COMPUTING SCIENCE

Practical Formal Methods in Railways - The SafeCap Approach

Alexei Iliasov, Ilya Lopatkin and Alexander Romanovsky

TECHNICAL REPORT SERIES

No. CS-TR-1404

January 2014

Practical Formal Methods in Railways - The SafeCap Approach

A. Iliasov, I. Lopatkin and A. Romanovsky

Abstract

This paper presents the SafeCap Platform approach to the verification of railway safety properties. We discuss how the hierarchy of formal theories is used to capture the railway domain and interface with verification tools; we explain the contribution of each individual theory to the overall task of safety verification and capacity assessment. Finally, we briefly relate our experience of using two independent verification chains to validate concrete track layouts and control tables against the SafeCap safety theories.

Bibliographical details

ILIASOV, A., LOPATKIN, I., ROMANOVSKY A.

Practical Formal Methods in Railways - The SafeCap Approach
[By] A. Iliasov, I. Lopatkin and A. Romanovsky

Newcastle upon Tyne: Newcastle University: Computing Science, 2014.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1404)

Added entries

NEWCASTLE UNIVERSITY

Computing Science. Technical Report Series. CS-TR-1404

Abstract

This paper presents the SafeCap Platform approach to the verification of railway safety properties. We discuss how the hierarchy of formal theories is used to capture the railway domain and interface with verification tools; we explain the contribution of each individual theory to the overall task of safety verification and capacity assessment. Finally, we briefly relate our experience of using two independent verification chains to validate concrete track layouts and control tables against the SafeCap safety theories.

About the authors

Alexei Iliasov is a Researcher Associate at the School of Computing Science of Newcastle University, Newcastle-upon-Tyne, UK. He got his PhD in Computer Science in 2008 in the area of modelling artefacts reuse in formal developments. His research interests include agent systems, formal methods for software engineering and tools and environments supporting modelling and proof.

Ilya Lopatkin was awarded his PhD degree by Newcastle University in 2012. During his PhD study he contributed to the PF7 DEPLOY Integrated Project. In 2012-2014 he was an RA on the EPSRC/RSSB SafeCap and the EPSRC SafeCap Impact Projects.

Alexander (Sascha) Romanovsky is a Professor in the Centre for Software and Reliability, Newcastle University. His main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system structuring and verification of fault tolerance. He received a PhD degree in Computer Science from St. Petersburg State Technical University and has worked as a visiting researcher at ABB Ltd Computer Architecture Lab Research Center, Switzerland and at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993 he became a postdoctoral fellow in Newcastle University, and worked on the ESPRIT projects on Predictable Dependable Computing Systems (PDCS), Design for Validation (DeVa) and on UK-funded projects on the Diversity, both in Safety Critical Software using Off-the-Shelf components. He was a member of the executive board of EU Dependable Systems of Systems (DSoS) Project, and between 2004 and 2012 headed projects on the development of a Rigorous Open Development Environment for Complex Systems (RODIN), and latterly was coordinator of the major FP7 Integrated Project on Industrial Deployment of System Engineering Methods Providing High Dependability and Productivity (DEPLOY). He now leads work on fault tolerance in Systems of Systems within the COMPASS project and is Principal Investigator of Newcastle's Platform Grant on Trustworthy Ambient Systems.

Suggested keywords

FORMAL METHODS
VERIFICATION
RAILWAY LAYOUTS
SAFETY
TOOLS
FORMAL THEORIES

Practical formal methods in railways - the SafeCap approach

Alexei Iliasov, Ilya Lopatkin, Alexander Romanovsky

Newcastle University, UK

Abstract. This paper presents the SafeCap Platform approach to the verification of railway safety properties. We discuss how the hierarchy of formal theories is used to capture the railway domain and interface with verification tools; we explain the contribution of each individual theory to the overall task of safety verification and capacity assessment. Finally, we briefly relate our experience of using two independent verification chains to validate concrete track layouts and control tables against the SafeCap safety theories.

1 Introduction

Ensuring and demonstrating railway system dependability is crucial for the way our society operates. Formal methods have been successfully used in developing various railway control systems. The best-known examples include the use of the B Method [1] for designing various metro and suburban lines, and airport shuttles all over the world [5, 6]. The formal methods here are used to trace the requirements to system models and to ensure and demonstrate the system safety. Our work builds on this work.

