a signal *m*, to *Gate* and then moves to *Counter*. Let

$$\begin{aligned} t_s & \stackrel{\text{def}}{=} \xi_1 - \epsilon_2 + \epsilon_1 - \gamma_d && \text{distance between } U_1 \text{ and } S_1 \\ t_o & \stackrel{\text{def}}{=} \min\{t_{u2} - t_3, \epsilon_2 - \xi_1\} && \text{maximum delay before sending an } up \\ t_{u1} & \stackrel{\text{def}}{=} \epsilon_2 - \xi_1 && \text{the first utility point} \\ t_{u2} & \stackrel{\text{def}}{=} \xi_2 - \gamma_u && \text{the second utility point} \end{aligned}$$

We have

$$\begin{aligned} Gate & \stackrel{\text{def}}{=} k \xrightarrow{t_{u1}} down@t_1\{t_s\} \rightarrow P_1 \\ P_1 & \stackrel{\text{def}}{=} m \rightarrow (up@t_2\{t_{u2}\} \rightarrow Gate \\ & \quad \Box k@t_3 \rightarrow up@t_4\{t_o\} \xrightarrow{t_{u1}-t_4} down@t_5\{t_s\} \rightarrow P_1). \end{aligned}$$

Process *Gate* is composed in parallel with *Counter*, synchronising on $\{k, m\}$. When the *k* signal is received, *Gate* sends a *down* signal to the gate after *t_{u1}* time units and within $\epsilon_1 - \gamma_d$ time units, which means that the gate will be fully down before the train enters the crossing; the gate will remain down until the last train leaves *R* (*m* is received), when the signal to raise the gate will be sent before the second utility point if this does not violate the Safety Property.

Note: There are a number of points needing discussion.

- N1. Why do we need to wait for $t_{u1} = (\epsilon_2 - \xi_1)$ time units instead of $(\epsilon_1 - \xi_1)$ to send the *down* signal to preserve the Utility Property?

Suppose that a slow train enters region *R* at time *t₁*. It takes ϵ_2 time units to reach the crossing *I*. The utility area, i.e. the interval during which the gate *must* be open, should be $[t_1, t_1 + \epsilon_2 - \xi_1]$. If we wait for only $(\epsilon_1 - \xi_1)$ time units, then we cannot guarantee the Utility Property that the gate should be up before the first utility point $(t_1 + \epsilon_2 - \xi_1)$ for this train because $\epsilon_1 \leq \epsilon_2$.

- N2. Then, is it safe to wait for $(\epsilon_2 - \xi_1)$ time units before sending the *down* signal if the train is the fastest train which takes ϵ_1 time units to reach the crossing?

If the train is the fastest train, we need to make sure that before it reaches the crossing, the gate should be down already. That is,

$$\epsilon_1 - \gamma_d \geq \epsilon_2 - \xi_1 \equiv \xi_1 \geq \epsilon_2 - \epsilon_1 + \gamma_d,$$

which is our assumption.

- N3. The calculation for the upper bound *t_s* of time before sending a *down* signal is as follows.

$$\underbrace{(\epsilon_2 - \xi_1)}_{\text{utility area}} + \underbrace{(\xi_1 - (\epsilon_2 - \epsilon_1 + \gamma_d))}_{t_s} = \underbrace{\epsilon_1 - \gamma_d}_{\text{earliest safety point}}$$

To prove the Safety and Utility Properties, we need the following lemmas.

Lemma 1 *Counter monitors the number of trains in R correctly and sends a signal k when the first train enters R and a signal m when the last train leaves R . That is, if*

$$s \downarrow a - s \downarrow l = i,$$

then the control of Counter is at the beginning of Counter _{i} ; furthermore, if $\text{last}(s) = a$, the control is transferred from Counter _{$i-1$} ; if $\text{last}(s) = l$, the control is transferred from Counter _{$i+1$} .

Proof: (By induction on the length of the trace.) As this property is independent of timing consideration, we can use the untimed trace proof system. See Appendix A. $\square$

A number of predicates for structured specifications are defined as follows.

$$a \text{ from } t \text{ until } t' (s, X) \stackrel{\text{def}}{=} a \notin \sigma(X \uparrow [t, \min\{t', \text{begin}(s \uparrow [t, \infty) \downarrow a\})).$$

If the offer of a has not been accepted by time t' , the process may retract without violating the liveness specification. If a process satisfies this specification, event a must become available at time t , and must remain available until either time t' or the time at which the next a is observed, whichever is smaller.

$$a \text{ at } I (s, X) \stackrel{\text{def}}{=} \exists t : I \bullet \langle (t, a) \rangle \text{ in } s.$$

Event a must be observed at some time during time interval I . Then, we can define the liveness property of event a during time interval I as follows:

$$a \text{ live } [t, t'] (s, X) \stackrel{\text{def}}{=} a \text{ from } t \text{ until } t' \vee a \text{ at } [t, t'],$$

which means that either event a is available during time interval $[t, t']$, or it occurs during it. We also define a predicate

$$a \text{ precedes } (t, b) (s, X) \stackrel{\text{def}}{=} \text{last}(s \uparrow [0, t)) = a,$$

which means that a immediately precedes b at time t .

