Decomposing scheme plans to manage verification complexity

Phillip James[2], Faron Moller[2], Hoang Nga Nguyen[3], Markus Roggenbach[2], Steve Schneider[1], and Helen Treharne[1]

[1] University of Surrey, UK {H.Treharne,S.Schneider}@surrey.ac.uk
[2] University of Swansea, Wales {P.D.James,M.Roggenbach}swansea.ac.uk
[3] University of Nottingham, UK

Abstract. Several formal methods have been proposed for the specification and safety verification of railway applications. In order to be successful they need industrial strength tools to support the animation, proof, model checking and simulation of such systems. The complexity of railway systems means that capability of the analysis tools have consistently been improving. In our approach we propose that the complexity of analysis of railway interlocking systems can also be managed through incremental addition of system detail and decomposition of system specifications themselves. We propose a domain specific language (DSL) which describes the core aspects of a railway interlocking system and demonstrate how we can identify suitable decompositions in terms of the DSL. The DSL informs our system engineering approach which uses a graphical editor to input railway scheme plans, supports the automatic generation of CSP || B specifications of the plans and uses the ProB tool for their animation and model checking.

1 Introduction

Fantechi in [3] notes that no one formal method has emerged as the single mature technology in the railway domain and that the verification of complex railway signalling systems remains a significant challenge. He argues that model checking is one of most promising automated verification techniques and has been referred to in the EN50128 guidelines [1]. Fantechi also notes that the EN50128 guidelines recognise the value in modelling and that it can be utilised at different stages of the development cycle of a railway signalling system. Thus, defining a formal model of a system together with its safety properties in terms of temporal logic provides the basis required for using model checking at an early stage of development. Model checking tools are used to demonstrate that the system description satisfies the safety properties. Failure to satisfy the properties yields counter examples which illustrate how the system behaviour evolves to break these properties. For example there may be a track missing from the control table of a route or a point is being released incorrectly giving rise to a collision, derailment or run-through.

It is well known in the literature that model checking large railway interlocking systems can lead to a state space explosion problem. One potential solution is the parallelisation of the verification effort to multi-core machines but this is looking at tackling the problem from the point of view of the supporting tools, for example [13]. Alternatively, SAT-based bounded model checking of formal models of a system is regarded as a promising solution to the problem from a modelling perspective. An example of an approach that utilises constraint solving via the ProB [9] model checker is the Safecap approach [4]. The Safecap platform uses logical invariants instead of temporal logic to encode the safety properties of a railway interlocking system. Failure to satisfy the properties through constraint solving yields logical inconsistencies which are hard to read and understand but the authors have overcome this challenge by illustrating them through animation; this is a significant achievement which makes the approach usable to an engineer. The models in the Safecap approach are developed through a series of Event-B refinements and capture a behavioural model of train movement, route reservation, point locking and route cancellation as well as a formal description of the scheme topology and control tables and release tables. The models are automatically generated from a graphical editor to aid productivity.

1.1 CSP||B approach

Our research also focuses on model based design and the definition of interlocking systems using the formal modelling language of CSP||B [12]. We have similar verification aims to that of the Safecap approach. However, our approach is to use ProB to model check CSP||B formal descriptions of interlocking systems against temporal logic safety properties. This means that we have also experienced the state space explosion problem. We have documented the limits of the models that we can verify in [10]. Thus, one of our research challenges is to find a solution to the state space explosion problem so that our CSP||B approach can viably be used to model and verify industrial railway interlocking systems; we elaborate on this in Section 1.2.

There are two main motivations for using CSP||B. Firstly, we want to define models that are readable by engineers and which reflect the information flow in railways. The following information flow is clearly separated in our models: the controller sending a request message to the interlocking to which the interlocking responds; the interlocking sends signalling information to the trains; and the trains inform the interlocking about their movements. The interlocking serves as the systems clock: messages can be exchanged once per cycle. We also want to decouple the detail of the topology of the track plan, control tables and release tables from the information flow since these details represent the state information of a railway system. Using the complementary notations of CSP and B facilitated this dual view, helping to create models made up of components that individually focused on information flow and logical aspects. Secondly, we wanted to use modelling languages that will also allow us to naturally extend our reasoning beyond that of safety properties. Since CSP [11] is a process algebra that has a timed extension (Timed CSP) we have been using this extended

notation to reason about capacity. Our initial capacity analysis results have also been achieved as part of the EPSRC Safecap project [5].