The SafeCap Platform [17] is an integrated modelling environment aimed at a railway signalling engineer. Its focus is the microscopic analysis of a railway node (station, marshalling yard, complex junction) with the purpose of investigation, with the maximum level of detail, the performance of a node under various service patterns, train types and stability margins. Central to achieving this goal is an efficient and scalable approach to the verification of operational safety. Indeed, any change in track layout, train detection circuit boundary and control rule potentially renders a node unsafe. It is our goal to address the highly challenging verification problem in such a way that it does not unnecessarily distract an engineer from the primary goal of optimization. This means a push-button approach where no expertise in verification technology is required and a feedback is presented promptly and in a friendly manner.

We continue with the discussion of principal problems and objectives of railway signalling verification. Within the hierarchy defined by Fokkink and Hollingshead [20], we focus exclusively on the middle interlocking layer leaving out details of lower layer of physical equipment functioning and upper layer of railway logics and exploitation largely out of the view. We identify five kinds of railway safety verification concerns.

Fig. 1. A simple junction and an excerpt from its control table.

The first three are the fundamental safety properties that may be reasoned about at a specification level abstracting away from minute details of physical track topology and the setting in which the track is laid. The remaining two require consideration of concrete track geometry, topography, train exploitation characteristics and prospective service requirements.

A schema must be free from collisions. A collision happens when two trains occupy the same part of a track. Reasoning about collisions must take into account concrete topology, requires an explicit train notion, the definition of laws of train movement and assumptions about train driver (either human or automatic) behaviour. Note that if train drivers choose to ignore whatever means of indication of track occupation states are available to them (e.g., track side signals) there is nothing preventing two trains from colliding. Hence, the absence of collisions is ensured by demonstrating the compatibility of specific topology, signalling and certain driving rules.

The basic safety mechanism is that of route locking and holding. A train is given permission to enter an area of a railway once there is a continuous and safe path through this area assigned exclusively to the this train. Such a path is normally called a *route* and is delineated by *signals* - either physical track-side signals with lamps or conceptual signals displayed to a driver via a computer screen. Two-aspect signals (red/green or stop/proceed) are positioned at the maximum braking distance from each other and this defines the smallest train separation. Three and four aspect signalling allow trains to come closer by advising drivers on the safe speed and the extent of free track available in front.

In Fig. 1, one route example is route **S10.S12** between signals **S10** and **S12**. This route consists of two train detection circuits: **AB** and **AC**. A train detection circuit is a part of a railway (a sub-graph in an abstract topology) with some equipment capable of reporting the presence or absence of a train in this part. An additional circuit **AD** is also in the dependency of this route playing the role of

an *overlap*. An overlap is an extra part of track reserved together with a route to protect a train running slightly past a red signal (when, for instance, rail adhesion is lower than normal and a driver misjudges the stopping distance); alternatively, it is an area to automatically engage train brakes should a driver fail to notice a red signal (this is generally known as an automated train protection (ATP) system).

When a route is locked, all the movable equipment such as *points* or level crossings must be set and detected in a position that would let a train safely travel on its desired route. They must remain locked in such a state until the train passage is positively confirmed.

A schema must be free from derailments. A derailment may happen when a train moves over a point that is not set in any specific direction and thus may move under a train. To avoid this, a point must be positively confirmed to be *locked* before a train may travel over it. In a control table one writes a condition defining when a point reconfiguration may happen. For the example in Fig. 1 such a rule requires that circuit **AD** is clear.

Another reason for derailment is driving a train through a curve at an unsafe speed. As a train goes over a curve, the combination of gravitational, centripetal and centrifugal forces exerts a rolling force on train carriages and a substantial lateral force on rails. This effect can be mitigated by track canting although no single canting is a perfect fit for all train types. Hence, enforcing a safe speed limit before a train enters a curved track area is an essential safety consideration. There are several ways of doing this. One is a static speed limit. This can be a sign board warning a driver (circle with number 30 over circuit **AD** in Fig. 1) or an electronic signal sent to an on-board computer. A speed restriction may be also enforced by signalling: a signal does not switch into a permissive aspect until a train is detected to occupy some preceding detection circuit for a duration time. A combination of such time duration and track length gives an upper train speed limit. Example in Fig. 1 uses this technique for route **S12_S24**: the control table mandates that a train travelling over the route occupies **AC** for at least 15 seconds.