Lemma 2 *When the first train enters R , the gate will be lowered before the fastest train reaches the crossing and will be kept open at least until it reaches the first utility point, provided that the environment cooperates, accepting the up signal from P_1 in time. That is,*

$$\begin{aligned} \text{Gate } \text{sat} \\ k \text{ at } t \wedge \neg(m \text{ precedes } (t, k)) \Rightarrow \\ \text{down } \text{live} [t + t_{u1}, t + \epsilon_1 - \gamma_d] \wedge \neg(\text{down } \text{at} [t, t + t_{u1})) \\ \wedge k \text{ at } t \wedge m \text{ precedes } (t, k) \wedge \text{up } \text{at} (t + t_4) \wedge t_4 \leq t_o \Rightarrow \\ \text{down } \text{live} [t + t_{u1}, t + \epsilon_1 - \gamma_d] \wedge \neg(\text{down } \text{at} [t, t + t_{u1})) \end{aligned} \quad (1)$$

Proof: The proof of this lemma uses the Timed CSP proof system. In the following, $S_P(s, X)$ represents the strongest specification satisfied by process P .

We begin by observing that the bodies of the mutual recursions are constructive, providing that $t_{u1} > 0$. We may then apply the inference rule for mutual recursion, reducing our proof obligation to

$$k \xrightarrow{t_{u1}} \text{down} @ t_1 \{t_s\} \rightarrow X_2 \text{ sat}_\rho \quad (1) \quad (2)$$

and

$$\begin{aligned} m \rightarrow (\text{up} @ t_2 \{t_{u2}\} \rightarrow X_1 \\ \square k @ t_3 \rightarrow \text{up} @ t_4 \{t_o\} \xrightarrow{t_{u1} - t_4} \text{down} @ t_5 \{t_s\} \rightarrow X_2) \text{ sat}_\rho \quad (1) \end{aligned} \quad (3)$$

under the assumption that X_1 and X_2 both satisfy (1).

It is easy to prove by induction that the k in (2) is not immediately preceded by an m , but the k in (3) is.

Then, to prove (2), we apply the inference rules for event prefix and bounded timer operator in two steps and have

$$\begin{aligned}
 \text{(a)} \quad & k \xrightarrow{t_{u1}} P_2 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge k \notin \sigma(X) \\
 & \vee s = \langle (t, k) \rangle \wedge s_1 \wedge k \notin \sigma(X \upharpoonright t) \wedge \text{begin}(s_1) \geq t + t_{u1} \\
 & \quad \wedge S_{P_2}((s_1, X) - (t + t_{u1})) \\
 \text{(b)} \quad & \text{down}@t_1\{t_s\} \rightarrow X_2 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge \text{down} \notin \sigma(X \upharpoonright t_s) \\
 & \vee s = \langle (t_1, \text{down}) \rangle \wedge s_2 \wedge \text{down} \notin \sigma(X \upharpoonright t_1) \wedge \text{begin}(s_2) \geq t_1 \\
 & \quad \wedge t_1 \leq t_s \wedge S_{X_2}((s_2, X) - t_1).
 \end{aligned}$$

Putting (a) and (b) together by another application of the rule for event prefix, we can deduce that

$$\begin{aligned}
 & k \xrightarrow{t_{u1}} \text{down}@t_1\{t_s\} \rightarrow X_2 \\
 \text{sat}_\rho \quad & k \left\{ \begin{array}{l} s = \langle \rangle \wedge k \notin \sigma(X) \\ \vee s = \langle (t, k) \rangle \wedge k \notin \sigma(X \upharpoonright t) \wedge \text{down} \notin \sigma(X \upharpoonright [t + t_{u1}, t + t_{u1} + t_s]) \end{array} \right. \\
 \text{down} \quad & \left\{ \begin{array}{l} \vee s = \langle (t, k), (t + t_{u1} + t_1, \text{down}) \rangle \wedge s_2 \wedge k \notin \sigma(X \upharpoonright t) \\ \quad \wedge \text{down} \notin \sigma(X \upharpoonright [t + t_{u1}, t + t_{u1} + t_1]) \\ \quad \wedge \text{begin}(s_2) \geq t + t_{u1} + t_1 \wedge t_1 \leq t_s \\ \quad \wedge S_{X_2}((s_2, X) - (t + t_{u1} + t_1)). \end{array} \right. \\
 \implies & \{ \text{under the assumptions and that } k \text{ is not immediately preceded by } m \} \\
 & (1).
 \end{aligned}$$

To prove (3), we only need to consider the k branch of the process (for a complete treatment of deterministic choice, see Lemma 3). By applying the inference rules for event prefix and bounded timer, we have

$$\begin{aligned}
 \text{(c)} \quad & m \rightarrow P_3 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge m \notin \sigma(X) \\
 & \vee s = \langle (t, m) \rangle \wedge s_3 \wedge m \notin \sigma(X \upharpoonright t) \wedge \text{begin}(s_3) \geq t \\
 & \quad \wedge S_{P_3}((s_3, X) - t) \\
 \text{(d)} \quad & k@t_3 \rightarrow P_4 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge k \notin \sigma(X) \\
 & \vee s = \langle (t_3, k) \rangle \wedge s_3 \wedge k \notin \sigma(X \upharpoonright t_3) \wedge \text{begin}(s_3) \geq t_3 \\
 & \quad \wedge S_{P_4}((s_3, X) - t_3) \\
 \text{(e)} \quad & \text{up}@t_4\{t_o\} \xrightarrow{t_{u1}-t_4} P_5 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge \text{up} \notin \sigma(X \upharpoonright t_o) \\
 & \vee s = \langle (t_4, \text{up}) \rangle \wedge s_4 \wedge \text{up} \notin \sigma(X \upharpoonright t_4) \wedge \text{begin}(s_4) \geq t_{u1} \\
 & \quad \wedge t_4 \leq t_o \wedge S_{P_5}((s_4, X) - t_{u1}) \\
 \text{(f)} \quad & \text{down}@t_5\{t_s\} \rightarrow X_2 \\
 \text{sat}_\rho \quad & s = \langle \rangle \wedge \text{down} \notin \sigma(X \upharpoonright t_s) \\
 & \vee s = \langle (t_5, \text{down}) \rangle \wedge s_5 \wedge \text{down} \notin \sigma(X \upharpoonright t_5) \wedge \text{begin}(s_5) \geq t_5 \\
 & \quad \wedge t_5 \leq t_s \wedge S_{X_2}((s_5, X) - t_5).
 \end{aligned}$$