1.2 Modelling and Verification challenge

In our previous work we identified a decomposition approach which enables us to split up problems at the level of the application domain prior to formal modelling and verification using CSP||B [6]. The novelty of this decomposition is that it is defined in terms of a domain specific language (DSL) description. Hence, it is independent of CSP||B and has the potential for other modelling techniques to adopt it. Our decomposition technique, referred to as "*covering*", proves that the safety of several decomposed scheme plans implies the safety of the original scheme plan. Thus in [6] we demonstrated that CSP||B formal models of scheme plans which were previously not verifiable now are thanks to decomposition.

Our previous work on defining a DSL made several simplifying modelling assumptions that we relax here: we did not deal with crossings and were limited to track circuits and points that were uni-directional. Thus, in this paper we 1) summarise the current extensions to the DSL, 2) highlight one aspect of the resulting formal models when crossings are introduced and 3) outline the applicability of the decomposition technique in the bi-directional setting. Thus we are able to provide a rigorous technique that captures the main artifacts of a railway interlocking system.

The additional benefit of defining a DSL is that we can build graphical tools from the DSL and use them to generate the formal models automatically. Hence, we can repeatably generate CSP‖B formal models using our OnTrack model driven engineering tool [8].

The remainder of the paper is structured as follows: Section 2 summarises the extension to the DSL, Section 3 illustrates how the additional artifacts impacts on the formal model of interlocking and Section 4 highlights how a scheme plan is decomposed when it includes the additional scheme plan artifacts. Section 5 outlines the ongoing challenges for our CSP||B approach.

2 Domain Specific Model

A Domain Specification Language (DSL) description of the railway domain has long since been proposed by Bjørner [2]. Our DSL is inspired by Bjørner and differs from the one presented in [6] as we include the new artifacts mentioned in Section 1.2. We believe that a DSL which is independent of a formal method provides a basis for discussing concepts and principles without being tied to a particular formal method.

A railway network is provided by a scheme plan $SP = (Top, CT, RTs)$ which is comprised of a track plan Top defining its topology; a control table CT; and a set RTs of release tables. This paper focuses on the topology.

A track plan consists out of Units representing types of track, and Connectors whose elements serve as glue between units. There are different kinds of units.

We consider Track, Point and Crossing which are finite sets of tracks, points, and crossings respectively, and TerminalTrack $\subseteq$ Track. We let Unit = Track $\uplus$ Point $\uplus$ Crossing. At this point, one can extend our DSL by further types of unit.

A track, having two endpoints, is associated with two distinct connectors; a point, having three endpoints, is associated with three distinct connectors; and a crossing with four distinct connectors. We write $\mathit{connectors}(u)$ to denote the set of all connectors of a unit u. For a given a unit u, a Direction d is a pair $d = (c_1, c_2) \in \mathsf{Connector} \times \mathsf{Connector}$, where $c_1, c_2 \in \mathit{connectors}(u)$ and $c_1 \neq c_2$. A direction indicates that a train can travel on u from c_1 to c_2.

A track can be passed in one or two directions; terminal tracks have two directions; points and crossings have up to four directions. Points also have an orientation in the sense that movement between two specific connectors is excluded; similarly, crossings have an orientation in the sense that their four connectors, say $c_1, \ldots, c_4$, are organised into the two branches on which a train can travel, i.e., there are disjoint sets $L = \{c_1, c_2\}$ and $R = \{c_3, c_4\}$ such that the directions of a crossing are a subset of (c_1, c_2), (c_2, c_1), (c_3, c_4), or (c_4, c_3), i.e, either in L or in R. Points are dynamic entities which can change their position. There are two positions a point p can be in: *normal* and *reverse* where $\mathit{directions}(p) = \mathit{normal}(p) \uplus \mathit{reverse}(p)$.

The directions of a unit can be read as the "intended use" of the unit, which the signal engineer provides when designing a track plan. Given a direction $d = (c_1, c_2) \in \mathit{directions}(u)$ of a unit u, we write $\mathit{from}(d) = c_1$, $\mathit{to}(d) = c_2$.

A *path* $P = \langle (u_1, d_1), \ldots, (u_k, d_k) \rangle$, $k \geq 1$, in a railway topology is a non-empty sequence of units and their directions without direct repetitions: $\mathit{to}(d_i) = \mathit{from}(d_{i+1})$ and $u_i \neq u_{i+1}$ for all $1 \leq i < k$. As usual, $\mathit{hd}(P) = (u_1, d_1)$ and $\mathit{last}(P) = (u_k, d_k)$, and $u \in P$ if $u = u_i$ for some $1 \leq i \leq k$. When the connectors are clear, we also write $\langle u_1, \ldots, u_k \rangle$ for P.

