Verification of Scheme Plans using CSP||B

Philip James[1], Faron Moller[1], Hoang Nga Nguyen[3], Markus Roggenbach[1],
Steve Schneider[2], Helen Treharne[2], Matthew Trumble[2], and David Williams[4]

[1] Swansea University, UK
[2] University of Surrey, UK
[3] University of Nottingham, UK
[4] VU University Amsterdam, Netherland

Abstract. The paper presents a tool-supported approach to graphically editing scheme plans and their safety verification. The graphical tool is based on a Domain Specific Language which is used as the basis for transformation to a CSP||B formal model of a scheme plan. The models produced utilise a variety of abstraction techniques that make the analysis of large scale plans feasible. The techniques are applicable to other modelling languages besides CSP||B. We use the ProB tool to ensure the safety properties of collision, derailment and run-through freedom.

1 Introduction

In a series of papers [1–7] we have been developing a new modelling approach for railway interlockings. This work has been carried out in conjunction with railway engineers drawn from our industrial partner. By involving the railway engineers from Invensys, we benefit twofold: they provide realistic case studies, and they guide the modelling approach, ensuring that it is natural to the working engineer.

We base our approach on CSP||B [8], which combines event-based with state-based modelling. This reflects the double nature of railway systems, which involves events such as train movements and – in the interlocking – state based reasoning. The formal models are by design close to the domain models. To the domain expert, this provides traceability and ease of understanding. The validity of this claim was demonstrated in particular in [3] where a non-trivial case study – a complex double junction – was provided, a formal model of which was understandable and usable by our industrial partners.

In the UK, the development of interlockings follows prescribed processes from Railway Authorities such as the *Governance for Railway Investment Projects* (GRIP) process from Network Rail. In this process, the development of an interlocking consists of five phases where the first four phases are responsible for defining the track plan and determining routes to be used while, in the last phase, a contractor such as Invensys participates and is responsible for designing a control table for the track plan, implementing the interlocking and choosing appropriate track equipment. To this end, our paper offers a work flow which enables safety to be validated in each of these phases.

In [4, 5] we addressed how to *effectively* and *efficiently* verify safety properties within our CSP||B models. The properties of interest are collision, derailment and run-through freedom. To this end we developed a set of abstraction techniques for railway verification that allow the transformation of complex CSP||B models into less involved ones; we proved that these transformations are sound; and we demonstrated that they allow one to verify a variety of railway systems via model checking. The first set of abstractions reduces the number of trains that need to be considered in order to prove safety for an unbounded number of trains. Their correctness proof involves slicing of event traces. Essentially, these abstractions provide us with finite state models. The second set of abstractions simplifies the underlying track topology. Here, the correctness proof utilizes event abstraction specific to our application domain similar to the ones suggested by Winter in [9]. These abstractions make model checking faster.

Still present in our approach from the aforementioned papers was the need to write the formal models by hand. In [6] we described our OnTrack toolset[5], an open tool environment allowing graphical descriptions to be captured and supported by formal verification. This enables an engineer to visually represent the tracks and signals etc., within a railway network.

In this paper we continue the dissemination of our modelling approach which now also incorporates multi-directional tracks. We demonstrate that when changes are made to the models they are systematic and traceable; again this addition will be incorporated within our OnTrack tools.

The paper is organised as follows. In Section 2 we introduce our modelling language CSP||B so that we have the basis for discussing our workflow and provide examples. In Section 3 we introduce concepts in railway systems. In Section 4 we describe the workflow for our CSP||B modelling approach and summarise where the different abstraction techniques fit into the workflow. In Section 5 we introduce the modelling concepts of multi-directional travel and provide two illustrative examples. The application of our approach is presented in Section 6 via verification of our example scenarios. In Section 7 we put our work in the context of related approaches and finally conclude with future plans for the approach.

2 Background to CSP||B

The CSP||B approach [8] allows us to specify communicating systems using a combination of the B-Method [10] and the process algebra CSP (Communicating Sequential Processes) [11]. The overall specification of a combined communicating system comprises two separate specifications: one given by a number of CSP process descriptions and the other by a collection of B machines. Our aim when using B and CSP is to factor out as much of the "data-rich" aspects of a system as possible into B machines. The B machines in our CSP||B approach are classical B machines, which are components containing state and operations on that

[5] OnTrack available for download from http://www.csp-b.org.

state. The CSP||B theory [8] allows us to combine a number of CSP processes *Ps* in parallel with machines *Ms* to produce *Ps* ∥ *Ms* which is the parallel combination of all the controllers and all the underlying machines. Such a parallel composition is meaningful because a B machine is itself interpretable as a CSP process whose event-traces are the possible execution sequences of its operations. The invoking of an operation of a B machine outside its precondition within such a trace is defined as divergence [12]. Therefore, our notion of consistency is that a combined communicating system *Ps* ∥ *Ms* is *divergence-free.* We do not consider deadlock-freedom in this paper as it is concerned with liveness, and the focus of the paper is on safety.