Putting (c), (d), (e), and (f) together, we have

$$\begin{aligned}
& m \rightarrow k @t_3 \rightarrow up @t_4 \{t_o\} \xrightarrow{t_{u1}-t_4} down @t_5 \{t_s\} \rightarrow X_2 \\
\text{sat}_\rho & \\
m & \left\{ \begin{array}{l} s = \langle \rangle \wedge m \notin \sigma(X) \\ \vee s = \langle (t, m) \rangle \wedge m \notin \sigma(X \upharpoonright t) \wedge k \notin \sigma(X \upharpoonright t) \end{array} \right. \\
k & \left\{ \begin{array}{l} \vee s = \langle (t, m), (t + t_3, k) \rangle \wedge m \notin \sigma(X \upharpoonright [t, t + t_3]) \wedge k \notin \sigma(X \upharpoonright t_3) \\ \wedge up \notin \sigma(X \upharpoonright [t + t_3, t + t_3 + t_o]) \end{array} \right. \\
up & \left\{ \begin{array}{l} \vee s = \langle (t, m), (t + t_3, k), (t + t_3 + t_4, up) \rangle \wedge m \notin \sigma(X \upharpoonright t) \wedge k \notin \sigma(X \upharpoonright [t, t + t_3]) \\ \wedge up \notin \sigma(X \upharpoonright [t + t_3, t + t_3 + t_4]) \wedge t_4 \leq t_o \\ \wedge down \notin \sigma(X \upharpoonright [t + t_3 + t_{u1}, t + t_3 + t_{u1} + t_s]) \end{array} \right. \\
down & \left\{ \begin{array}{l} \vee s = \langle (t, m), (t + t_3, k), (t + t_3 + t_4, up), (t + t_3 + t_{u1} + t_5, down) \rangle \wedge s_6 \\ \wedge m \notin \sigma(X \upharpoonright t) \wedge k \notin \sigma(X \upharpoonright [t, t + t_3]) \wedge up \notin \sigma(X \upharpoonright [t + t_3, t + t_3 + t_4]) \wedge t_4 \leq t_o \\ \wedge down \notin \sigma(X \upharpoonright [t + t_3 + t_{u1}, t + t_3 + t_{u1} + t_5]) \\ \wedge t_5 \leq t_s \wedge \text{begin}(s_6) \geq t + t_3 + t_{u1} + t_5 \\ \wedge S_{X_2}((s_6, X) - (t + t_3 + t_{u1} + t_5)) \end{array} \right. \\
\Rightarrow & \{ \text{under the assumptions that } k \text{ is immediately preceded by } m \} \\
& (1)[t_3/t].
\end{aligned}$$

This completes the proof. $\square$

Lemma 3 *When the last train in R leaves the crossing, the gate will be fully up before it passes the second utility point.*

$$\text{Gate sat } m \text{ at } t \Rightarrow up \text{ live } [t, t + t_{u2}].$$

Proof: The proof of this lemma is similar to that of Lemma 2 and is omitted to save space.

As already mentioned in Lemma 2 and Lemma 3, it can only be ascertained that the CSP processes will be ready to send the proper signals at the required times to achieve the Safety and Utility Properties. However we cannot guarantee the Safety and Utility Properties if the environment does not cooperate. For example, if the gate is jammed while up, the Safety Property will be violated even if the CSP processes behave properly. Therefore, in the proof of the following theorems, the assumptions about the environment found in Lemma 2 and Lemma 3 are made.

Theorem 1 (The Safety Property) *Main satisfies the Safety Property provided that the environment cooperates as expected, that is,*

$$t \in \bigcup_i [\tau_i, \nu_i] \Rightarrow g(t) = 0,$$

where τ_i is the time when the i th train enters crossing I and ν_i is the time when the i th train leaves the crossing.

Proof: By induction on the number of trains which have passed R .

Base: The initial state of the system is: No train has entered R yet ($i = 0$) and the gate is up. So the theorem holds.

Induction hypothesis: Assume the theorem holds for the first n trains, that is, for $n \geq 0$,

$$t \in \bigcup_{i=0}^n [\tau_i, \nu_i] \Rightarrow g(t) = 0.$$

Induction step: The next train enters R . There are two cases:

- (a) It is the first train entering an empty R region at time t_{n+1} after the last train in R has left the crossing, i.e., $t_{n+1} > \nu_{last}$. By Lemma 1, the control of *Counter* is at the beginning of *Counter*₁ and (t_{n+1}, k) *in* s . By Lemma 2, *down* will be sent before $(t_{n+1} + \epsilon_1 - \gamma_d)$, i.e., the gate will be fully closed before $(t_{n+1} + \epsilon_1) \leq t_{n+1} + \epsilon_{n+1} = \tau_{n+1}$. We have $g(t_{n+1} + \epsilon_1) = 0$. By Lemma 1, we have that m will be sent only after this train has left the crossing, which means that the gate will remain closed during $[\tau_{n+1}, \nu_{n+1}] (\subseteq [t_{n+1} + \epsilon_1, \nu_{n+1}])$, as required.