We assume a set Signal of signals, along with a labelling function $\mathit{signalAt} : \mathsf{Signal} \to \mathsf{Track} \times \mathsf{Connector} \times \mathsf{Connector}$ indicating tracks at which signals are placed and the direction they are facing. Each track may be labelled by at most one signal (signals are not placed at a point or a crossing).

When track plans are decomposed open topologies are created. Therefore, a DSL needs a notion of entry and exit tracks. Units without predecessors are called entries, units without successors are called exits. We denote the set of entry and exit tracks as

$$\mathsf{Entry} = \{(t, d) \mid t \in \mathsf{Track} \wedge \mathit{predecessor}(t, d)) = \emptyset\}$$
$$\mathsf{Exit} = \{(t, d) \mid t \in \mathsf{Track} \wedge \mathit{successor}(t, d)) = \emptyset\}$$

where $\mathit{successor}(u, d) = \{(u', d') \mid \langle (u, d)(u', d') \rangle \textit{ is a path}\}$ and $\mathit{predecessor}(u, d) = \{(u', d') \mid \langle (u', d)'(u, d) \rangle \textit{ is a path}\}$.

We require that there is a signal at every entry track. Without such an entry signal, trains could unrestrictedly enter the scheme plan. This would cause collision on the successor of an entry track.

As we deal with open railway topologies, we need to give two different definitions of what a (topological) route is: the first definition caters for the case

in which the route is completely within the railway topology, while the second definition caters for the case in which a route ends at the border of the topology. A path $r = \langle (u_1, d_1), \ldots, (u_k, d_k) \rangle$ is a topological route if one of the following holds:

- there is a unit u_0 with direction d_0 such that

$$\langle (u_0, d_0), (u_1, d_1), \ldots, (u_k, d_k) \rangle$$

  is a path in which u_0 and u_{k-1} are labelled with signals but there are no signals on $u_1, \ldots, u_{k-2}$. In this case, u_k is called the *overlap* of r; or
- there are units u_0 and u_{k+1} and directions d_0 and d_{k+1} such that

$$\langle (u_0, d_0), (u_1, d_1), \ldots, (u_k, d_k), (u_{k+1}, d_{k+1}) \rangle$$

  is a path, u_0 is labelled with a signal, there are no signals on $u_1, \ldots, u_k$, and u_{k+1} is an exit track.

In both cases, we define $topoUnits(r) = \{(u_1, d_1), \ldots, (u_k, d_k)\}$ and $topoSignal(r) = s$ where $signalAt(s) = (u_0, d_0)$. Finally, we let TopoRoute denote the set of all topological routes in the railway topology, so that $topoUnits : \mathsf{TopoRoute} \rightarrow \wp(\mathsf{Unit} \times \mathsf{Direction})$ and $topoSignal : \mathsf{TopoRoute} \rightarrow \mathsf{Signal}$.

3 Modelling and Verification

In earlier sections we noted our aim of introducing crossings and bi-directional travel to our CSP‖B models. In [7] we presented how to capture bi-directional units. Here we illustrate how the inclusion of crossings in the topology of a scheme plan impacts on the CSP‖B modelling of the interlocking behaviour.

Additional constraints are needed in the formal model to control the access to units which are not explicit in *CTs* and *RTs* of a scheme plan. *CTs* include normal and reverse tables which govern the directions of points for particular routes and also clear tables which govern the units that need to be clear in order to grant a route request. The information in a *RT* is modelled as a function: $releaseTable \in \mathsf{Track} \leftrightarrow (\mathsf{Route} \times \mathsf{Point})$ that provides details of when a point can be released when a train occupies a particular track on a route. In a CSP‖B model the set of locked points at a given time is given by $currentPointLocks \in \mathsf{Route} \leftrightarrow \mathsf{Point}$, mapping route names to points. This set is updated as appropriate when a *move* event occurs, i.e., when a train moves from one unit to another.

