A Case Study in Timed CSP: The Railroad Crossing Problem

Luming Lai and Phil Watson

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

Abstract. We use timed CSP, which is an extension of the formal method CSP to problems with a real-time component, to tackle a benchmark problem for real-time systems, the railroad crossing problem.

1 The problem and associated assumptions

The railroad crossing problem was specified in Heitmeyer [5, 4] and the description was further clarified in [3].

> 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 ith occupancy interval is represented as $\lambda_i = [\tau_i, \nu_i]$ where τ_i is the time of the ith 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: $t \in \bigcup_i \lambda_i \implies g(t) = 0$ (The gate is down during all occupancy intervals.)

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

The description, though seemingly comprehensive, contains a number of ambiguities and omissions. One defect in the specification is as follows. It is nowhere explicitly forbidden in the English statement of the problem, that the gate should be raised and almost immediately lowered. In reality there would be an interval γ_c sufficient for at least one car to cross between the gate reaching the fully raised position and a *down* signal being sent; otherwise the gate would not be raised at all. We call this 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 railroad crossings drives us to do this.

Various constraints on the parameters of the problem are given in [4] or derived in [3]. These include:

1. $\epsilon_2 \geq \epsilon_1$. The time for the slowest train to reach I after entering R, ϵ_2, is greater than or equal to the time for the fastest train to reach I, ϵ_1
2. $\epsilon_1 \geq \gamma_d + \gamma_c + \gamma_u$. Here γ_d and γ_u are the times required to lower and raise the physical gate, respectively. If the property does not hold we can never open the gate after the first train has passed without losing Safety. Clearly this is not the intent of the English statement of the problem
3. $\xi_1 - \gamma_d \geq \epsilon_2 - \epsilon_1$. The speeds of the fastest and slowest trains on the track must not differ so greatly that safety and utility are impossible to guarantee together
4. $\xi_2 \geq \gamma_u$. The time taken to raise the gate, γ_u, is small enough that the Utility Property is not violated
5. $\epsilon_2 \geq \xi_1$. Otherwise we could send a *down* signal before a train is detected approaching. Clearly this would be unrealistic.

We make only one assumption which is not strictly justified by the English specification: that for an *up* signal to be sent, R should have been empty at some time since the last *up* signal was sent. This is not an unreasonable assumption.

2 The timed CSP specification of the railroad crossing

Space limitations preclude a full description of the features and use of Timed CSP [1]. We introduce some of its features as and when they occur.

***GateController* and *Counter* Processes.** The main process for the railroad crossing in timed CSP consists of two component processes.

$$GateController \stackrel{\text{def}}{=} (Counter \,|[\,\{k, m\}\,]|\, Controller) \setminus \{k, m\}$$

Here $P\,|[\,A\,]|\,Q$ means that processes P and Q synchronise on the events in the set A.

Most of the interest lies in the *Controller* process. *Counter* keeps an implicit count (in the index of the current sub-process) of the number of trains currently in R and communicates with the *Controller* process in two ways: a k signal means that a train has arrived in an otherwise empty R; an m signal means that the last train in R has just left.

$$\begin{aligned}
Counter &\stackrel{\text{def}}{=} a \rightarrow k \rightarrow Counter_1 \\
Counter_i &\stackrel{\text{def}}{=} a \rightarrow Counter_{i+1} \;\Box\; d \rightarrow Counter_{i-1} && \text{for } i \geq 1 \\
Counter_0 &\stackrel{\text{def}}{=} m \rightarrow Counter,
\end{aligned}$$

Here $\sqcap$ is non-deternministic choice, while a represents the arrival of a train in R and d the departure of a train from the crossing.

***Controller** Process.* For brevity we define the following abbreviations, which are based on required properties of the specification. Let

$t_{s1} \stackrel{\text{def}}{=} \xi_1 - \epsilon_2 + \epsilon_1 - \gamma_d$. The length of the interval during which a *down* signal may be sent. Note that $t_{s1} \geq 0$ by assumption 3. above

$t_{u1} \stackrel{\text{def}}{=} \epsilon_2 - \xi_1$. The lower time bound for sending a *down* signal. We call t_{u1} the *inward utility point* and $t_{u1} \geq 0$ by 5. above

$t_{u2} \stackrel{\text{def}}{=} \xi_2 - \gamma_u$. We call t_{u2} the *outward utility point* and $t_{u2} \geq 0$ by 4. above.