- (b) The next train enters R before it becomes empty. Then, by the induction hypothesis and our assumption, the gate will be lowered before the first train in R reaches the crossing and remain closed at least until the last train (including this one) leaves R .

□

Theorem 2 (The Utility Property) *Main satisfies the Utility Property, that is,*

$$t \notin \bigcup_i [\tau_i - \xi_1, \nu_i + \xi_2] \Rightarrow g(t) = 90.$$

Proof: By induction on the number of trains which have passed R . The proof method is similar to the preceding, so is omitted to save space. □

5.1.2 Conclusions

Timed CSP is an elegant formal method for real-time systems. The language is concise and expressive; the proof system is easy to use.

One addition to the original Timed CSP is the bounded timer, described in Davies [3]. This operator allows us to model the railway crossing system in such a way that once the *up* signal is sent and the gate is going up, there should be enough time for the gate to be fully raised and some cars to pass through the crossing. We cannot enforce this without a timer with an upper bound, $a@t\{d\}$. It can also simplify process *Gate* in that we could put an upper time bound for sending an *up* or *down* signal by using a timer instead of a time-out operator.

5.2 Timed CCS

Timed CCS [1] is an extension of CCS [12]. The time domain is assumed to have a total ordering and unique top and bottom elements, allowing time to be either discrete or dense. The only change Timed CCS makes to the syntax of CCS is in the definition of the prefix operator. The expression $\alpha(t)_e^{e'}.E$ means that the action α takes place at some time in the closed interval $[e, e']$ then behaves as E , where only e' is allowed to be ∞ . The time variable t takes on the value of the time at which action α takes place, and E proceeds with all free occurrences of t in E substituted for this time, allowing time dependencies between actions to be described. For example $\alpha(t)_e^{e'}. \beta(s)_{t+1}^{t+3}.E$ denotes that β occurs between 1 and 3 time units after α . For convenience $\alpha(t)_0^\infty.E$ is written as $\underline{\alpha}.E$. A derived operator, *time prefix*, has been used below, where the expression $(e)E$ represents a process which behaves as E after a delay of time e .

5.2.1 The Timed CCS Specification of the Railway Crossing

If the Common Sense Property is to be satisfied then further constraints can be derived. In particular, if the gate is ever to be raised after the first train has passed through, as implied by the Utility Property, then the Safety and Common Sense Properties together imply that $\epsilon_1 \geq \gamma_u + \delta_{car} + \gamma_d$. The consequence of this is that if a train enters R after the last train has left and the gate is still down then there is enough time to open the gate without violating either the Common Sense Property or the Safety Property. This has lead to a more complicated specification with a number of branching points, where each branch corresponds to the amount of time between trains arriving at and departing from R .

As in the Timed CSP specification, the main process consists of the parallel composition of two processes:

$$Main \stackrel{def}{=} (Counter \mid Gate) \setminus \{k, m\}$$

The *Counter* has the same purpose and behaviour as that in section 5.1.1:

$$\begin{aligned} Counter &\stackrel{def}{=} \underline{a.k}.Counter_1 \\ Counter_1 &\stackrel{def}{=} \underline{a}.Counter_2 \\ &\quad + \underline{l.m}.Counter \\ Counter_i &\stackrel{def}{=} \underline{a}.Counter_{i+1} \quad i > 1 \\ &\quad + \underline{l}.Counter_{i-1} \end{aligned}$$

Introducing the Common Sense Property has complicated the definition of the *Gate* process:

$$\begin{aligned}
 \text{Gate} &\stackrel{def}{=} \overline{k}.down(s)^{t_1}.Gate' \\
 \text{Gate}' &\stackrel{def}{=} \overline{m}.(up(s)^{t_3}.((\gamma_u + \delta_{car})Gate + \overline{k}(r)^{t_4}.down(p)^{t_5}.Gate') \\
 &\quad + \overline{k}(s')^{t_3}.up(r')^{t_7}.down(p')^{t_8}.Gate')
 \end{aligned}$$

This is best explained by reference to Figure 2. The remarkable increase in complexity of the *Gate* process by comparison with the analogous Timed CSP process is due solely to the strengthening of the requirements to account for the Common Sense Property. This allows three forms of behaviour in the case where the only train in *R* leaves that region:

1. The gate is fully raised and there is enough time (δ_{car}) for a car to cross the crossing before the next train enters *R*. In this case control follows the branch labelled 1 in Figure 2.
2. The next train enters *R* after the signal to raise the gate has been sent, but before a car has had time to cross the crossing. In this case control follows branch 2.
3. The next train enters *R* before the signal to raise the gate has been sent. In this case control follows branch 3. It is this behaviour that is not permitted in the Timed CSP specification.

Figure 2: The state space of *Graph*

The values of the various time expressions, $t_1 \dots t_9$, are given below along with a discussion of their significance.