A B MACHINE clause declares a machine and gives it a name. The VARIABLES of a B machine define its state. The INVARIANT of a B machine gives the type of the variables, and more generally it also contains any other constraints on the allowable machine states. There is an INITIALISATION which determines the initial state of the machine. The machine consists of a collection of OPERATIONS that query and modify the state. Besides this kind of machine we also define static B machines that provide only sets, constants and properties that do not change during the execution of the system.

The language we use to describe the CSP processes for B machines is as follows:

$$\begin{aligned} P ::= {} & e?x!y \rightarrow P(x) \;\mid\; P_1 \,\square\, P_2 \;\mid\; P_1 \sqcap P_2 \;\mid \\ & \textbf{if}\ \ b\ \ \textbf{then}\ \ P_1\ \ \textbf{else}\ \ P_2\ \textbf{end} \;\mid\; N(exp) \;\mid \\ & P_1 \parallel P_2 \;\mid\; P_1 \;{}_A\|_B\; P_2 \;\mid\; P_1 \,|||\, P_2 \end{aligned}$$

The process $e?x!y \rightarrow P(x)$ defines a channel communication where x represents all data variables on a channel, and y represents values being passed along a channel. Channel e is referred to as a *machine channel* as there is a corresponding operation in the controlled B machine with the signature $x \longleftarrow e(y)$. Therefore the input of the operation y corresponds to the output from the CSP, and the output x of the operation to the CSP input. Here we have simplified the communication to have one output and one input but in general there can be any number of inputs and outputs. The other CSP operators have the usual CSP semantics.

In this paper we omit a detail discussion of the semantic models used for reasoning of CSP||B models. In [5] we discuss that the traces models is enough to deal with the safety properties of railway interlockings.

3 Railway systems

Together with railway engineers we developed a common view on the information flow in railways. In physical terms a railway consists of, at least, four different components as illustrated in Figure 1:

- The *Controller* selects and releases routes for trains.

- The *Interlocking* serves as a safety mechanism with regards to the Controller and, in addition, controls and monitors the Track equipment. The
- *Track equipment* consists of elements such as signals, points, and track circuits. Signals can show the different aspects to indicate when trains can proceed; points can be in normal position (leading trains straight ahead) or in reverse position (leading trains to a different line) and track circuits detect if there is a train on a track.
- Finally, *Trains* have a driver who determines their behaviour.

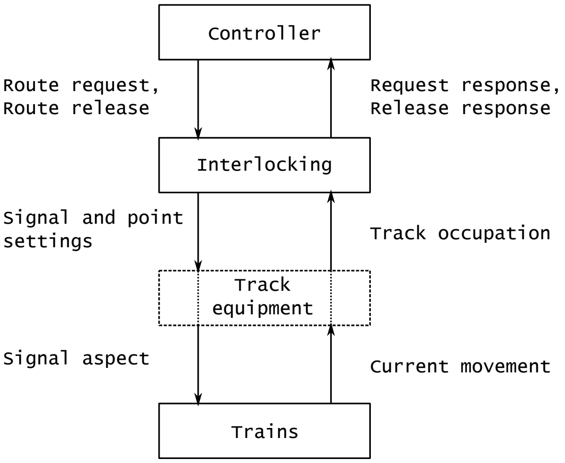


Fig. 1. Information flow.

For the purposes of modelling, we simplify the signals in railway systems to have only two aspects. We also make a further assumption that track equipment reacts instantly and is free of defects.

The information flow shown in Figure 1 is as follows: the controller sends a request message to the interlocking to which the interlocking responds; the interlocking sends signalling information to the track equipment and receives information from track sensors on whether a track element is occupied. The interlocking and the trains interact indirectly via the track equipment only. The interlocking serves as the systems clock: in a cycle the status of all the track sensors are read then the interlocking reacts to all of them with one change of state. Routes cannot be in conflict since requests to select and release routes are sequentialised. In our modelling we will abstract away from modelling the track equipment explicitly.

Each railway system is provided from the railway industry as a *scheme plan* which consists of a track plan (describing the topological relation between elements of the track equipment such as which tracks are connected and where signals are), a control table (determining how the interlocking of the railway system sets signals, moves points and lock points where, for each signal, there is one or more rows describing the condition under which the signal can show proceed) and a number of release tables (specifying when locks on points can be released). More details about control tables and release tables can be found in [3]. To this end, our task is to provide models which faithfully capture the behaviour associated with these railways systems.