Flank protection Consider the example in Fig. 1 and assume some train (2) approaching to and stopping at signal **S40** (going right to left) while some other train (1) travels over route **S12_S14**; also, assume that there is no overlap track for route **S40_S42**. Should train 2 fail to stop at the signal it might roll into the path or hit the side of train 1. Another common scenario is having no signalling in operation at siding **BE**, **BF**, **BG**. A train approaching from a siding would be operated on sight or using less restrictive shunting signalling. If the siding goes up a slope there is an additional danger of carriages rolling freely onto the main line **A_B**. Avoidance of such situations is generally known as *flank protection*. Defining route overlaps (e.g., requiring **AD** to be clear and point **P** set in reverse) would generally provide a sufficient safety margin albeit at a cost of lower performance and extra signalling equipment. An alternative is to set points in such a way that any interfering traffic would be diverted away or even derailed using *trap points*. In our example, we protect main line route **S12_S14** by

setting *flanking point* T to normal (straight) direction to divert any interfering traffic coming from the siding.

Physical layout properties. A range of properties pertinent to safety requires analysis of land topography over which track is build. As one example, it must be ensured that physical signals have certain minimum sighting distance giving a sufficient time for a driver to react. Sometimes tracks are so close together that carriages of a train going through a curve may come into a contact with carriages of a train located on a parallel track. A signalling engineer must identify and protect such areas (known as *fouling points*) via signalling rules. Further examples include gradients at stopping points (e.g., signals) that may be unsafe for heavy trains, parts of track susceptible to landslips, derbies on the track due to nearby trees and overpasses, and so on. An important consideration is the spacing between signals and speed restriction signs: it must be possible for all trains to brake within the given limits to meet signal or speed limit restriction. Signal positioning and speed restriction would be wastefully conservative if one does not consider a specific properties of traffic, in particular train acceleration and braking performance.

Quality of service. It is never sufficient to consider the safety aspect of a railway in isolation from its performance. Indeed, setting all signals permanently to red (stop) state trivially satisfies all the safety concerns discussed above. As a less extreme example, there could be a signalling mistake preventing or hindering train progress but not violating safety properties. Typically, when signalling a station or a junction, an engineer would have access to a provisional time table. A time table defines traffic class and station calling and dwelling times. It must be ensured that signalling is able to accommodate such traffic with some extra margin for unaccounted or delayed traffic. Simulation of train runs is the common way to check quality of service requirements.

2 Safety verification

In this section we discuss requirements to verification of railway signalling, in particular verification of control tables, and characterise major approaches used in the industry and proposed by academia. We start with the summary of *aspects* of verification - the prominent traits we believe to be relevant to industrial adoption. Our focus is largely on the application of formal verification to railway signalling.

2.1 Aspects of verification

Rigour A good verification technique is both exhaustive and thorough; this should be a natural consequence of innate properties of an applied technique rather than achieved through measures of extra diligence or man power. Clearly, a truly rigorous techniques must be based on an objective, formal handling of safety-related artefacts expressed in a mathematical notation. The results achieved with a rigorous technique must be readily reproducible from the record

of verification activity: a property highly desirable for the certification of signalling solutions.

Productivity It is not enough to simply offer a rigorous techniques; there must be means available and accessible to an engineer to apply the technique with some definite prospect of success within a given lapse of time. It means that a technique must provide automation for most of the routine activities and allow an engineer to assess the problem at differing levels of detail. A good presentation of railway artefacts and verification concerns is also an important issue.

Expertise Signalling engineers are expected to be experts in railway signalling but are rarely also experts in formal verification. Conversely, formal method experts are rarely proficient in the signalling domain. Hence, a technique levying an unlikely blend of expertise - i.e., the knowledge of both formal modelling and railways - would be costly to apply in practice. Lowering of expertise requirements is always welcome as it correlates with the reduction of user involvement and increase in productivity.

Scalability Safety verification is a pressing issue for truly large and complex stations and junctions. Small-scale signalling may be confidently executed by an experienced engineer without need for any advanced modelling and verification techniques. A large layout, however, is beyond mental capacity of a human being and has to be processed in a piece-wise manner greatly increasing the chance of both oversights and overly conservative decisions. Therefore, to be truly useful, a verification technique must be able to scale to the level of a large station or an area spanning several stations.

Expressiveness In Section 1 we have presented the main classes of verification conditions. A perfect verification techniques would address all of them in a satisfactory manner. For this, the notation employed to express verification conditions must be rich and flexible enough to capture verification goals without undermining legibility and, by extension, productivity. Automation tools must also be capable of dealing with the complete notation, not just a subset of it.