$t_1 = t + \epsilon_2 - \xi_1$ the signal to close the gate must not be sent before the utility point

$t_2 = t + \epsilon_1 - \gamma_d$ the signal to close the gate must be sent no later then the safety point

$t_3 = t + \xi_2 - \gamma_u$ ensure that the signal to open the gate is sent before the second utility point

$t_4 = s + \gamma_u + \delta_{car}$ if a $\overline{k}$ is received by this time then a new train has arrived before a car has gone through the crossing

$t_5 = r + \max(\gamma_u + \delta_{car} - r + s, \epsilon_2 - \xi_1)$ if a new train has arrived then wait at least long enough for a car to get through or until the first utility point, whichever is later

$t_6 = r + \epsilon_1 - \gamma_d$ analogous to t_2 , but for a different train

$t_7 = s' + \min(\xi_2 - \gamma_u - s' + t, \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d)$ if a new train arrives before the up signal has been sent then send it at least early enough for a car to get through, but if the second utility point is earlier then send it at that time

$t_8 = r' + \max(\gamma_u + \delta_{car}, \epsilon_2 - \xi_1 - r' + s')$ the earliest time to send a signal to close the gate is after a car has got through or the first utility point, whichever is later

$t_9 = s' + \epsilon_1 - \gamma_d$ analogous to t_2

5.2.2 Proofs of Properties

Timed CCS has a proof system for strong equivalence between processes, which is useful for refinement, but has no system for demonstrating that a process has particular properties. An extension of Hennessey-Milner logic is discussed in [1], but is not developed sufficiently to be useful. Therefore an analysis of the various pathways through the state space of *Gate* was used to establish the required properties of Safety, Utility and Common Sense. The properties required by *Counter* have been established in the section on Timed CSP, so their proof will not be repeated here.

Theorem 3 *The specification satisfies the Safety Property.*

Proof 1 This is established by demonstrating that the upper bound on the time taken to send a *down* signal after a *k* has been received guarantees safety. The analysis for branch 2 of Figure 2 is given below; other cases are similar.

Branch 2 corresponds to the sequence of actions $\bar{k}(r)_s^{t_4}.down(p)_{t_5}^{t_6}.Gate'$. In order to prove safety it is necessary to show that the upper bound on *down* is safe, and that the upper bound is greater than the lower bound. Since $t_6 = r + \epsilon_1 - \gamma_d$, where r is the time of the last *k* event, the upper bound is clearly safe. The second requirement is discharged by demonstrating that

$$r + \epsilon_1 - \gamma_d \geq r + \max(\gamma_u + \delta_{car} - r + s, \epsilon_2 - \xi_1)$$

which requires two cases:

Case 1:

$$\begin{aligned} r + \epsilon_1 - \gamma_d &\geq r + \epsilon_2 - \xi_1 \\ \epsilon_1 - \gamma_d &\geq \epsilon_2 - \xi_1 \end{aligned}$$

This is one of the initial assumptions.

Case 2:

$$\begin{aligned} r + \epsilon_1 - \gamma_d &\geq s + \gamma_u + \delta_{car} \\ \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d &\geq s - r \end{aligned}$$

From the initial assumptions, the left hand side of this inequality is ≥ 0 , while it is known that time s is before r , therefore the right hand side is ≤ 0 , concluding the proof. $\square$

Both the Utility and the Common Sense Properties can be proven in a similar way to this. The details are omitted to save space.

5.3 TE-LOTOS

The process algebra TE-LOTOS (Time Extended LOTOS) is a timed extension to the ISO standard specification language LOTOS. It is a proper superset of LOTOS, and so allows all the great expressiveness of that language, with enabling, hiding, interrupts, guards, etc. Certain LOTOS operators are given new features in TE-LOTOS, notably (timed) transitions, in which events may be specified to occur between given time bounds.

No proof system has yet been associated with TE-LOTOS, so it should be regarded as a formal notation, rather than as a formal method. The proof that the TE-LOTOS specification has the desired properties is therefore rather *ad hoc* by comparison with the CSP proof in Section 5.1.

5.3.1 The TE-LOTOS Specification of the Railway Crossing

To aid comparison the structure and most of the event names of the Timed CSP specification have been retained.

$$\begin{aligned}
Counter &::= (let\ x = 0\ in\ C) \mid [a, l] \mid Sensor \\
C &::= [x = 0] -> First \\
&\quad + [x = 1] -> (Another + Last) \\
&\quad + [x > 1] -> (Another + OneFewer) \\
First &::= a; k; exit(x + 1) \gg Loop \\
Another &::= a; exit(x + 1) \gg Loop \\
OneFewer &::= l; exit(x - 1) \gg Loop \\
Last &::= l; m; exit(x - 1) \gg Loop \\
\\
Loop &::= accept\ x\ in\ C \\
\\
Gate &::= k; P_1 \\
P_1 &::= Wait(\epsilon_2 - \xi_1); P_2 \\
P_2 &::= down\{t' in\ 0 \dots \epsilon_1 - \epsilon_2 + \xi_1 - \gamma_d\}; P_3 \\
P_3 &::= m; P_4 \\
P_4 &::= (up\{t'' in\ 0 \dots \xi_2 - \gamma_u\}; exit \gg P_5) [> (k\{s in\ 0 \dots \xi_2 - \gamma_u\}; Q_1) \\
P_5 &::= (Wait(\gamma_u + \delta_{car}); exit \gg Gate) [> (k\{t^{(5)} in\ 0 \dots \gamma_u + \delta_{car}\}; R_1) \\
\\
Q_1 &::= up\{t''' in\ 0 \dots \min\{\xi_2 - \gamma_u - s, \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d\}\}; Q_2 \\
Q_2 &::= Wait(\max\{\gamma_u + \delta_{car}, \epsilon_2 - \xi_1 - t'''\}); Q_3 \\
Q_3 &::= down\{t^{(4)} in\ 0 \dots \min\{\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1\}\}; P_3 \\
\\
R_1 &::= Wait(\max\{\gamma_u + \delta_{car} - t^{(5)}, \epsilon_2 - \xi_1\}); R_2 \\
R_2 &::= down\{t^{(6)} in\ 0 \dots \min\{\epsilon_1 - \gamma_d - \delta_{car} - \gamma_u, \epsilon_1 - \epsilon_2 + \xi_1 - \gamma_d\}\}; P_3 \\
\\
Main &::= hide\ \{k, m\}\ in\ (Gate \mid [k, m] \mid Counter)
\end{aligned}$$