When granting a route request that includes a crossing not only do we need to make sure that the track in the direction of travel through the crossing is clear (e.g. (c_2, c_1)), we also have to make sure that the other direction of travel is not being considered as part of another active route (c_3, c_4). We have modelled a crossing as just two tracks with directions and introduced the notion of locking all tracks on requested routes $currentRouteLocks \in \mathsf{Route} \leftrightarrow directedTrack$, where $directedTrack \in \mathsf{Track} \leftrightarrow \mathsf{Connector} \times \mathsf{Connector}$. Thus when a route request is made the *currentRouteLocks* is used in the encoding of a control condition, as

```
1  IF((signalStatus(signal(route)) = red) ∧
2     (clearTable(route) ⊆ emptyTracks)) ∧
3        ((clearTable(route) ◁ ran(currentRouteLocks)) ⊆
4           directedClearTable(route)) THEN ...
```

Fig. 1. part of control condition in route *request* event from *Interlocking* machine.

shown in Figure 1 using B notation. It ensures that a route request is granted only when the signal for the route is red (1), all the tracks on the route are empty (2) and there are no conflicting locks on the tracks on the route (3,4) (as well as no lock on points — omitted here for brevity).

The set *currentRouteLocks* is similarly updated when a *move* event occurs, but unlike *currentPointLocks* the relase can happen immediately after the train has passed over the locked unit (and thus the information is not required to be captures in a *RT*). Hence with the one additional locking model for routes the CSP‖B model extends to the safe inclusion of crossings.

4 Decomposition Strategy

In this section we describe a technique for decomposing a scheme plan with a set of smaller sub-scheme plans. The aim is that instead of having to verify the larger scheme plan we can perform a number of smaller verifications. We consider each unit in turn from the scheme plan and construct a zone of influence for that unit. This means that all the ways that a unit can be influenced is captured in the zone. Thus, the verification of each unit means that we examine the violation of the safety properties: collision, derailment and run-through on each unit within its zone of influence, i.e., a localised safety property. In [6] we proved that the safety of all the sub-scheme plans implies the safety of the original scheme plan. In this paper we introduce the extension of the decomposition technique to deal with bi-directional units. The natural consequence of bi-directional units is that the zones constructed as slightly larger than for those when the scheme plans were only made up of uni-directional units. Nonetheless, each zone has the potential to be significantly smaller than the whole scheme plan.

Let us define a localised scheme plan for a particular unit as follows: $SP_L = (Top_L, CT_L, RTs_L)$. The scheme plan SP_L will be used to investigate safety at units in $L \subseteq \mathsf{Unit} \setminus (\mathsf{Entry} \cup \mathsf{Exit})$.

In a first step, we consider all tracks over which a train can travel on the topology towards a unit in L. Figure 2 provides an illustration for all notions introduced below for a bidirectional track.

First, we give a construction that collects the tracks of L together with all units over which a train can travel on the topology towards a track in L:

$$Zone(L) = \{(u, d) \mid u \in \mathsf{Unit} \wedge d \in directions(u) \wedge \\ \exists \ \text{path } p : hd(p) \in \mathsf{Entry}, last(p) \in L, (u, d) \in p\}.$$

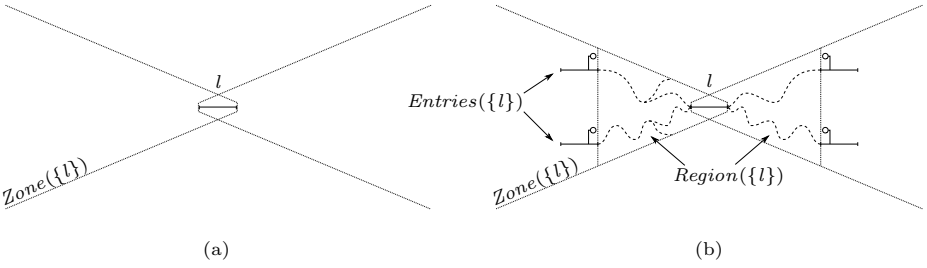

Fig. 2. Bi-directional influence region.

One can think of each element of L as the apex of two cones (one from each direction of travel). Then, we define the set of all topological routes that share a unit with L:

$$Routes(L) = \{r \in \mathsf{TopoRoute} \mid \exists (u, d) \in L : (u, d) \in r\}$$

The *Region* of L consists of those units which are on a route directly leading to L:

$$Region(L) = Zone(L) \cap \Big(\bigcup_{r \in Routes(L)} topoUnits(r) \Big)$$

Since we are dealing with open topologies we close the region by adding suitable entry and exit units:

$$\begin{aligned} Entries(L) &= \big(predecessor(Region(L)) \setminus Region(L)\big) \cap\ Zone(L) \\ Exits(L) \quad &= \{(u, d) \in successor(Region(L)) \setminus Region(L) \mid \exists\, r \in Routes(L) : \\ &\qquad hd(r) \in Entries(L) \wedge (u, d) \in r\} \end{aligned}$$

where the *successor* and *predecessor* functions are applied point-wise to the set. The *ClosedRegion* finally is defined as:

$$ClosedRegion(L) = Region(L) \cup Entries(L) \cup Exits(L).$$

With these definitions we are able to verify all closed regions of a scheme plan are collision free which implies collision freedom of the scheme plan.