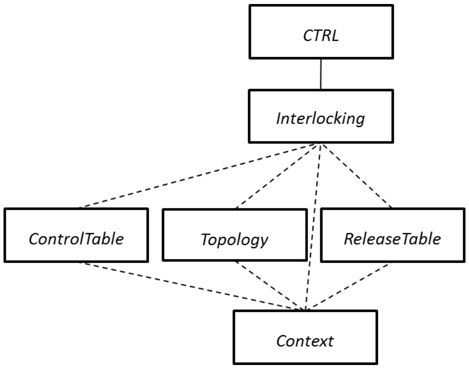


Fig. 2. CSP||B Architecture.

In this setting, we consider three safety properties:

1. collision-freedom excludes two trains occupying the same track;
2. runthrough-freedom says that whenever a train enters a point, the point is set to cater for this;
3. no-derailment says that whenever a train occupies a point, the point does not move.

Our modelling approach of railway systems in CSP||B, presented in [3], is restated in Figure 2. The centralised control logic is represented in the *Interlocking* machine, whereas the train behaviour is controlled by CSP processes defined in the *CTRL* script. These process and machine synchronise on common events. In the next sections we illustrate some aspects of the CSP processes and machines via examples [6] and focus on how multi-directional travel of trains on tracks is modelled.

4 Workflow

In this section, we present the workflow that we employ in our methodology in order to verify safety properties of railway systems. Figure 3 demonstrates the essential steps of the workflow which makes use of two tools: OnTrack [6]

[6] Examples available for download from http://www.csp-b.org.

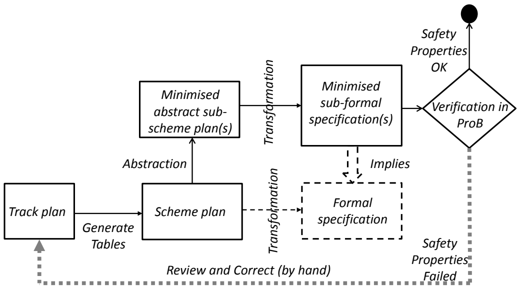
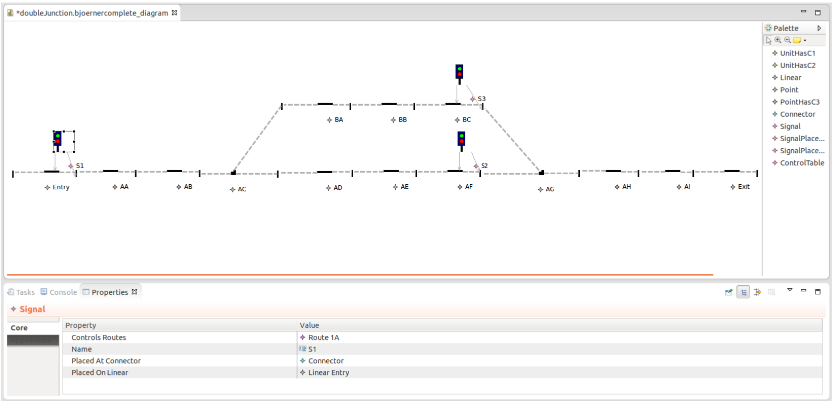


Fig. 3. CSP||B modelling and verification workflow.

and the ProB model checker [13]. Here, OnTrack is implemented in a typical EMF/GMF/Epsilon[7] architecture [14, 15] where a graphical editor realised in GMF is the front end for the user.

In this workflow, a user initially draws a *Track Plan* using the graphical front end in the OnTrack tool. Figure 4 shows the OnTrack editor that consists of a drawing canvas and a palette. Graphical elements from the palette can be positioned onto the drawing canvas.



Property	Value
Controls Routes	Route 1A
Name	S1
Placed At Connector	Connector
Placed On Linear	Linear Entry

Fig. 4. A screenshot of "OnTrack" modelling a track plan.

.

Then the first transformation, *Generate Tables* leads to a *Scheme Plan*, which is a track plan and its associated control and release tables. Track plans and scheme plans are models formulated relative to a modified version of the DSL developed by Bjørner [16]. The concepts of such a DSL can be easily captured within an ECORE meta-model which underlies our toolset. A small excerpt of

[7] EMF and GMF stand for Eclipse Modeling Framework and Graphical Modeling Project, respectively.

topological concepts within our meta-model is given in Figure 5. In this DSL,

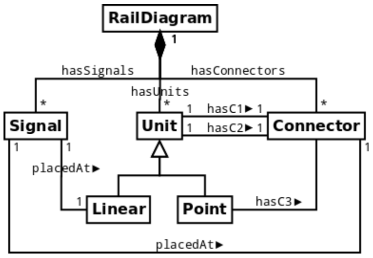


Fig. 5. Static concepts from Bjørner's DSL.

a *Railway Diagram* is built from *Units*, *Connectors* and *Signals*. *Units* come in two forms: *Linear* representing straight tracks, or *Point* representing a splitting track. All *Unit(s)* are attached together via *Connector(s)*. Finally, *Signals* can be placed on *Linear* units and at *Connectors*. To this end, the implementation of the GMF front-end for this meta-model involves selecting the concepts of the meta-model that should become graphical constructs within the editor and assigning graphical images to them.

