To appear in Formal Methods for Real-Time Computing, *C. Heitmeyer and D. Mandrioli (eds)*
(Jan 1996)

Railroad Crossing Systems

Insup Lee, Hanêne Ben-Abdallah, Jin-Young Choi
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

The railroad crossing system [HJL93] has been a benchmark example for real-time formal methods. In this section, we illustrate how ACSR can be used to model three versions of the railroad crossing system: the *Generalized Railroad Crossing* [HL94] which has multiple tracks in the crossing; the *Standard Railroad Crossing* which is an instance of the Generalized Railroad Crossing with a single-track crossing; and the *Busy Railroad Crossing* which augments the Standard Railroad Crossing with cars. We also illustrate how ACSR can be used to verify some properties of the Standard Railroad Crossing and Busy Railroad Crossing.

To make the ACSR specifications easier to read, we adopt the following naming convention: synchronization events start with lower-case letters and observable events start with upper-case letters.

1 Informal Description And Parameters

The system to be developed controls a gate at a railroad crossing. The railroad crossing lies in a region delimited by entry and exit sensors that detect the entry and exit of trains. The goal is to develop a system that operates the crossing gate subject to *safety* and *utility* properties [HL94]. The safety property states that when a train is in the crossing, the gate is down. The utility property states that when no train is in the crossing, the gate is up as much as possible.

Our designs contain three main components: *train* and *gate* which describe the environment, and *control* which describes the computer system. These components execute concurrently, produce *observable* events to mark the status of critical activities, and coordinate with one another through *synchronization* events. Tables 1 and 2 describe the names of events that we use in our designs.

Table 1: Observable Events

Train			**Gate**		
Nc	:	near crossing	Up	:	is up
Ic	:	in crossing	$Down$	:	is down
Pc	:	past crossing			

Table 2: Synchronization Events

Train → Control			**Control → Gate**		
ns	:	signal entry sensor	$raise$	:	signal gate to move up
ps	:	signal exit sensor	$lower$	:	signal gate to move down

Train. We model a train as a process that sends three observable events, Nc, Ic, and Pc, to mark the train's critical positions, and that coordinates with the control component through two synchronization events, ns and ps. When a train approaches the crossing region, it marks this activity by sending Nc. At this time, the train is detected by the entry sensor and the train sends the event ns to the control component. When a train enters the crossing, it sends Ic. Finally, when a train passes the crossing, it sends Pc and sends the exit-sensory event ps to control.

Gate. We model the gate, which is initially up, as a process that receives signals from the control component. When it receives $lower$, it moves down and marks the end of this activity by the event $Down$. When gate receives $raise$, it moves up and marks the end of this activity by the event Up.

Control. We model the control component as a process that synchronizes the train activities with the gate: When the control receives ns from the train, it sends $lower$ to the gate; and when it receives ps, it sends $raise$ to the gate.

Being part of a safety-critical system, the components of the railroad crossing system operate under timing assumptions. The timing assumptions are expressed in terms of delays between observable events in the system. Table 3 shows the timing assumptions we use in our designs of the railroad crossing system. Note that, in this paper, we elected to use single valued delays, as oppose to intervals, to keep the examples simple; however, we can describe in ACSR any interval delay through the Prefix and Scope operators.

Table 3: Timing Assumptions

	Bounds	**From Event$_1$ to Event$_2$**
Train	ni	$Nc \rightarrow Ic$
	ip	$Ic \rightarrow Pc$
	$[\delta, \infty]$	$Pc_i \rightarrow Nc_{i+1}$
Gate	γ	to move up or down

2 Generalized Railroad Crossing

The Generalized Railroad Crossing, GRC, contains n tracks in the crossing. Thus, at any time, there can be up to n trains approaching, leaving, or in the crossing. Figure 1 shows our design specification of the GRC system.

Each $Train_i$ process can be either far from the crossing region, which is indicated by the idle action, or it can be near the crossing region, which is marked by the event Nc_i. The *Control* process idles while waiting to receive events from trains that approach or leave the crossing. This process remembers the number of trains that consecutively enter the crossing region and sends a *lower* event to the *Gate* process when the first train enters the crossing region and a *raise* event when the last train passes the crossing region. The *Gate* process idles while waiting for events from the *Control* process to move down or up. Note that in our design of the gate if a train enters the crossing before the gate is fully raised, the gate-raising activity is interrupted and the gate-closing activity is started. On the other hand, if no train is near the crossing before the gate is fully raise, the gate-raising activity signals its completion by sending the event $\overline{open}$ after which the gate returns to the initial state. This is described by the Scope operator in the process *GU* of Figure 1.