The syntax of TE-LOTOS is reasonably intuitive. Here $\gg$ and $[>$ are the enable and interrupt operators respectively. Full details can be found in [11].

The *hide* operator is used to ensure that events k and m occur instantly when processes *Gate* and P_3 (respectively) have control. TE-LOTOS has a Maximal Progress Assumption that ensures this.

Wherever a *min* or *max* occurs, it means that there are two properties which must both be satisfied, and we have no information from the English specification which is the stronger property.

5.3.2 Discussion of Proofs

The proof style we shall adopt is a state-based analysis of the time bounds existing between moments when control is passed from one sub-process to another. We observe that two of the sub-processes, namely *Gate* and P_3 , act as ‘time cancellers’ in that after one of these sub-processes has taken control (say at time t) then no further references are made to times $s < t$. It is no coincidence that the time cancelling processes feature an instantaneous event (k or m) whose exact time of execution is known to the process, through the *Counter*, nor that these processes relinquish control at moments crucial to the Safety, Utility and Common Sense Properties.

For clarity we have split the flow of control in the diagram into four streams, one beginning at *Gate* and three at P_3 . Which of the streams is followed after P_3 is uniquely determined by the timing of the next k event. The time bounds given on an arc between two processes A and B represent the earliest and latest times *relative to the time when the last time cancelling process relinquished control* that control can be passed from A to B . Some of these bounds are necessarily complex, but the work done in constructing the four fragments of the flow-of-control graph is justified when we come to prove the Safety, Utility and Common Sense Properties: we find that the worst case timing can simply be read off the graph and the properties are easily proved.

Figure 3: Time bounds between processes

5.3.3 The Properties

Various generic properties of the specification must be checked: the specification is a legal piece of TE-LOTOS; all upper bounds are greater or equal to zero, etc. We will not give these proofs here.

In order to simplify the proof, we observe that the *up* and *down* events are driven by the *m* and *k* events respectively, and that in any trace:

1. if we consider only the *up* and *down* events, we find that they alternate, beginning with *down*;
2. if we consider only the *k* and *m* events, we find that they alternate, beginning with *k*.
3. $|down| \leq |k| \leq |down| + 1$
4. $|up| \leq |m| \leq |up| + 1$

where $|x|$ is the number of occurrences of event *x* in the trace.

This means that in order to show the desired properties, we need only consider situations in which an *up* or *down* event is ‘triggered’ by a *m* or *k* event respectively. There are no extraneous occurrences of *up* or *down*.

It has already been shown in the proof of correctness of the Timed CSP specification that the Counter process functions correctly. We will not duplicate the proof here.

Theorem 4 *The specification satisfies the Safety Property.*

Proof 2 We are required to show that in a state in which a *k* event occurs, i.e. a train enters an otherwise empty *R*, the greatest upper bound until the gate is fully down is no greater than ϵ_1 , the shortest time in which the train can reach *I*.

We will demonstrate the proof in the most complex case, when *P3*, *P4*, *Q1*, *Q2*, *Q3*, *P3* successively have control. We require only to show that the upper bound on the time that *Q3* passes control to *P3* - i.e. the event *down* occurs - is no greater than $\epsilon_1 - \gamma_d$, and then (under the assumption that the physical gate functions properly) we have $g(s) = 90$ (the gate is fully down) if *s* is the moment the next train reaches *I*.

The proof proceeds by cases.

Case 1: $\xi_2 - \gamma_u - s \leq \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d$. Then $t''' = \xi_2 - \gamma_u - s$.

Subcase 1a: $\gamma_u + \delta_{car} \geq \epsilon_2 - \xi_1 - t'''$.

Now

$$\min\{\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1\} = \epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d$$

Now by adding the upper time bounds of *Q1*, *Q2* and *Q3* we find that the maximum amount of time that can pass between a *k* event and a *down* event is

$$\begin{aligned}
& (\xi_2 - \gamma_u - s) + (\gamma_u + \delta_{car}) + \min\{\epsilon_1 - (\xi_2 - \gamma_u - s) - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1\} \\
& \leq (\xi_2 - \gamma_u - s) + (\gamma_u + \delta_{car}) + \epsilon_1 - (\xi_2 - \gamma_u - s) - \gamma_u - \delta_{car} - \gamma_d \\
& = \epsilon_1 - \gamma_d
\end{aligned}$$

as required.

Subcase 1b: $\gamma_u + \delta_{car} < \epsilon_2 - \xi_1 - t'''$.