A scheme plan is the basis for subsequent workflows that support its verification. Scheme plans can be captured as formal specifications. The simplest transformation, indicated by the *Transformation* dashed arrow, is to produce one *Formal specification* that is a faithful representation of the scheme plan. This transformation is a mapping from the railway DSL meta-model to the CSP||B meta-model and its subsequent representation as CSP||B script files that can be inputted into ProB. This automated transformation makes use of the finitisation theory in order to be able to perform bounded model checking of the formal specification [4, 7]. The finitisation theory allows us to reduce the problem of verifying of scheme plans for safety (i.e., freedom from collision, derailment, and run-through) for any number of trains to that of a two-train scenario.

Nonetheless, even when examining a reduced number of trains the formal specifications of realistic examples will inevitably contain too many states for safety analysis. Thus, our methodology enables us to carry out two forms of abstraction on a scheme plan:

(1) Covering Abstraction supports the decomposition of a scheme plan with a set of smaller sub-scheme plans. Any particular track in a scheme plan has a 'zone of influence': the other tracks which need to be considered to see what will happens on that track (e.g., when routes including it are enabled, when trains are approaching it, etc.). In particular, we only need to look at the zone of influence in order to see if a collision is possible on that track. To analyse if a collision, derailment or run-through is possible on that track, it is enough just to analyse the behaviour of trains within the zone of influence. We can do this for all the tracks, in each case just analysing for collisions, derailment or run-through within its zone of influence. This is called a covering. In general each zone of influence is much smaller than the overall track plan, so the analyses will be much quicker, and in practice can be done efficiently.

(2) Topological Abstraction supports the collapsing of tracks of a scheme plan to minimise the number of superfluous tracks in a plan, i.e., ones which do not impact on safety. Thus, for a particular track plan we take a sequence of tracks, and think of them as one single track. We do this for a number of sequences of tracks along the way. It is a topological abstraction if we can match moves around the original track plan with moves around the smaller one, so changes such as routes being enabled, points being released, trains being on particular routes, points being set, trains being at lights must still match for this collapsing to be a topological abstraction. If this is true then it means that we can analyse the behaviour of trains on the smaller scheme plan (which is easier because there are fewer positions to consider) and the results that we get will still be true for the original larger scheme plan.

We have proved the soundness of these abstractions in [4, 7]. In our methodology we first apply covering abstraction to generate sub-scheme plans and then apply topological abstraction to each of them. Using these abstractions we follow the *Abstraction* vertical workflow from the scheme plan to produce one or more *Minimised abstract sub-scheme plan(s).* One or more such plans may be produced because as we shall see in our examples, in Section 5, it may not always be possible to perform covering, and in which case the only abstraction that may yield a reduction in the number of tracks in the plan will be topological abstraction. Applying these abstractions is done at the DSL level and is independent of the formalism being used to represent the abstract CSP||B specification. Currently, the covering abstraction is not fully automated but is ongoing development work within the OnTrack tool.

Following abstraction (top left box on the diagram) the *Transformation* workflow, described earlier, can be applied to the minimised abstract sub-scheme plans to produce corresponding sub-formal specifications. All of the transformations that are performed by the OnTrack tool are validated via manual review. The verification of all of these sub-formal specifications implies the safety of the formal specification, as illustrated by the *Implies* arrow workflow; this result has been formally proved [4, 7].

Once OnTrack produces the sub-formal specifications they are all systematically verified using the ProB model checker to ensure that the models are collision- and derailment-free and contain no run-throughs. Successful checks verify that the safety properties hold for the particular scheme-plan. The workflow has the potential for round-trip engineering where the counter examples produced from unsuccessful model checking are automatically fed back into the OnTrack tool. This has not, as yet, been incorporated into the tool but it would provide an improved tool-supported workflow; this is illustrated using the dotted *Review and Correct* arrow on the workflow.