3 Standard Railroad Crossing

The Standard Railroad Crossing, SRC, is a special instance of the GRC with only one track in the crossing; that is, at any time, there is at most one train in the crossing region. We can therefore instantiate the GRC specification of Figure 1 for $n = 1$ and a fixed set of timing assumptions. Figure 1 shows an example of SCR for the timing assumptions in Table 4.

To verify formally that the design specification *SRC* satisfies the safety and utility properties, we construct a requirements specification that is simpler to inspect for the two properties. Figure 3 shows two requirements specifications for the SRC: $Spec_1$ describes all possible actions including synchronized events which are represented by the internal

$$
\begin{array}{rcl}
GRC & \stackrel{\text{def}}{=} & (Trains \parallel Control \parallel Gate) \setminus \{ns, ps, lower, raise\} \\
\hline
Trains & \stackrel{\text{def}}{=} & Train_1 \parallel \cdots \parallel Train_n \\
Train_1 & \stackrel{\text{def}}{=} & \emptyset : Train_1 + \\
 & & (\overline{Nc}_1, 1).(\overline{ns}, 1).\emptyset^{\mathsf{ni}} : (\overline{Ic}_1, 1).\emptyset^{\mathsf{ip}} : (\overline{Pc}_1, 1).(\overline{ps}, 1).\emptyset^{\delta} : Train_1 \\
 & \vdots & \\
Train_n & \stackrel{\text{def}}{=} & \emptyset : Train_n + \\
 & & (\overline{Nc}_n, 1).(\overline{ns}, 1).\emptyset^{\mathsf{ni}} : (\overline{Ic}_n, 1).\emptyset^{\mathsf{ip}} : (\overline{Pc}_n, 1).(\overline{ps}, 1).\emptyset^{\delta} : Train_n \\
\hline
Control & \stackrel{\text{def}}{=} & \emptyset : Control + (ns, 1).(\overline{lower}, 1).Control_1 \\
Control_1 & \stackrel{\text{def}}{=} & \emptyset : Control_1 + (ns, 1).Control_2 + (ps, 1).(\overline{raise}, 1).Control \\
Control_2 & \stackrel{\text{def}}{=} & \emptyset : Control_2 + (ns, 1).Control_3 + (ps, 1).Control_1 \\
 & \vdots & \\
Control_n & \stackrel{\text{def}}{=} & \emptyset : Control_n + (ps, 1).Control_{n-1} \\
\hline
Gate & \stackrel{\text{def}}{=} & \emptyset : Gate + (lower, 1).GD \\
GD & \stackrel{\text{def}}{=} & \emptyset^{\gamma} : (\overline{Down}, 1).rec\, X.(\emptyset : X + (raise, 1).GU) \\
GU & \stackrel{\text{def}}{=} & (\emptyset^{\gamma} : (\overline{Up}, 1).(\overline{open}, 1).\text{NIL}) \triangle_{\infty}^{open} (Gate,\ \text{NIL},\ (lower, 1).GD)
\end{array}
$$

Figure 1: Generalized Railroad Crossing

event τ, while $Spec_2$ abstracts out the internal events.

In both requirements specifications, the idling for 20 time units corresponds to the time to lower the gate. The idling for 5 time units ($5 = 25 - 20$) corresponds to the time it remains for a train to actually enter the crossing after signaling its approach. The idling for 10 time units is the time for a train to pass the crossing. The idling for 15 time units is the minimum delay between consecutive trains. Finally, the time-out of 5 time units in the scope operator corresponds to the remaining time to fully raise the gate after a train exits the crossing.