Feedback Although it is valuable to know whether a control table and layout are safe, it is even more valuable to obtain an explanation of why a given combination of a control table and layout are not safe and what steps should be taken to make it safe. Thus, the value of a verification technique is also determined by the form and kind of feedback a verification activity presents to an engineer. A feedback in the terms of logical constructs obtained through a mechanical translation of a railway model into a formal modelling notation is useless in a real life scenario. A simple yes or no is not good enough either as it leaves an engineer in the dark about the likely source of a problem. An ideal feedback would narrowly identify a responsible part and step an engineer through events and actions leading to a concrete manifestation of safety violation.

2.2 Techniques

In this section we discuss main approaches to signalling verification. In practice, several techniques may be combined although this has unclear implications on the overall rigour.

Manual review Just as compilation of control tables is often a manual process, verification may also be accomplished via a carefully set up but otherwise manual review procedure. In most cases, to facilitate legibility, control tables are written in a highly structured tabular form following a common standard, i.e., UK Railway Group Standard GK/RT 0202 [15] although historic and regional peculiarities are not uncommon. One possible arrangement is having one company to design signalling and a competing company to verify it. The reasoning is that this way both parties are incentivized to do their best.

Manual review is a slow process with very high requirements to reviewers' expertise. It does not deliver any objective proof of safety. At the same time, it does not suffer from any limitations of a formal verification process.

Simulation Railway industry widely employs railway simulation tools. These range from coarse-grained simulation of a national railway network to a detailed simulation of various aspects of mechanical performance of specific engines and carriages in a combination with specific rail and balast types. Verification concerns span from analysis of digital communication protocols connecting trains and regional control to stressing of tunnels and bridges by passing trains. Simulation is widely applied for time table optimisation and interactive 3D simulation is sometimes used for driver training. RailSys [14] and OpenTrack [13] are two of the more well-known simulation suites applied in time table optimisation and general analysis of signalling performance.

The main attraction of simulation is that it does not require deep understanding of railway functioning. Simulation tools present many aspects of railway performance in an intuitive, visual manner helping to quickly obtain the big picture of overall layout and signalling performance. There is, however, no guarantee of safety as simulation can only ever consider a tiny proportion of all scenarios.

Model checking The safety challenge of railways and the fact that collision and derailment properties may be dealt with within the setting of discrete, inertia-less train movement makes railway safety verification especially appealing for formal method practitioners. The principal idea of railway model checking is quite simple: a model of train movement laws is combined with the definitions of track topology and signalling rules. A model checking tool attempts to go through all or many execution scenarios to confirm that unsafe scenarios are ruled out. The list of modelling notations used in this setting is practically endless. Notable examples include Coloured Petri nets [4], process algebra CSP [9], a continuation work based on the model-based notation ASM [10], an algebraic language Maude [8] and the B Method together with ProB model checking tool [12].

	<i>Review</i>	<i>Simulation</i>	<i>Theorem proving</i>	<i>Model checking</i>	<i>SafeCap</i>
<i>Rigour</i>	--	--	++	+	+
<i>Productivity</i>	--	~	--	+	++
<i>Expertise</i>	--	++	--	+	+
<i>Scalability</i>	+	-	++	--	+
<i>Expressiveness</i>	++	++	~	-	-
<i>Feedback</i>	+	++	-	+	~

Table 1. Characterisation of major verification techniques and the SafeCap approach.

Almost all model checking approaches allow automatic instantiation of template models making application of model checking relatively straightforward for engineers. Many tools are able to report a sequence of steps leading to a safety violation. While model checkers are able to analyse many more scenarios than a simulator this comes at a price of reduced expressiveness (i.e., inability to reason about track geometry) and proof certificate is generally not ultimate: there could be a false negative when a model is too large and too complex to analyse exhaustively.

Theorem proving Model checking imposes limitations on the model size and performs best with a relatively limited logical language. Theorem proving overcomes these limitations and offers potentially unlimited opportunities for verifying safety with the utmost level of rigour. Theorem proving is not necessarily an all-manual process: there is a large and successful community developing automated theorem provers [18]. At the moment, automated prove support is best in the domain of first order logic and set theory; an attempt at reasoning about continuous train dynamics is likely to require an intervention by a highly skilled verification expert - the kind of people mostly found in academia. From our experience, even reasoning about track geometry is surprisingly difficult as this is a problem outside of the typical application domain of verification tools. One success story with theorem proving is the ongoing application of B method in the railway domain [5]. J.-R. Abrial has published methodological guidelines on an economical use of basic logic and set theory to reason about railwat safety in a discrete setting [3].