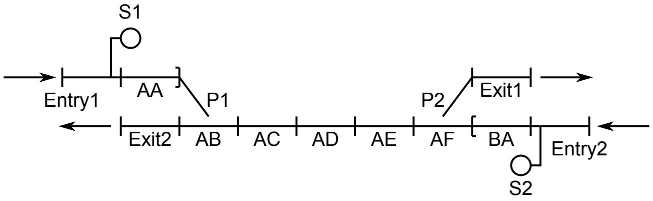


Fig. 6. Track plan for the tunnel example

5 Modelling of Multi-directional Examples of CSP||B railway models

In this section we provide details of the extension to our modelling approach in CSP||B which allows for multi-directional railway systems.

5.1 Tunnel Example

Consider the track plan in Figure 6 where tracks AB, AC and AD are bi-directional tracks. For route $R1$ associated with signal $S1$ their direction is left to right, whereas for route $R2$ associated with signal $S2$ their direction is right to left. The CSP process that controls the movement of trains is *TRAIN_CTRL*. Figure 7 illustrates the fragment of it controlling the movement of a train from a track that is neither an *exit* one or one which has a signal on it. The *move* event is parameterised with the train identifier t and its current position p. This event is a synchronisation with a *move* B operation which returns its new position *newp*. Therefore, moving from track AC to AD corresponds to the event $move.t.AC.AD$ for a particular train t.

```
TRAIN_CTRL(t, pos) = ...
□ pos ∉ EXIT ∧ pos ∉ SIGNALHOMES &
          move!t.pos?newp → TRAIN_CTRL(t, newp)
□ ...
```

Fig. 7. Fragment of the CSP control process for trains.

Note, there is no information in the CSP event that corresponds to the direction of travel. All this information is contained in the *Topology* machine and used in the *move* operation within the *Interlocking* machine. In the *Topology* machine there are three relations which define the direction of tracks. For example, the relation *direction* shown in Figure 8 shows that the model needs to contain details of the way tracks are connected together, and this is explicitly done via the notion of identified *connectors* — the glue between tracks and points.

```
$direction \in TRACK \leftrightarrow CONNECTOR * CONNECTOR \wedge$
  $direction = \{\ldots,$
              $AA \mapsto (C1, C2), \ldots,$ /* uni-directional tracks */
              $AC \mapsto (C3, C4), AC \mapsto (C4, C3), \ldots$ /* bi-directional tracks */ $\}$
```

Fig. 8. Fragment of the *direction* relation from *Topology*.

As we saw above the notion of a train's position in the CSP was captured using two parameters (t, pos). In the INVARIANT of the *Interlocking* machine a similarly named function *pos* also includes information about the connectors, as shown in Figure 9. In its INITIALISATION $pos := \emptyset$ since there are no trains on the tracks. The *move* operation updates the track and connectors related to train t in *pos* each time the train moves. (In earlier papers, e.g., [3], *pos* was simply a partial function between trains and tracks and *direction* was not required.)

```
$pos \in TRAIN \rightarrowtail\!\!\!\!\!\!+ \; ALLTRACK *$
          $(ALLCONNECTOR * ALLCONNECTOR)$
```

Fig. 9. *pos* function from *Interlocking*.

In addition to B operations which define the behaviour of movement, granting and releasing of route requests the OnTrack tool automatically produces B operations to support the verification of safety properties. Three B operations are produced, *collision*, *derailment* and *run-through*. Collision is encoded as follows:

```
$collision =$
SELECT
  $\exists\, t_1, t_2 \in TRAIN \wedge t_1 \neq t_2 \wedge$
  $t1 \in dom(pos) \wedge t2 \in dom(pos)$
  $(dom(pos(t_1)) - (EXIT \cup ENTRY)) \cap$
  $(dom(pos(t_2)) - (EXIT \cup ENTRY)) = \emptyset$
 THEN $skip$
 END;
```

Here collision is detected when two different trains t_1 and t_2 occupy the same track segment (different from the *EXIT* and *ENTRY* tracks). The collision condition will be enabled when the two trains are at the same position.

Collision freedom can then be established by model checking the validity of the following CTL formula:

$$AG(not(e(collision)))$$

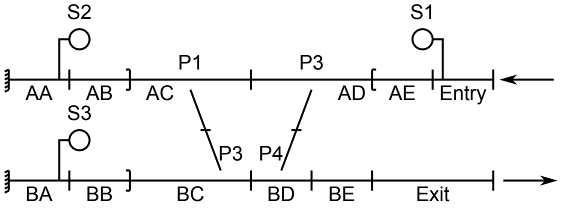


Fig. 10. Track plan for the buffer example

This formula is false if *collision* is enabled. In the CTL variant of ProB AG, stands for "on all paths it is globally true that", and $e(a)$ stands for "event a is enabled".