In both requirements specifications, one can see that any occurrence of the event Ic is preceded by a $Down$ event. In addition, any occurrence of the event Up is preceded by the event Pc. The gate is, therefore, down whenever there is a train in the crossing. Furthermore, one can easily inspect that in both requirements specifications, if no event Nc ever occurs, the gate remains in its initial state which is, by assumption, up. On the

Table 4: Instances of Timing Assumptions

δ	$= 15$	γ	$= 20$
ni	$= 25$	ip	$= 10$

$$
\begin{array}{rcl}
SRC & \stackrel{\text{def}}{=} & (Train \parallel Control \parallel Gate)\backslash\{ns, ps, lower, raise\} \\
\hline
Train & \stackrel{\text{def}}{=} & \emptyset : Train \; + \; (\overline{Nc}, 1).(\overline{ns}, 1).\emptyset^{25} : (\overline{Ic}, 1).\emptyset^{10} : (\overline{Pc}, 1).(\overline{ps}, 1).\emptyset^{15} : Train \\
\hline
Control & \stackrel{\text{def}}{=} & \emptyset : Control + (ns, 1).(\overline{lower}, 1).Control + \; (ps, 1).(\overline{raise}, 1).Control \\
\hline
Gate & \stackrel{\text{def}}{=} & \emptyset : Gate + (lower, 1).GD \\
GD & \stackrel{\text{def}}{=} & \emptyset^{20} : (\overline{Down}, 1).rec\, X.(\emptyset : X + (raise, 1).GU) \\
GU & \stackrel{\text{def}}{=} & (\emptyset^{20} : (\overline{Up}, 1).(\overline{open}, 1).\text{NIL}) \; \triangle_{\infty}^{open} \; (Gate, \; \text{NIL}, \; (lower, 1).GD)
\end{array}
$$

Figure 2: Standard Railroad Crossing

other hand, if no event Nc occurs within 20 time units of the last occurrence of Pc, the event Up occurs. The gate is therefore up when no train is in the crossing, as much as possible taken the time to raise the gate.

We can use the ACSR analysis techniques to prove that the design specification of the standard railroad crossing, SRC, satisfies the safety and utility properties by showing it is equivalent to the requirements specifications in Figure 3. More precisely, we can prove that

$$
\begin{array}{rcl}
SRC & \sim_{\pi} & Spec_1 \text{ and} \\
SRC & \approx_{\pi} & Spec_2
\end{array}
$$

4 Busy Railroad Crossing

To illustrate the use of resources and priorities of ACSR, we augment the SRC system with cars. We call this version the *Busy Railroad Crossing*, BRC.

Figure 4 shows our design specification of the BRC. (The $Control$ process is the same as in Figure 2.) The specification of the train in Figure 4, $Train'$, differs from $Train$ in Figure 2 in the use of the *crossing* resource when the train enters the crossing. We

$$Spec_1 \stackrel{\text{def}}{=} \emptyset : Spec_1 \ + \ Spec'_1$$

$$Spec'_1 \stackrel{\text{def}}{=} (\overline{Nc}, 1).(\tau, 2).(\tau, 2).\emptyset^{20} : (\overline{Down}, 1).\emptyset^{5} : (\overline{Ic}, 1).\emptyset^{10} : (\overline{Pc}, 1).(\tau, 2).(\tau, 2).\ \emptyset^{15} : ((rec\, X.\emptyset : X) \triangle_5 (\text{NIL},\ (\overline{Up}, 1).(\tau, 2).Spec_1,\ Spec'_1))$$

$$Spec_2 \stackrel{\text{def}}{=} \emptyset : Spec_2 \ + \ Spec'_2$$

$$Spec'_2 \stackrel{\text{def}}{=} (\overline{Nc}, 1).\emptyset^{20} : (\overline{Down}, 1).\emptyset^{5} : (\overline{Ic}, 1).\emptyset^{10} : (\overline{Pc}, 1).\ \emptyset^{15} : ((rec\, X.\emptyset : X) \triangle_5 (\text{NIL},\ (\overline{Up}, 1).Spec_2,\ Spec'_2))$$

Figure 3: Requirements specification for the Standard Railroad Crossing

modify the *Gate* process of Figure 2 to signal *up* while the gate is up. This event is used by the *Car* process to find out the status of the gate. We assume that a car takes three time units to pass the crossing and returns to the crossing immediately afterward. When a car passes the crossing, it uses the *crossing* resource. To arbitrate resource conflict, we assign a higher priority to the train than to the car on their use of the *crossing* resource. To simplify the requirements specification, we use the same event Nc for a car and a train that signal their arrival to the crossing with a car having a higher priority.

