SAFECAP domain language for reasoning about safety and capacity

Alexei Iliasov
School of Comp. Science
Newcastle University
Newcastle upon Tyne, England
alexei.iliasov@ncl.ac.uk

Alexander Romanovsky
School of Comp. Science
Newcastle University
Newcastle upon Tyne, England
alexander.romanovsky@ncl.ac.uk

***Abstract*—The on-going UK SAFECAP project [1] develops modelling techniques and tools for improving railway capacity while ensuring that safety standards are maintained. This paper reports recent SAFECAP results on designing a Domain Specific Language (DSL) that will allow engineers to improve the node and junction capacity while guaranteeing operational safety. The SAFECAP DSL is introduced to define railway topology, its logical structure and signalling rules. The formal semantics of this graphical DSL, defined as part of our work, allows us to reason about system safety. The tooling environment, the SAFECAP Platform, offers graphical editing of railway schemas and an interface to a range of verification for ensuring railway operational safety. The work on extending the environment and its deployment in the railway sector continues with our SAFECAP partners: Invensys Rail and Swansea University.**

I. 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 [2] for designing various metro and suburban lines, and airport shuttles all over the world [3], [4]. 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.

Our aim is to develop methods and tools that allow engineers to improve the node and junction capacity while guaranteeing operational safety. By capacity we understanding the capability of a given railway layout and associated signalling rules to provide a certain quality of services determined by specific traffic patterns. More information about the SAFECAP approach to capacity may be found on the project website [1].

To achieve capacity improvement we offer engineers to model various changes of the layouts and signalling rules in the vicinity of the junctions or stations and evaluate the effect of the changes on the overall capacity. The safety of the original and modified models are checked by automated tools via a translation of railway topology and signalling rules into formal verification conditions.

A common domain language for the description of railway schemes and signalling rules was necessitated by the desire to apply and compare the differing modelling techniques used in the project to a common set of problems as defined by the industrial partner of the project, Invensys. The domain language is meant to satisfy the requirements of a railway engineer, who prefers to see a detailed layout presented in a way as close as possible to the established practice, and a researcher, who needs to work with a relatively small and precisely defined set of concepts. It is, perhaps, impossible to completely meet such conflicting requirements in a single language but we see it as one of the project goals to make a solid effort in this direction.

The main influence on this effort is, perhaps, the Dines Bjørner's railway domain language [5], [6]. Since we do not consider the macro level of railway networks our language has a far narrower scope. We have found it necessary to differ in the details of topology definition and treatment of control rules.

A relatively important aspect is the role of the domain language as means for communicating problems, solutions and ideas related to the investigation of capacity. This is important as otherwise the partners work with differing notations, tools and methodological frameworks. Such diversity may be an advantage but only when there is a cross-fertilization of ideas and methods.

Our domain language is at the heart of a modelling environment called the SAFECAP Platform [7]. The environment offers a visual modelling interface that may be used by a railway engineer, not trained in any manner in the use of such tools, to enter and update railways schemas and access 'by a press of a button' a range of verification tools.

II. SAFECAP DSL

In SAFECAP we study in isolation complex stations and junctions of a railway network. The focus is on the analysis and improvement of the throughput of a given layout (also called a *schema*) to meet or exceed the conditions of a service pattern defined by the wider context of the railway

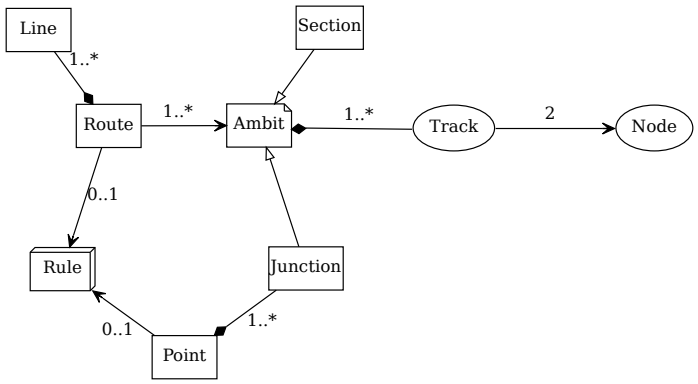


Figure 1. Core concepts of SAFECAP DSL and their interrelations.

network containing the layout. In geographical terms, a larger layout may span over an area of few square miles.