5.2 Buffer Example

Our next example is also multi-directional as shown in Figure 10. Interestingly, track BC has three directions, i.e., $\{BC\} \lhd \mathit{direction} = \{(C12, C11), (C11, C12), (C7, C12)\}$, where $C7$ is the connector between tracks AC and BC, $C11$ is between BB and BC, and $C12$ is between BC and BD, respectively.

It also serves to illustrate how additional complexity can easily be traced within a formal specification. We model the behaviour of buffers, i.e., tracks where trains can turn around; in our example the buffers are AA and BA. Two routes are associated with signal $S1$, i.e., route $R1A$ is associated with AE, AD, AC, AB and AA and $R1B$ is associated with AE, AD, BD, BC, BB and BA. Thus, when a train is on route $R1A$ and is on track AA it can change direction and then follow route $R2$ which is associated with signal $S2$. Similarly, for route $R3$ associated with signal $S3$.

This additional behaviour requires three additions to the CSP processes and B machines:

- The additional definition of $BUFFER = \{AA, BA\}$ in the *Context* machine and similarly in the CSP types.
- A new *changeDirection* operation as shown in Figure 11. The purpose of this operation is to simply modify the direction of the connectors for the particular buffer track on which the train t currently resides. Hence, changing the direction of train t on track AA means changing the maplet $(t \mapsto AA, (C1, C0))$ to $(t \mapsto AA, (C0, C1))$ within the *pos* function. This means that we can leave the *move* operation unchanged.
- Within the CSP, rather than disturb the existing processes, we define a new process, $BUFFER_P(b, t)$ in Figure 12 which defines that after a train moves onto the buffer track b it must change direction before it can move off it. In the model there will be a separate buffer process for each buffer and they are independent of each other. These new processes are combined to reformulate the overall CSP processes contained in the $CTRL$ script.

```
changeDirection(t, currp) =
PRE t ∈ TRAIN ∧ t ∈ dom(pos)∧
      {currp} = dom({pos(t)}) ∧ currp ∈ BUFFER
THEN
  movedPoints := {} ||
  LET(track, d) BE (track, d) = pos(t) IN
    LET(d1, d2) BE (d1, d2) = d IN
      pos(t) := (track, (d2, d1))
    END
  END
END;
```

Fig. 11. *changeDirection* method from *Interlocking*.

```
BUFFER_P(b, t) = move!t?p!b → changeDirection.t.b
                              → move.t.b?newp → BUFFER_P(b, t)
```

Fig. 12. $BUFFER_P$ process in *CTRL*.

6 Experimental result

In this section, we present the experiment result when verifying safety properties of the tunnel and buffer examples presented in the previous section. These experiments are carried out by following the verification workflow defined in Section 4.

In order to verify the tunnel example, an engineer first uses the OnTrack tool to draw the track plan as depicted on Figure 6. Then the safety properties of the example can be verified by loading the formal specification produced by OnTrack into the ProB tool and performing this check. Here, a total of 1,516 distinct states were examined in order to determine that no collision was possible. Our methodology currently requires us to do this loading by hand but automating this as a batch process for all the safety properties could easily be done.

Similarly, verification of the buffer example can be carried out by using the OnTrack tool to draw the track plan as depicted on Figure 10. The state space of the formal specification produced by OnTrack required by ProB to model check the safety properties for the formal specification of the Buffer example was 18,510 states, significantly more than in the tunnel example. In Section 4 we noted that it may not always be feasible to model check a complex scenario but our methodology supports the systematic generation of all the sub-scheme plans for a particular scheme plan. The track plan for one of the sub-scheme plans of the buffer example is shown in Figure 13. It illustrates the plan for the track AC constructed using the covering abstraction. The highlights from this plan are as follows:

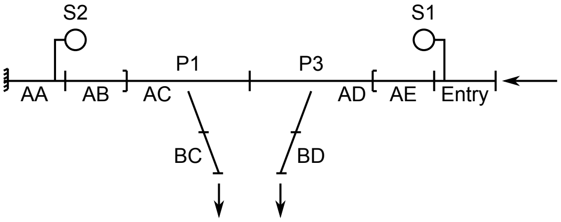


Fig. 13. Sub-track plan for track AC of the buffer example

- The point BC in the overall buffer example can now be considered as an *exit* track and after which we do not need to consider the behaviour of subsequent linear tracks and points. The reason being is that all that needs to be captured is what happens to the state when a train moves off the point AC and that this can be represented using a simple linear track rather than a point.
- The point BD is similarly converted to an *exit* track.
- The current version of our covering technique has not considered the impact of buffers on the abstraction of the scheme plans. Therefore, we must include them in the zone of influence. Therefore, both tracks AA and AB retain their bi-directional properties in the sub-scheme plan. We shall of course examine in future work whether such tracks can be further reduced.
- Notice also that we need not consider the path along *Entry*, AE, AD, BD, BC, BB, BA because it does not belong to the zone of influence as it does not contain the track AC, but of course *Entry*, AE and AD are included because they are on the normal route $R1A$ and BD is included for the above reason.