Now

$$\min\{\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1\} = \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1$$

Again adding the upper bounds we find that the maximum amount of time that can pass between a k event and a *down* event is

$$\begin{aligned}
& t''' + (\epsilon_2 - \xi_1 - t''') + \min\{\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1\} \\
& \leq t''' + (\epsilon_2 - \xi_1 - t''') + \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1 \\
& \leq \epsilon_1 - \gamma_d
\end{aligned}$$

as required.

Case 2: $\xi_2 - \gamma_u - s > \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d$. Then $t''' \leq \epsilon_1 - \gamma_u - \delta_{car} - \gamma_d$.

Subcase 2a: $\gamma_u + \delta_{car} \geq \epsilon_2 - \xi_1 - t'''$.

Again adding the upper bounds we find that the maximum amount of time that can pass between a k event and a *down* event is

$$\begin{aligned}
& t''' + (\gamma_u + \delta_{car}) + \min(\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1) \\
& \leq t''' + \gamma_u + \delta_{car} + \epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d \\
& = \epsilon_1 - \gamma_d
\end{aligned}$$

as required.

Subcase 2b: $\gamma_u + \delta_{car} < \epsilon_2 - \xi_1 - t'''$.

Again adding the two upper bounds we find that the maximum amount of time that can pass between a k event and a *down* event is

$$\begin{aligned}
& t''' + (\epsilon_2 - \xi_1 - t''') + \min(\epsilon_1 - t''' - \gamma_u - \delta_{car} - \gamma_d, \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1) \\
& \leq t''' + \epsilon_2 - \xi_1 - t''' + \epsilon_1 - \gamma_d - \epsilon_2 + \xi_1 \\
& = \epsilon_1 - \gamma_d
\end{aligned}$$

as required.

These four cases are exhaustive, so we have proved safety for the path $P4, Q1, Q2, Q3, P3$.

The proofs for the other two paths from P_3 are similar but rather easier.

□

Theorem 5 *The specification satisfies the Utility Property.*

Proof 3 We may divide this property into *Inward Utility* and *Outward Utility*.

Inward Utility. We are required to show that from all states at which a *down* signal is sent, the delay until the arrival of a train in I is no greater than ξ_1 , or equivalently that no *down* signal is sent unless at least $\epsilon_2 - \xi_1$ time units have passed since a k event, and there has been no m event since this k event.

Outward Utility We are required to show that within ξ_2 time units of an m event the gate reaches a fully up position.

In each case a little manipulation of the information on the graph suffices to prove the property and we omit the details.

□

Theorem 6 *The specification satisfies the Common Sense Property.*

Proof 4 We are required to show that if an *up* signal is sent at time t then no down signal is sent between t and $t + \gamma_u + \delta_{car} + \gamma_d$. It is a simple matter to read this information off the graph.

□

6 Conclusions

6.1 Evaluating the Specifications

6.1.1 Comparison of the Process Algebra Specifications

It was found that the expressiveness of the process algebras used was essentially equivalent, though individually they have different features. In particular the lack of indexed families of processes in TE-LOTOS required the passing of values between processes; also Timed CCS is the only one to use absolute rather than relative time. It should be noted that the difference in complexity between the Timed CSP and the Timed CCS and TE-LOTOS specifications is solely a consequence of the different assumptions used in their development. The major difference between them became evident during the development of the proofs—only Timed CSP has a well developed proof system. In conclusion it appears that the choice of which process algebra to use is a matter of personal taste.

6.1.2 Comparison of the Timed Z and Process Algebra Specifications

The major difference between the Timed Z specification and the process algebra specifications is the level of abstraction. The Timed Z specification can be thought of as closer to the English description of the problem than are the process algebra specifications. This is evident in the fact that fewer of the auxiliary assumptions that were discovered while developing the process algebra specifications were needed in the Timed Z specification. The consequences of this observation are that a Timed Z specification will need considerable refinement to reach an implementation, but that the initial specification gives a readily understandable overview of the system under consideration.

6.2 Evaluating the Generalised Railway Crossing

The Generalised Railway Crossing problem is a useful benchmark problem for real time systems. It has proved to be more difficult than was evident upon initial investigation, specifically determining the auxiliary assumptions and time bounds for events was quite demanding, particularly since not all could be derived by analysing the statement of the problem—others were a consequence of knowledge of real world railway crossings.

The approach taken in this paper follows the statement of the problem very closely unlike some other attempts in the literature. For example, [16] narrows the problem so far as to have produced an implementation, while [10] makes unjustified assumptions about extra sensors. In general the customer's statement of requirements should be respected as far as possible, and deviations from it should only be made when justified by ambiguities and omissions in the original statement.

References

- [1] L. Chen. *Timed Processes: Models, Axioms and Decidability*. PhD thesis, The University of Edinburgh, Department of Computer Science, 1993.
- [2] J. Davies. *Specification and Proof in Real-Time CSP*. Cambridge University Press, 1993. D. Phil thesis, Oxford University Computing Laboratory.
- [3] J. Davies. *Setting Real-Time CSP*. Computing Laboratory, Oxford University, 1994.
- [4] A. S. Evans. An improved recipe for specifying reactive systems in Z. Submitted to ZUM '97.
- [5] A. S. Evans. *Z for concurrent systems*. PhD thesis, Leeds Metropolitan University, May 1996.
- [6] C.L. Heitmeyer, R.D. Jeffords, and B.G. Labaw. A benchmark for comparing different approaches for specifying and verifying real-time systems. In *Proceedings of the 10th International Workshop on Real-Time Operating Systems and Software*, May 1993.