$t_{c1} \stackrel{\text{def}}{=} \gamma_u + \gamma_c$. The time it takes to raise the gate and let one car through

$t_{s2} \stackrel{\text{def}}{=} \epsilon_1 - (\gamma_u + \gamma_c + \gamma_d)$. The upper time bound for sending a *down* signal to satisfy the Safety and Common Sense Properties; $t_{s2} \geq 0$ by assumption 2. in Section 1.

Thus far we have used (untimed) CSP. The subsequent specification will omit the fixed delay δ between CSP events for clarity. It is untidy (especially in the proofs) but not difficult to introduce δ to the following specification.

*The specification of **Controller**.* To define new processes which can (additionally) satisfy the Common Sense Property, we need the following abbreviations in addition to those above: The *Controller* is defined as follows:

$$Controller \stackrel{\text{def}}{=} k \xrightarrow{t_{u1}} down@t_1\{t_{s1}\} \rightarrow P$$

$$P \stackrel{\text{def}}{=} m \rightarrow (up@t_2\{t_{u2}\} \rightarrow k@t_3 \xrightarrow{max\{t_{c1}-t_3,\, t_{u1}\}} down@t_4\{min\{t_{s1},\, t_{s2}+t_3\}\} \rightarrow P$$
$$\Box\ k@t_5\{t_{u2}\} \rightarrow up@t_6\{min\{t_{u2}-t_5,\, t_{s2}\}\} \xrightarrow{max\{t_{c1},\, t_{u1}-t6\}}$$
$$down@t_7\{min\{t_{s1},\, t_{s2}-t_6\}\} \rightarrow P).$$

Note the additional complexity entailed by the various ways in which the requirements for our Common Sense Property can conflict with those for Safety and Utility.

To prove that the above processes satisfy the Safety, Utility, and Common Sense 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 an empty R and a signal "m" when the last train leaves R.*

We define a number of predicates for structured specifications as follows.

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

If a process satisfies this specification, then 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.

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

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

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

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

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

which means that a immediately precedes the b at time t.

Notice that, in the following lemmas, theorems and their proofs, we can only prove that our processes will behave as required to satisfy the Safety, Utility, and Common Sense Properties. To guarantee these properties, the cooperation of their environment is also required, such as the gate will not be jammed and the *up* and *down* signals will be received by the gate whenever they are sent.

Lemma 2 *When the first train enters the empty region R, Controller will be ready to send a "down" signal in time for the gate to be fully closed before the train enters the crossing, i.e.,*

$$\begin{aligned} &\textit{Controller}\ \ \textbf{sat} \\ &\quad k\ \underline{at}\ t \wedge \neg(m\ \underline{precedes}\ (t,\ k)) \ \Rightarrow\ down\ \underline{live}\ [t + max\{t_{c1}, t_{u1}\},\ t + \epsilon_1 - \gamma_d] \\ &\wedge\ k\ \underline{at}\ t \wedge m\ \underline{precedes}\ (t,\ k) \wedge up\ \underline{at}\ (t + t_6) \wedge t_6 \leq min\{t_{u2},\ t_{s2}\} \Rightarrow \\ &\qquad\qquad down\ \underline{live}\ [t + max\{t_{u1}, t_6 + t_{c1}\},\ t + \epsilon_1 - \gamma_d], \end{aligned} \tag{1}$$

and $max\{t_{u1},\ t_6 + t_{c1}\} \ \leq\ \epsilon_1 - \gamma_d$.