Running the formal specification of the sub-scheme plan for AC through ProB gives a state space of 3,995 compared to 18,510 states for the full specification. We have also verified that the three important safety properties hold for this sub-scheme plan. Methodologically, we would then be required to run all the sub-scheme plans through ProB and by appealing to our theoretical results we would conclude that the overall buffer example from Figure 10 preserves the safety properties.

7 Related work

The railway interlocking problem has long been studied by the Formal Methods community, and our work builds upon prior approaches to the modelling and verification of railways. Prominent studies from the B community include [17, 18] whilst [19, 20] are classical contributions from process algebra and [21] uses techniques from Algebraic Specification. On a lower abstraction layer, [22–25] verify the safety of interlocking programs with logical approaches.

Our modelling is most related to Winter's uni-directional approach in CSP [26] and Abrial's bi-directional modelling in Event-B [27], which however excludes that train can turn around at end stations. Winter [26] presents a generic, event-based railway model in CSP as well as generic formulations of two safety properties: CollisionFreedom and NoMovingPoints. Overall, this results in a generic architecture and a natural representation of two safety properties. Traceability, however, is limited. There are relations in the model which are *derived* from the control table. For example, the driving rule "trains stop at a red signal" is distributed over different parts of the model: it is a consequence of the fact that (1) the event "move to the first track protected by a signal" belongs to a specific synchronziation set and (2) a red signal does not offer this event. Purely event-based modelling leads to such decentralized control. Consequently, the model has no interlocking cycle. Chapter 17 of the book by Abrial [27] gives an excellent detailed description and analysis of the railway domain, deriving a total of 39 different requirements. The modelling approach is generic, even though no concrete model is proven to be correct. Traceability in a tower of specifications can be complex for various reasons. For instance, a requirement can be the consequence of invariants from different levels. The relation between intended properties and the model remains an informal one. This is in contrast to other approaches (including Winter's and our own) which directly represent the intended property in the formal world and then prove that the modelled property is a mathematical consequence of the formal model. Furthermore, the approach is monolithic: behaviour is not attached to different entities to which they relate.

To put our work into context we must first clarify that railway verification falls into two categories: the verification of railway designs prior to their implementation and the verification of the implementation descriptions themselves. Our work is in the first area. A comparison using different model checkers in the analysis of control tables has been conducted by Ferrari *et al.* [28] and falls into the first category. Winter in a recent paper [29] considers different optimising strategies for model checking using NuSMV and demonstrates the efficiency of their approach on very large models. These analyses also fall into the first category but the models are flat in structure compared to our models as they are defined in terms of boolean equations and do not focus on providing behavioural models. The analysis of interlocking tables (cf. control tables) by Haxthausen [30] also falls into the first category and is supported by automated tools that generate the models. Cimatti *et al.* [25] also have had considerable success using NuSMV but their analysis is focussed on the implementation descriptions.

8 Conclusion

In this paper we provided an overview of our methodology that uses the OnTrack tool to provide a graphical front-end for the automatic generation of formal specifications. The formal specifications are then separately model checked using the ProB tool. We described the architecture of a CSP||B formal specification of a scheme plan giving details of the new aspects that allow the modelling of multi-directional travel. We appreciate the absolute necessity to include these aspects in our CSP||B formal specifications and recognise that the majority of the related work includes such detail, for example [30]. Our aim by demonstrating its inclusion incrementally was to show the robustness of the CSP||B architecture and the ease by which new modelling aspects can be included. Similarly, additional development of the OnTrack tool-support can also be achieved incrementally. We are currently completing the implementation of the covering abstractions and the integration of the output from ProB model checking with OnTrack in order to provide round-trip engineering to the graphical editor. This will mean that the engineer is not required to manipulate the formal specifications when safety properties are violated. Instead, the engineer will be able to change a graphical scheme plan, re-generate the formal specifications and re-run the model checking in order to verify that the amended scheme plan preserves safety (i.e., freedom from collision derailment and run-through).

Heitmeyer in [31] discusses the importance of complete abstractions. Our abstractions are sound. It is future theoretical work to investigate if completeness can be established. Furthermore, we also would like to extend our methodology so that capacity and safety of large-scale railway systems can be studied simultaneously. One way to obtain this goal is to combine our modelling approach with others which take capacity into account such as presented in [32].

Acknowledgment

Thanks to S. Chadwick and D. Taylor from the company Invensys Rail for their support and encouraging feedback.