5 Conclusions

In this paper we discussed how a DSL informs our CSP‖B models. A formal model forces clarity on the modelling assumptions that are made, e.g., points do not move under trains. One of the modelling challenges is to have confidence in the formal models that are defined and to ensure that the effort is not expended on writing the model but on its verification. Thus it is essential to be able to

automatically generate formal models from scheme plans that are drawn using graphical editors. Our OnTrack tool which generates CSP‖B models is one such example and is based on our DSL to ensure robustness. The other challenge is to ensure scalability of the verification approach. In the paper we provided an overview of our decomposition technique. It is important to have techniques that are not only rigorous but have the proofs behind the techniques to justify them. Our current work involves adapting the proofs in [6] to ensure that they remain applicable for bi-directional units. One of our future challenges is to adapt the CSP‖B architecture to ETCS Level 2 in the first instance.

Acknowledgement: The authors would like to thank S. Chadwick from Siemens Rail Automation for support and feedback during the EPSRC SafeCap project. Research partly funded by a Royal Academy of Engineering/Leverhulme Trust Senior Research Fellowship.

References

1. European Committee for Electrotechnical Standardiazation, CENELEC EN50128, Railway Applications - Communication, signalling and processing systems - Software for railway cotnrol and protection systems, 2011.
2. D. Bjørner. Dynamics of Railway Nets: On an Interface between Automatic Control and Software Engineering. In *CTS*. Elsevier, 2003.
3. A. Fantechi. Twenty-five years of formal methods and railways: What next? In *SEFM Workshops*, volume 8368 of *LNCS*, pages 167–183. Springer, 2013.
4. A. Iliasov, I. Lopatkin, and A. Romanovsky. The SafeCap platform for modelling railway safety and capacity. In *SAFECOMP*, volume 8153 of *LNCS*, 2013.
5. Y. Isobe, F. Moller, H. N. Nguyen, and M. Roggenbach. Safety and line capacity in railways - an approach in Timed CSP. In *IFM*, volume 7321 of *LNCS*, pages 54–68. Springer, 2012.
6. P. James, F. Moller, H. Nguyen, M. Roggenbach, S. Schneider, and H. Treharne. Techniques for modelling and verifying railway interlockings. *Int. J. Softw. Tools Technol. Transf.*, pages 1–27, 2014.
7. P. James, F. Moller, H. N. Nguyen, M. Roggenbach, S. Schneider, H. Treharne, M. Trumble, and D. M. Williams. Verification of scheme plans using CSP || B. In *SEFM Workshops*, volume 8368 of *LNCS*, pages 189–204. Springer, 2014.
8. P. James, M. Trumble, H. Treharne, M. Roggenbach, and S. Schneider. OnTrack: An open tooling environment for railway verification. In *NASA Formal Methods*, volume 7871 of *LNCS*, pages 435–440. Springer, 2013.
9. M. Leuschel and M. Butler. ProB: an automated analysis toolset for the B method. *Int. J. Softw. Tools Technol. Transf.*, 10(2):185–203, Feb. 2008.
10. F. Moller, H. N. Nguyen, M. Roggenbach, S. Schneider, and H. Treharne. Defining and model checking abstractions of complex railway models using CSP‖B. In *Haifa Verification Conference*, volume 7857 of *LNCS*, pages 193–208. Springer, 2012.
11. S. Schneider. *Concurrent and Real-time Systems: The CSP approach*. Wiley, 1999.
12. S. Schneider and H. Treharne. CSP theorems for communicating B machines. *Formal Asp. Comput.*, 17(4):390–422, 2005.
13. T. van Dijk, A. W. Laarman, and J. C. van de Pol. Multi-core and/or symbolic model checking. In *AVoCS 2012*, volume 53 of *Electronic Communications of the EASST*, pages 773:1–773:7, Berlin, 2012. EASST.