Lemma 3 *When the first train enters an empty region R, Controller will wait until it reaches the inward utility point t_{u1} before sending a "down" signal, i.e.,*

$$\textit{Controller}\ \ \textbf{sat}\ \ k\ \underline{at}\ t \Rightarrow \neg(down\ \underline{at}\ [t,\ t + t_{u1})) \tag{2}$$

Lemma 4 *When the last train leaves R, Controller will be ready to send an "up" signal before the outward utility point t_{u2} so that the gate will be fully up before the train passes the second utility point ξ_2, i.e.,*

$$\textit{Controller}\ \textbf{sat}\ m\ \underline{at}\ t \Rightarrow up\ \underline{live}\ [t,\ t + t_{u2}].$$

Theorem 1 (The Common Sense Property) *Controller satisfies the Common Sense Property, i.e., after an "up" signal being sent, it will wait for at least t_{c1} time units before sending a "down" signal:*

$$\textit{Controller}\ \textbf{sat}\ up\ \underline{at}\ t \Rightarrow \neg(down\ \underline{at}\ [t,\ t + t_{c1}]).$$

Theorem 2 (The Safety Property) *GateController satisfies the Safety Property, provided that the gate behaves properly, i.e.,*

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

Theorem 3 (The Utility Property) *GateController satisfies the Utility Property, provided that the gate behaves properly, i.e.,*

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

3 Related Work

The railroad crossing problem is a real-time specification problem of great generality and surprising subtlety. It has been attempted by numerous authors with as many different techniques [5], though not until now with timed CSP, to the best of the authors' knowledge, although a simpler version of the system was specified in timed CSP in [3].

Due to space restrictions, it has not proved possible to present the proofs of the Lemmas and Theorems presented here. These appear in [7]. The use of the Timed CSP Proof System was found to be natural, simple and efficient. Comparison [3] with the proof systems available in other formal notations is a strong argument in favour of Timed CSP.

We have considered the case of a crossing over a single track, which may be easily extended to multiple tracks, and indeed other authors take the same approach. We assume 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. Other authors [8] attempting this problem have assumed the existence of additional sensors, but this is not justified by the English specification above.

Our general approach has been to regard our task as one of presenting the original English statement of the problem in a formal language and proving that any implementation of this specification has certain properties. In particular, we have made our additional assumptions explicit. This approach - regarding the provider of the English specification as a customer and the final CSP specification as the customer's desired product - seems natural but has not been universally adopted. Some authors [9] have narrowed the specification so far from the English statement of the problem that they have produced an implementation instead of a specification.

4 Conclusions

We have specified the benchmark railroad crossing problem in Timed CSP and noted some of the lessons learned about the problem and the formal method.

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

of a dedicated proof system for timed CSP means that it scores over other timed process algebras which lack such a system.

The interplay of the conflicting requirements leads to a many-branching process. It is instructive to consider the ways in which more or less justified real-world assumptions can be used to control and reduce this branching.

The biggest benefit of doing the detailed proofs (see [7]) is that they have helped a great deal in the design of the system. Even during the proof, we often had to go back to correct the definitions of our processes. There are well-known difficulties in scaling up the application of formal methods to real-world problems, but in such cases the difficulties of producing a correct design without use of formal techniques are scaled up also. We believe that it would be very difficult to design a correct real-time system of any realistic size without the help of formal methods.

References

1. J. Davies. *Specification and Proof in Real-Time CSP.* D.Phil thesis, Computing Laboratory, Oxford University, published by Cambridge University Press, 1993.
2. J. Davies. Setting real-time CSP. Internal note, Computing Laboratory, Oxford University, 1994.
3. A.S. Evans, D.R.W. Holton, L. Lai and P. Watson. A comparison of formal real-time specification languages. In *Proceedings of the Northern Formal Methods Workshop*, Ilkley, UK, 1996.
4. C.L. Heitmeyer and N. Lynch. The generalized railroad crossing: a case study in formal verification of real-time systems. NRL Memorandum Report NRL/MR/5540-94-7619, Navy Research Laboratory, Washington DC, USA, 1994.
5. C.L. Heitmeyer and D. Mandrioli (eds.). *Formal Methods for Real-Time Computing.* John Wiley & Sons, 1996.
6. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, London, 1985.
7. L. Lai, P. Watson, A case study in Timed CSP: the Railroad Crossing Problem, Technical Report CS-01-97, Dept. of Computing, University of Bradford, January 1997.
8. I. Lee, H. Ben-Abdallah and J. Choi. A process algebraic method for the specification and analysis of real-time systems. In [5].
9. W.D. Young. Modelling and verification of a simple real-time railroad gate controller. In M.G. Hinchey and J.P. Bowen (eds.), *Applications of Formal Methods*, Prentice Hall, 1995.