- [7] C.L. Heitmeyer and N. Lynch. The generalized railroad crossing: A case study in formal verification of real-time systems. Technical Report NRL/MR/5540-94-7619, Naval Research Laboratory, Washington DC, USA, December 1994.
- [8] C.L. Heitmeyer and Dino Mandrioli, editors. *Formal Methods for Real-Time Computing*. Number 5 in Trends in Software. John Wiley and Sons, 1996.
- [9] G. Jones. *A Timed Model of Communicating Process*. PhD thesis, Computing Laboratory, Oxford University, 1982.
- [10] I. Lee, H. Ben-Abdallah, and J. Choi. A process algebraic method for the specification and analysis of real-time systems. In C. Heitmeyer and D. Mandrioli, editors, *Formal Methods for Real-Time Computing*, chapter 7. Wiley, 1996.
- [11] L. Léonard, G.Leduc, C. Miguel, J. Quemada, G. Rabay, D. de Frutos, and L. Llana. Time extended LOTOS, draft proposal. Obtained via anonymous ftp from `ftp.dit.upm.es`, 1994.
- [12] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.
- [13] G.M. Reed and A.W Roscoe. Metric spaces as models for real-time concurrency. In *Proceedings of the Third Workshop on the Mathematical Foundations of Programming Language Semantics*, number 298 in LNCS. Springer Verlag, 1989.
- [14] S.A. Schneider. *Correctness and Communication in Real-Time Systems*. PhD thesis, Computing Laboratory, Oxford University, 1989.
- [15] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 1992.
- [16] W.D. Young. Modelling and verification of a simple real-time railroad gate controller. In M.G. Hinchey and J.P. Bowen, editors, *Applications of Formal Methods*, chapter 8. Prentice Hall, 1995.

A Appendix

Lemma 1 *Counter monitors the number of trains in R correctly and sends a signal k when the first train enters R and a signal m when the last train leaves R . That is, if*

$$s \downarrow a - s \downarrow l = i,$$

then the control of Counter is at the beginning of $Counter_i$; furthermore, if $last(s) = a$, the control is transferred from $Counter_{i-1}$; if $last(s) = l$, the control is transferred from $Counter_{i+1}$.

Proof: (By induction on the length of the trace.) As this property is independent of timing considerations, we use the untimed trace proof system. We observe that

$$tr \downarrow a - tr \downarrow l = i \Rightarrow s \downarrow a - s \downarrow l = i.$$

Base: When *Counter* starts, $tr = \langle \rangle$. We have

$$\langle \rangle \downarrow a - \langle \rangle \downarrow l = 0.$$

We apply the inference rule for event prefix to yield, at the beginning of $Counter_1$, $tr = \langle a, k \rangle$ and

$$\langle a, k \rangle \downarrow a - \langle a, k \rangle \downarrow l = 1.$$

By the definition of internal choice, there are two ways for $Counter_1 \stackrel{\text{def}}{=} a \rightarrow Counter_2 \sqcap l \rightarrow Counter_0$ to go:

(a) If another train enters R , we have

$$\langle a, k, a \rangle \downarrow a - \langle a, k, a \rangle \downarrow l = 2$$

and the control goes to $Counter_2$.

(b) If the first train leaves R , we have

$$\langle a, k, l \rangle \downarrow a - \langle a, k, l \rangle \downarrow l = 0$$

and $Counter$ goes to $Counter_0$; signal m is then sent and we have

$$\langle a, k, l, m \rangle \downarrow a - \langle a, k, l, m \rangle \downarrow l = 0$$

and the control goes to the beginning of $Counter$.

Induction hypothesis: Assume the lemma holds for trace tr with $|tr| = n$, for $n \geq 0$.

Induction step: We consider $tr \frown \langle e \rangle$, where e is in $\{a, l, k, m, \}$. Assume the control is at the beginning of $Counter_i$ with tr , for $i \geq 0$, and $tr \downarrow a - tr \downarrow l = i$. There are two cases to consider.

First, we consider the case where $i \geq 1$, that is,

$$\begin{aligned} Counter_i &\stackrel{\text{def}}{=} a \rightarrow Counter_{i+1} \\ &\quad \sqcap l \rightarrow Counter_{i-1}. \end{aligned}$$

By the definition of internal choice $\sqcap$, there are two ways for $Counter_i$ to go:

(a) If another train enters R , we have

$$\begin{aligned} &tr \frown \langle a \rangle \downarrow a - tr \frown \langle a \rangle \downarrow l \\ &= tr \downarrow a + 1 - tr \downarrow l \\ &= i + 1 \end{aligned}$$

and the control goes to $Counter_{i+1}$.

(b) If another train leaves R , we have

$$\begin{aligned} &tr \frown \langle l \rangle \downarrow a - tr \frown \langle l \rangle \downarrow l \\ &= tr \downarrow a - (tr \downarrow l + 1) \\ &= i - 1 \end{aligned}$$

and the control goes to $Counter_{i-1}$.

Secondly, we consider the case where $i = 0$. We have

$$Counter_0 = m \rightarrow Counter \quad \text{and} \quad tr \downarrow a - tr \downarrow l = 0.$$

By the definition of event prefix, tr can only be extended by an m . We have

$$tr \frown \langle m \rangle \downarrow a - tr \frown \langle m \rangle \downarrow l = 0,$$

and the control goes to the beginning of $Counter$, which concludes the proof. $\square$