An effort at an improvement requires changing railway topology and signalling rules. Any such change must be analysed from the position of operational safety: absence of collisions and derailments. A complex project may require hundreds or thousands of changes (such large figures are achievable with mechanised transformation patterns) making an efficient and scalable tool support paramount. The aim of automated verification predicates a certain degree of formality in the definition of language concepts.

SAFECAP offers a fairly compact DSL (see Fig. 1 for the summary of language concepts). This is due to the limited scope of our study - railway junctions rather than complete networks - and also the desire to have an open-ended language where differing secondary elements may be defined for specific problem classes. The domain language defines the foundational concepts for the modelling of railway capacity. It attempts to capture, at a suitable level of abstraction, track topology, route and path definitions and signalling rules. By design, the language is extensible: one can dynamically define further attributes for all the predefined language elements and these are automatically picked up by the SAFECAP Platform [7] editor at no extra implementation effort.

In this paper we omit the discussion of various elements and attributes pertinent primarily to capacity assessment and focus on the safety part of the language. An emphasis is made on *route-based signalling* principles where track reservation in front of a train is done with the granularity of statically defined blocks.

We choose not to deal with the possibility of equipment failures and driver errors. Most such failures and errors have a catastrophic effect on the short-term capacity of a schema, making it difficult, at the current stage of research, to devise practical notions of capacity in presence of failures.

A. Topology definition

The basic element of a SAFECAP schema is the definition of railway topology.

A track is a piece of physical railway track; it is characterised by length, measured in meters and a *height map* - a detailed profile of track gradient used for capacity assessment. Track is not directed - a train may travel over track in either direction. The set of all tracks is T.

A node is a fictitious device gluing tracks into a continuous layout. Each track is associated with two nodes which it connects.

Set of all nodes of a schema is denoted by N. To formally capture the role of a node as track connector we say that a piece of track is defined by two ordered pairs $\{(f, s), (s, f)\}$ of nodes $f, s \in \mathsf{N}$. A single pair, i.e., (s, f), identifies a specific direction of one track that is helpful, for instance, in the description of point properties. Set T is a symmetric relation over nodes: $\mathsf{T} \subseteq \mathsf{N} \times \mathsf{N}$.

Tuple (N, T) defines an undirected graph known as the *topology graph of a schema*. Such a graph must not contain self-loops (tracks that start and end on the same node) and must satisfy certain restriction on the number of tracks connected to any given node (i.e., the *degree* of a node). A valid topology contains nodes of degrees 1 to 4. There can be no isolated nodes (degree 0) and nodes with degree higher than 4[1]. The following are the types of nodes, arranged by their degree:

degree 1: a *boundary node*; such a node marks the boundary between the considered layout and the rest of a railway network;
degree 2: a normal node; it is used to connect two pieces of track;
degree 3: a *point node*;
degree 4: a *diamond crossing*.

On the diagram in Fig. 2, node positions are identified by letters; boundary nodes are highlighted by pale green

[1]We choose, at this time, to not consider non-standard points and crossings.

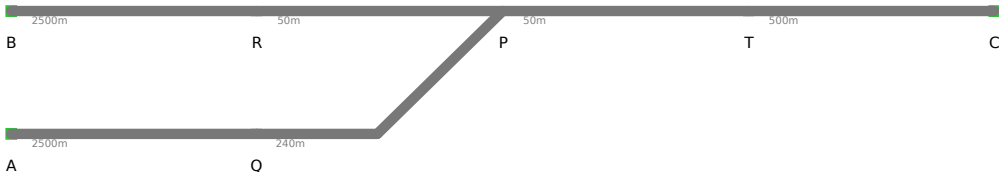


Figure 2. A topology graph example.