Theorem proving, even with excellent tool support, requires a high level of expertise in formal verification and mathematical modelling. The semantic gap between logic and railway concepts is formidable. This leads to generally low productivity (but we should notice efforts like the BART tool for automatic refinement of B models [11]), difficulties in interpreting tool feedback, and posing verification statements in a manner convincing to a non-expert reviewer.

Summary Table 1 summarizes our assessment of verification techniques. Each aspect is given a score ranging from -- (very poor) to ~ (satisfactory) to ++ (excellent). The last column characterises the SafeCap verification approach discussed in the following sections. The approach, as the table illustrates, offers

Fig. 2. SafeCap Platform architecture.

tangible advantages over both model checking and theorem proving: it scales better than model checking, offers good productivity and does not require a high level of expertise.

3 Safety verification in SafeCap

The purpose of the SafeCap Platform is to enable railway engineers to analyse complex junctions by experimenting with signalling rules, signalling principles, track topology, safety limits (e.g., speed limits for points and crossings) while receiving an on-line feedback from automated verification and analysis tools.

We have build the Platform around Eclipse - a mature and extensible IDE framework. We used Eclipse Modelling framework (EMF) to realise our Domain Specific Language (DSL) [7]. One important consideration was the ability to benefit from the extensive EMF ecosystem which offers a tool-kit for model manipulation and the construction of graphical and textual editing tools. Apart from the editing tools, the main components of the Platform are transformation patterns, model-based animation, simulation and verification (Fig. 2).

The SafeCap platform aims to provide a versatile tool-kit for analysing railway node safety and capacity. The approach we have taken is based on a combination of theorem proving and constraint solving.

We have applied the Event-B modelling notation an its refinement methodology to develop a theory of safe railway. This theory explicitly describes train movements, signal operation and points control. It does not, however, deal with any specific topology or control table. The proof of safety (we consider absence of collisions and derailments, and protection of flanks) is done for some class of topologies and control tables. The proof of Event-B model, although challenging, is done once and for all.

An important by-product is the set of axiomatic conditions characterising the class of *safe topologies* and *safe control tables*. To establish that a given track topology and control table are safe we only need to check that they do not contradict the mentioned axiomatic conditions. We do not need to redo the proofs of Event-B model. Safety verification is accomplished by putting together the definition of a concrete topology, control table and the axiomatic conditions derived by the Event-B model. If a *constraint solver* does not find a contradiction in logical statements encoded by this composition then the concrete topology and control table are deemed safe. Returning to the Event-B domain, the absence of contradiction established by a constraint solver means that our generic Event-B model of train behaviour is refined by a model instantiated with the given track topology and control table.

Schema topology and control table theories come in the form of a list of first order logic predicates; they do not define any state transitions or dynamic behaviour but rather well-formedness requirements to objects describing track topology and control table.

For constraint solving, we make use of two sets of formal notations and tools: B together with ProB, and Why3. In the short term, we aim to benefit from their complimentary strengths; in a longer term, the dual verification path provides a logical redundancy that makes a low-level encoding or tool bug unlikely to be left undiscovered.

3.1 Discrete driving model

The discrete driving model is an Event-B model capturing train, signal and point behaviour. It proves that the described behaviour is contained within a certain safety envelope by formulating and proving, through a number of refinement steps, *safety invariants* corresponding to the first three verification objectives of Section 1. This model gives a formal definition of principal phenomena observed in railway operation: train movement, route reservation, point locking, route cancellation and so on.

To construct the proof we have used Event-B [2] and the Rodin Platform [16]. Train driving rules, like those given in Section 3, are encoded by Event-B events - atomic state transitions - so that the overall model defines a state transition system. The safety properties are stated as a system invariant: a subset of possible states where the dangerous situations may not occur. The proof is done inductively by examining the effect of each event on a given safety property and discharging relevant proof obligation (first-order logic theorems).

The model in Fig. 3 illustrates the notation and modelling style of Event-B. This particular model is the very first (abstract) model in the development chain.