References

1. Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Combining event-based and state-based modelling for railway verification. Technical Report CS-12-02, University of Surrey (2012)
2. Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Using ProB and CSP||B for railway modelling. In: Proceedings of IFM'12 and ABZ 2012 Posters and Tool demos session. (2012) 31–35
3. Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Railway modelling in CSP||B: The double junction case study. Electronic Communications of the EASST **53** (2012)
4. Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Defining and model checking abstractions of complex railway models using CSP||B. In: Proceedings of HVC'12. LNCS 7857, Springer (2013)
5. James, P., Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: On modelling and verifying railway interlockings: Tracking train lengths. Technical Report CS-13-03, University of Surrey (2013)
6. James, P., Trumble, M., Treharne, H., Roggenbach, M., Schneider, S.: Ontrack: An open tooling environment for railway verification. In: NASA Formal Methods. LNCS 7871, Springer (2013) 435–440
7. James, P., Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Techniques for modelling and verifying railway interlockings. STTT to appear.

8. Schneider, S., Treharne, H.: CSP theorems for communicating B machines. Formal Asp. Comput. **17**(4) (2005) 390–422
9. Winter, K., Robinson, N.: Modelling large railway interlockings and model checking small ones. In: Proceedings of the 26th Australasian computer science conference-Volume 16, Australian Computer Society, Inc. (2003) 309–316
10. Abrial, J.R.: The B-Book: Assigning Programs to Meanings. CUP (1996)
11. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1985)
12. Morgan, C.: Of wp and CSP. Beauty is our business: a birthday salute to E. W. Dijkstra (1990) 319–326
13. ProB: The ProB animator and model checker (ProB 1.3.6-final). `http://www.stups.uni-duesseldorf.de/ProB` Accessed: 01/05/2013.
14. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional (2009)
15. Kolovos, D., Rose, L., Paige, R., García-Domínguez, A.: The Epsilon Book. The Eclipse Foundation (2012)
16. Bjørner, D.: Formal Software Techniques for Railway Systems. In: CTS2000. (2000)
17. Leuschel, M., Falampin, J., Fritz, F., Plagge, D.: Automated property verification for large scale B models with ProB. Formal Asp. Comput. **23**(6) (2011) 683–709
18. Sabatier, D., Burdy, L., Requet, A., Guéry, J.: Formal proofs for the NYCT line 7 (flushing) modernization project. In: ABZ. (2012) 369–372
19. Simpson, A., Woodcock, J., Davies, J.: The mechanical verification of solid-state interlocking geographic data. In: Formal Methods Pacific 97, Springer (1997)
20. Morley, M.J.: Safety in railway signalling data: A behavioural analysis. In: 6th International Workshop on HOLTPA, Springer (1993) 464–474
21. Haxthausen, A.E., Peleska, J.: Formal development and verification of a distributed railway control system. IEEE Trans. Software Eng. **26**(8) (2000) 687–701
22. Ferrari, A., Magnani, G., Grasso, D., Fantechi, A.: Model checking interlocking control tables. FORMS/FORMAT 2010 (2011) 107–115
23. Kanso, K., Moller, F., Setzer, A.: Automated verification of signalling principles in railway interlockings. ENTCS **250** (2009)
24. James, P., Roggenbach, M.: Automatically verifying railway interlockings using SAT-based model checking. Electronic Communications of the EASST **35** (2010)
25. Cimatti, A., Corvino, R., Lazzaro, A., Narasamdya, I., Rizzo, T., Roveri, M., Sanseviero, A., Tchaltsev, A.: Formal verification and validation of ERTMS industrial railway train spacing system. In: CAV, Springer (2012) 378–393
26. Winter, K.: Model checking railway interlocking systems. Australian Computer Science Communications **24**(1) (2002)
27. Abrial, J.R.: Modeling in Event-B. Cambridge University Press (2010)
28. Ferrari, A., Magnani, G., Grasso, D., Fantechi, A.: Model checking interlocking control tables. In: FORMS/FORMAT. (2010) 107–115
29. Winter, K.: Optimising ordering strategies for symbolic model checking of railway interlockings. In: ISoLA (2). Volume 7610 of Lecture Notes in Computer Science. (2012) 246–260
30. Haxthausen, A.: Automated generation of safety requirements from railway interlocking tables. In: ISoLA (2). Volume 7610 of Lecture Notes in Computer Science. (2012) 261–275
31. Heitmeyer, C.L., Kirby, J., Labaw, B.G., Archer, M., Bharadwaj, R.: Using abstraction and model checking to detect safety violations in requirements specifications. IEEE Trans. Software Eng. **24**(11) (1998) 927–948
32. Isobe, Y., Moller, F., Nguyen, H.N., Roggenbach, M.: Safety and line capacity in railways - an approach in Timed CSP. In: IFM. (2012) 54–68