(gray) squares. For capacity calculations, boundary nodes may be further qualified into *sink* and *source* nodes to define nodes where trains only appear or disappear. The number under a track is the track length in meters; height map is not depicted. [2] The diagram defines a topology graph (N, T) where

- $\mathsf{N} = \{A, B, C, R, Q, T, P\}$;
- $\mathsf{T} = T \cup T^{-1}$ where $T = \{(A, Q), (Q, P), (B, R), (R, P), (P, T), (T, C)\}$.

B. Logical structure

To have several trains travelling over the same physical topology it is normally required to have an additional layer of structuring. This layer assists in the formulation of train movement principles that ensure safety.

A point is a machine that moves loose ends of rails to alter the path of a train. In logical terms, a point identifies a sub-graph of the overall layout available at any given moment. A point is positioned over some node n with degree 3 with three connected tracks: $N = \{(n, x), (x, n)\}$, $R = \{(n, y), (y, n)\}$ and $L = \{(z, n), (n, z)\}$. Track N is said to be the *normal branch* of a point; track R is the *reverse branch* and L is the *lead track*. A train may travel through a point by going on a lead track and continuing on a branch or normal track. In the reverse direction, it may go from a branch or normal track to a lead track. Topologically, normal and reverse branch roles are interchangeable; a difference arises in capacity assessment where the reverse branch is, typically, taken at a lower speed than its normal counterpart.

The set of all points is identified as $\mathsf{P} \subseteq \mathsf{N} \times \mathsf{N} \times \mathsf{N}$. For some point $(a, b, c) \in \mathsf{P}$, a defines the node position of the point; tracks $\{(a, b), (b, a)\}$ and $\{(a, c), (c, a)\}$ are the normal and reverse branches of the point. Given a topology graph and a point triplet one can trivially compute the lead track of a point.

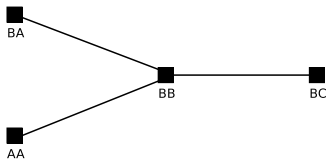
An ambit[3] is a connected sub-graph of the topology graph equipped with a *train detection circuit*. Such a circuit enables an ambit to detect the presence of a train. An ambit can only report whether there is a train anywhere on ambit tracks but not the number of trains, if there is more than one, exact positions or the occupation status of individual tracks. We do not consider the possibility of detection circuit failure.

[2]This diagram is generated by the SAFECAP Platform.

[3]We have opted for a new way to refer to a train detection circuit to avoid confusion with inconsistently used existing terminology.

The set of all ambits is A. Since an individual ambit $a \in \mathsf{A}$ is a sub-graph of the topology graph - $a \subseteq \mathsf{N} \times \mathsf{T}$ - the type of A is $\mathsf{A} \subseteq \mathbb{P}(\mathsf{N} \times \mathsf{T})$. It is required that each ambit is associated with a unique set of tracks so that no two ambits share tracks.

It happens that for any schema there is a unique connected graph where ambits are vertices (see Theorem 1). This fact means that, when necessary, one can reason at a coarser level of ambits in place of tracks and every path formulated in terms of ambits relates to a non-empty set of track-level paths. Conversely, every track path has a unique representation at the ambit level. The following is an ambit graph for the example in Fig. 2:



The ambit-level viewpoint on a schema topology is often preferable when reasoning about signalling and safety.

There are two important cases of ambits - those that include points and and those that do not. The latter are called *sections* and are made of linear track and diamond crossings. The former are known as *junctions* and, in addition to linear track and diamond crossing contain one or more points. From modelling viewpoint, sections are simpler entities as they do not change their configuration over time.

A route is a continuous connection between two nodes of a layout. In *route-based signalling*, a route is also the unit of track reservation - a process that ensures the availability of some free track in front of a train. Formally, a route is a path sub-graph of a topology graph. The set of a routes, R, is a subset of all such sub-graph; overlapping routes (sharing one or more ambits) are permitted.

A line is a sequence of routes; such a sequence must form a path through a layout starting and ending on boundary nodes. The set of all lines is known as L.

From the definition of individual routes, one can may compute the path of line as a sequence of ambits; in other words, the path sub-graph of the ambit graph of a schema. Similarly, a line may be seen a list of tracks, i.e., a path sub-graph of a topology graph, or as a list of ambits.