The overall model is made of seven refinement steps with 470 verification conditions of which 301 were discharged automatically by Rodin theorem provers.

```

machine route0
sees ctx_line
variables
   $t\_line$  // Train/line association
   $t\_r\_hd$  // Train head position on a line
   $t\_r\_tl$  // Train tail position on a line
invariant
   $t\_line \in TRAIN \leftrightarrow LINE$ 
  // A train is mapped to the id of a route occupied by the head of a train
   $t\_r\_hd \in TRAIN \leftrightarrow \mathbb{N}_1$ 
  // correspondingly,  $t\_r\_tl(t)$  is the id of the route occupied by the tail of train  $t$ 
   $t\_r\_tl \in TRAIN \leftrightarrow \mathbb{N}_1$ 
   $\text{dom}(t\_line) = \text{dom}(t\_r\_hd)$ 
   $\text{dom}(t\_line) = \text{dom}(t\_r\_tl)$ 
  // A train occupies a continuous route interval of route from tail till head
   $\forall t \in \text{dom}(t\_line) \Rightarrow t\_r\_tl(t) .. t\_r\_hd(t) \neq \emptyset$ 
  The routes a train occupies are the routes defined by the train line
   $\forall t \in \text{dom}(t\_line) \Rightarrow t\_r\_tl(t) .. t\_r\_hd(t) \subseteq \text{dom}(Line(t\_line(t)))$ 
  // Initially, there are no trains in the system
initialisation
   $t\_line, t\_r\_hd, t\_r\_tl := \emptyset, \emptyset, \emptyset$ 
events
  // A train may appear in the system with this operation
  appear =
    any  $t, l$  where
       $t \in TRAIN \setminus \text{dom}(t\_line)$  // a train must be not already in the system
       $l \in LINE$ 
    then
       $t\_line(t) := \text{lset the train line to } l$ 
       $t\_r\_hd(t), t\_r\_tl(t) := 1, 1$  // set head and tails routes
    end
  // Moves the head of a train from one route to another
  move_route_hd =
    any  $t$  where
       $t \in \text{dom}(t\_line)$ 
       $t\_r\_hd(t) < LineLen(t\_line(t))$  // train head must not be on the last line route
    then
       $t\_r\_hd(t) := t\_r\_hd(t) + 1$  // move the head one step forward
    end
  // Moves the tail of a train between routes
  move_route_tl =
    any  $t$  where
       $t \in \text{dom}(t\_line)$ 
       $t\_r\_tl(t) < t\_r\_hd(t)$  // a tail must be strictly behind the head of the train
    then
       $t\_r\_tl(t) := t\_r\_tl(t) + 1$  // move the tail one step forward
    end
  ...

```

Fig. 3. An Event-B model of abstract, route-level train movement (an excerpt).

```

/* (1) */ {} <<: NODE &
/* (2.a) */ {} <<: TRACK &
/* (2.b) */ TRACK <: NODE * NODE &
/* (2.c) */ elm(TRACK) = NODE & /* all nodes are connected by tracks */
...
/* (10) */ AMBIT : LA --> (POW(NODE) * POW(TRACK)) &
/* (11) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p ) => p <: q * q & {} <<: p) &
/* (12) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p ) => p~ <: p) &
/* (13) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p ) =>
! (n) . (n : q => closure(p)[{n}] = q) ) &
/* (14) */ union({p | # (a, q) . (a : ran(AMBIT) & q <: NODE &
p <: TRACK & a = ( q |-> p))}) = TRACK &
/* (15) */ ! (a, b, r, s, t, q) . (a : ran(AMBIT) & b : ran(AMBIT) & a /= b &
a = (r |-> s) & b = (t |-> q) => s /\ q = {}) &
...

```

Fig. 4. Schema well-formedness rules (an excerpt)

3.2 Schema topology theory

The schema topology theory is responsible for verifying logical conditions expressed over track layout (i.e., track connections, point placement) and logical topology (i.e., routes and lines as paths through a schema). Few examples of verification conditions include the connectivity property (no isolated pieces of track), continuity of routes and lines, absence of cycles, correct traversal of points and valid placement of train detection circuit boundaries.

We do not expect these conditions to uncover a problem with an existing track layout for any such a defect would have a profound effect on overall integrity - something unlikely to be left undiscovered in an operational railway. However, semi-automated alteration and generation of track layouts (e.g., via the improvement patterns we are developing in the tool) necessitates a careful and strict inspection of these basic properties. An automated verification process ensures high productivity and enables an engineer to explore a large range of designs within a short time.