$$BRC \stackrel{\text{def}}{=} [\,(Train' \,||\, Control \,||\, Gate' \,||\, Car) \backslash \{ns, ps, lower, raise, up\}\,]_{\{crossing\}}$$

$$Train' \stackrel{\text{def}}{=} \emptyset : Train' + (\overline{Nc}, 1).(\overline{ns}, 1).\emptyset^{25} : TC$$

$$TC \stackrel{\text{def}}{=} (\overline{Ic}, 1).\{(crossing, 5)\}^{10} : (\overline{Pc}, 1).(\overline{ps}, 1).\emptyset^{15} : Train'$$

$$Gate' \stackrel{\text{def}}{=} \emptyset : Gate' + (\overline{up}, 1).Gate' + (lower, 1).GD$$

$$GD \stackrel{\text{def}}{=} \emptyset^{20} : (\overline{Down}, 1).rec\, X.(\emptyset : X + (raise, 1).GU)$$

$$GU \stackrel{\text{def}}{=} (\emptyset^{20} : (\overline{Up}, 1).(\overline{open}, 1).\text{NIL}) \triangle_{\infty}^{open} (Gate',\ \text{NIL},\ (lower, 1).GD)$$

$$Car \stackrel{\text{def}}{=} (\overline{Nc}, 2).rec\, X.(\emptyset : X + (up, 1).\{(crossing, 2)\}^{3} : Car)$$

Figure 4: Busy Railroad Crossing

In addition to the safety and utility properties of the SRC, the BRC system must

satisfy an additional safety property: a train and a car never collide into each other. This additional safety property is satisfied if the design specification, BRC, never reaches a deadlock state that is due to the concurrent execution of the action $\{(crossing, 5)\}$ and the action $\{(crossing, 2)\}$ as the only possible actions for $Train$ and Car, respectively.

$$
\begin{array}{rl}
Spec_3 & \stackrel{\text{def}}{=} (\overline{Nc}, 2).Spec'_3 \\
Spec'_3 & \stackrel{\text{def}}{=} (\overline{Nc}, 1).\emptyset^{20} : TPass \\
& \quad +\emptyset : ((\overline{Nc}, 1).\emptyset^{2} : (\overline{Nc}, 2).\emptyset^{18} : TPass \\
& \qquad +\emptyset : ((\overline{Nc}, 1).\emptyset : (\overline{Nc}, 2).\emptyset^{19} : TPass \\
& \qquad\quad +\emptyset : Spec_3)) \\
TPass & \stackrel{\text{def}}{=} (\overline{Down}, 1).\emptyset^{5} : (\overline{Ic}, 1).\emptyset^{10} : (\overline{Pc}, 1).Spec''_3 \\
Spec''_3 & \stackrel{\text{def}}{=} (rec\, X.\emptyset : X) \triangle_5 (\text{NIL}, (\overline{Up}, 1).Spec'_3, (\overline{Nc}, 1).\emptyset^{20} : TPass)
\end{array}
$$

Figure 5: Requirements Specification of the Busy Railroad Crossing

To verify our claims about the safety and utility of BRC formally, we use the requirements specification $Spec_3$ of Figure 5. The requirements specification $Spec_3$ is a purely sequential process that abstracts out synchronization and resource usage. It is simple to inspect for the safety and utility properties. Furthermore, it is obvious that $Spec_3$ never deadlocks due to resource contention. Thus $Spec_3$ describes the desired behavior of the BRC. We can use the ACSR analysis techniques to prove that

$$BRC \backslash\backslash \{crossing\} \approx_\pi Spec_3.$$

We can therefore conclude that our design specification BRC satisfies the augmented safety property and the utility property.

References

[HJL93] C. Heitmeyer, R. Jeffords, and B. Labaw. Comparing Different Approaches for Specifying and Verifying Real-Time Systems. In *Proc.* 10^{th} *IEEE Workshop on Real-Time Operating Systems and Software*, pages 122–129, May 1993.

[HL94] C. Heitmeyer and N. Lynch. The generalized railroad crossing: A case study in formal verification of real-time systems. In *Proceedings of Real-Time Systems Symposium (RTSS' 94)*, San Juan, Puerto Rico, December 1994.