Figure 4 gives a sample of verification conditions written in the Classical B notation [1] and ready to be processed by model checking tool ProB [12]. Not shown is the encoding of DSL elements (track graph, control tables) as sets, relations and functions of a B model. For a real-life example, such a model may be 6-14 thousand lines long. The same conditions and constructs are also generated in the Why3 theory notation. It is not a direct translation of the B model and we intentionally use a differing representation of relations and functions to introduce a form of modelling diversity. At the moment, for the topology theory, ProB and Why3 verifications chains deliver broadly similar performance.

3.3 Control table theory

On the platform of the topology verification we define the conditions of operational safety. These are derived, via a formal proof, from a set of discrete (inertia-less) train movement rules and expressed as a set of constraints over signalling rules.

```

...
/* @label (CT.1): A permissive signal may be lit only when all route ambits are clear */
! (l, r). (l |-> r : CTO_DOM => ! (n). (n : 1 .. RASPECT(l, r)-1 =>
    routeambits(r) <: CT_CLEAR(l, r, n) )) &
/* @label (CT.2): A route with an overlap may have permissive signal only
when its overlap is reserved and confirmed as clear */
! (l, r). (l |-> r : ROVERLAP & r : dom(LINE(l)) => ! (n). (n : 1 .. RASPECT(l, r)-1 =>
    TA[fst(ROUTE(LINE(l)(r))) ] <: CT_CLEAR(l, r, n))) &
/* @label (CT.3.a): No point is set both normal and reverse */
! (l, r). (l |-> r : CTO_DOM => CT_NORMAL(l, r) /\ CT_REVERSE(l, r) = {} ) &
...

```

Fig. 5. Control table conditions (an excerpt)

In SafeCap, we depart from the convention of associating control rules with track-side signals. Instead, we consider a more general situation where differing set of signalling rules are applied depending upon the ultimate train destination or train type and attach control logic to a pair of line and route. This permits, for instance, to model, on the same track, an express train using two aspect signalling and a freight train travelling over the same routes but in a three or four aspect mode. Such an arrangement may be used to achieve an optimal balance between headway and average speed in a heterogeneous traffic mix. Given the fact that in UK track-side signals are going to be made obsolete by 2030 [19] this represents a fairly modest scenario of using virtual signals to improve capacity.

The control table theory demonstrate such properties as the absence of potential collision (as may happen, for instance, when a proceed aspect is given while a protected part of track is still occupied) and derailment (due to incorrect point setting or point movement under a train). Other properties relate to the danger or circular dependencies between signals, dependencies between multi-aspect signals and operation of auto-signals, conformance with an ATP system, and verification of point and signal based flank protection. Certain properties, notably approach speed control via the timed occupation of a track section, are not verified at this stage as the formalisation at this layer does capture train inertia. Speed limit conformance and other time-related properties are formulated at the most detailed, fourth layer.

A list of sample control table theory conditions is given in Figure 5. For the shown rules, the outer quantification selects a pair of a line and a route that define a list of control rules (one per aspect). This models includes the topology model¹. Constraint solving is the primary verification strategy: we try to detect a contradiction between concrete data structures defining topology and control tables and the verification conditions. Again, the model is given in both B and Why3 notations although this time the Why3 verification route is not successful for larger examples. In addition, for any mid to large scale schema we currently have to exclude the verification of flank protection properties as these require complicated computations over track topology. We are working on a program that would output a proof term for each instance of flank protection property

¹ At the level, it is assumed that the topology theory has been verified and the topology constraints are turned into axioms

Benchmark	<i>Points/Lines/ Routes</i>	<i>Conditions, topology</i>	<i>Conditions, control table</i>	<i>Run time, topology</i>	<i>Run time, control table</i>
<i>Station 1</i>	8/12/14	117	230	4s	2s
<i>Junction 1</i>	23/4/21	280	602	24s	8s
<i>Station 2</i>	6/23/21	104	678	18s	6s
<i>Carlisle, west</i>	24/112/30	350	888	1m 17s	12s
<i>Carlisle</i>	63/161/79	892	1270	6m 4s	19s

Table 2. Verification run times for several sample layouts.

so that a theorem prover or a constraint solver would only have to check the elementary steps of a prepared proof.

4 Experimental results

We have tested our approach on a range of synthetic and real-life examples. Whenever possible, we have tried to accurately reproduce layout topology and control tables from paper-based documents provided by UK’s Network Rail. A summary of verification experiments is given in Table 2. Control table verification scales very well; the solving of topology suffers mainly from the conditions establishing absence of cycles in the definitions of lines and routes. If these properties are known to hold (i.e., topology is taken from an existing layout) these properties may be suppressed.

One of the larger examples we have tackled is the Carlisle Citadel station with the North, South, and Caldew junctions. The modelled fragment is 2.6km long and comprises 70 train detection circuits, 63 points, 79 routes and 161 valid paths. The translation from a scanned PDF drawing and printed control tables took 45 man-hours. The verification of the topology theory constraints using ProB took just over 6 minutes on a PC with i7 3730 CPU and utilized just under 2GB of RAM. The Why3 verification of the same theory takes approximately 70 minutes. The control table theory is verified under 20 seconds by ProB and not verified completely, with the current translation of conditions, using Why3.

The SafeCap Platform is freely available from [17] together with the verification models discussed in this paper.

5 Conclusions

In the paper we have surveyed the architecture of the verification infrastructure of the SafeCap Platform. Our approach has proven to be successful and we are working with our industrial partners to further improve the fidelity of the models and the robustness of the verification chain.

One of the advantages of applying formal modelling in the railway domain is the ability to transfer the modelling expertise accumulated analysing existing signalling laws to the verification of novel, unexplored ideas. The level of confidence a formal approach brings is especially valuable in overcoming a healthy

scepticism over a novelty in a field known for its conservatism. We plan to extend our approach to cover moving block, virtual signals and automated train operation in a uniform and coherent manner. We hope this will allow us to reason about hybrid solutions combining elements of fixed and moving block principles, human driving and fully automated operation. As one example, we would like to model how several similar trains travelling through a junction may be temporarily signalled using the train convoy principle where train separation control may be realised on a more relaxed assumption leading to smaller headway and less capacity consumption.

References

1. J.-R. Abrial. *The B-Book*. Cambridge University Press, 1996.
2. J.-R. Abrial and L. Mussat. Introducing Dynamic Constraints in B. Second International B Conference, LNCS 1393, Springer-Verlag, April 1998.
3. Jean-Raymond Abrial. Rigorous development of complex fault-tolerant systems. chapter Train Systems. Springer-Verlag, 2006.
4. C. W. Janczura. *Modelling and Analysis of Railway Network Control Logic using Coloured Petri Nets*. PhD thesis, School of Mathematics and Institute for Telecommunications Research, University of South Australia, 1998.
5. Didier Essamé and Daniel Dollé. B in Large-Scale Projects: The Canarsie Line CBTC Experience. In Jacques Julliand and Olga Kouchnarenko, editors, *B*, volume 4355 of *Lecture Notes in Computer Science*, pages 252–254. Springer, 2007.
6. P. Behm et al. METEOR: A successful application of B in a large project. Proceedings of the World Congress on Formal Methods, LNCS 1709, Springer, 1999.
7. A. Iliasov and A. Romanovsky. SafeCap domain language for reasoning about safety and capacity. Newcastle University, Technical Report CS-TR-1352. September 2012.
8. J. Bjork, A. M. Hagalisletto, and P. Enger. Large Scale simulations of Railroad Nets. 2006.
9. K. Winter. Model Checking Railway Interlocking Systems. 2002.
10. K. Winter and N. Robinson. Modelling Large Railway Interlockings and Model Checking Small Ones. 2003.
11. L. Burdy. Automatic Refinement.
12. M. Leuschel and M. Butler. ProB: A Model Checker for B. In Araki Keijiro, Stefania Gnesi, and Mandrio Dino, editors, *Formal Methods Europe 2003*, volume 2805, pages 855–874. Springer-Verlag, LNCS, 2003.
13. OpenTrack simulator. website. Available at <http://www.opentrack.ch/>.
14. RailSys simulation platform. website. Available at <http://http://www.rmcon.de>.
15. Railway Group Standards. Signalling Design: Control Tables. Available at <http://www.rgsonline.co.uk/>.
16. Rigorous Open Development Environment for Complex Systems (RODIN). IST FP6 STREP project, online at <http://rodin.cs.ncl.ac.uk/>.
17. SafeCap Project. SafeCap Platform website. 2013. Available at <http://safecap.sf.net>.
18. TPTP. Thousands of Problems for Theorem Provers. Available at www.tptp.org/.
19. TSLG. The Rail Technical Strategy (RTS). 2012. Available at <http://www.futurerailway.org/RTS/Pages/Intro.aspx>.
20. W.J. Fokkink and P.R. Hollingshead. Verification of Interlockings: from Control Tables to Ladder Logic Diagrams.