We extend our small example with notions of point, ambit, route and line. The following visual notation is used: the

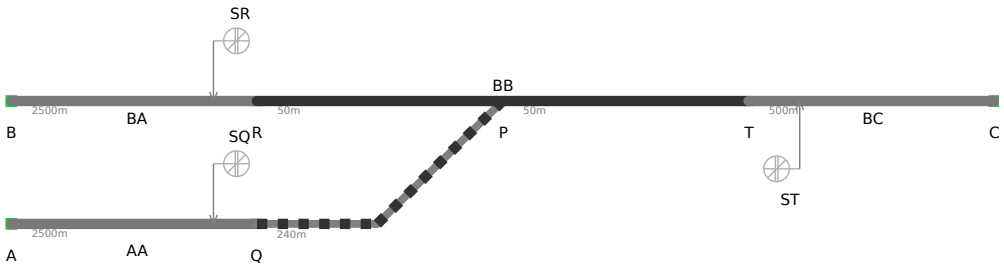


Figure 3. A topology graph overlaid with logical structuring.

normal and lead part branches of a point are highlighted in a darker shade while the reverse branch is shown as a dashed line; route start and end points are marked by signal[4] pictograms and boundary nodes (note that signals are oriented - signals over the track are for trains travelling from left to right; signals below the track are for right-to-left train).

The diagram introduces the following logical structuring elements:

- Ambits AA, BA, BB, BC, of which
  - AA, BA, BC are sections;
  - BB is a junction;
- Point (P, R, Q), as a part of junction BB.
- Routes: AQ, QC, BR, RC, CT, TA, TB;
- Lines: AC, BC, CB, CA.

Note that AB and BA are not valid lines since junction BB may not be used, due to the point orientation, to travel from ambit AA to ambit BA and vice versa.

C. Signalling rules

A signalling rule is a logical condition attached to routes and points defining when a route is available for travelling (and at which speed) and when a point may be moved. The following notation[5] is used to write signalling rules:

$$
\begin{array}{rcl}
rule & := & \textbf{clear}\ al \\
& | & \textbf{occupied}\ al \\
& | & \textbf{normal}\ pl \\
& | & \textbf{reverse}\ pl \\
& | & \textbf{state}\ rl\ \textbf{is}\ rs \\
& | & rule \wedge rule \\
& ; &
\end{array}
$$

where $al \subseteq \mathsf{A}$ is a set of ambits, $pl \subseteq \mathsf{P}$ is a set of points, $rl \subseteq \mathsf{L} \times \mathsf{R}$ is a set of (line, route) pairs of line such that for any $(l, r) \in rl$ it holds that r is a route of line l.

Value $rs \in \mathbb{N}$ is a natural number signifying a route state, or, equivalently, the current aspect of a physical or logical (e.g., displayed inside a driver's cabin) signal. State $rs = 0$ indicates that a route is not available; state $rs > 0$ indicates that a route is available and there are $rs - 1$ reserved and available routes following this route. The only connective we use to form complex rules is conjunction $\wedge$.

Another interpretation of route state rs may be given by introducing a constant function $safespeed \in \mathsf{R} \times \mathsf{L} \times \mathbb{N} \to \mathbb{R}$ converting the current aspect index into the maximum safe speed of train. This information is essential for SAFECAP capacity assessment methods that rely on estimates of time a train occupies a certain part of a track.

One straightforward definition of $safespeed$ is based on the following principle: *at all times, a train must be able to decelerate to complete stop within the guaranteed free distance in front.*

The minimum braking distance of a train is $BD(v, d) = v^2/(2d)$ where v is the current speed and d is the maximum train deceleration. Considering a situation when the front of a train is right at the edge of route r, distance $BD(v, d)$ is the guaranteed free distance in front of a train and is determined by the cumulative lengths of track of rs successor routes. Let $sr(r, l, n)$ be n successors of route r on some line l: $sr(r, l, n) = \{l^i(r) \mid i \in 0 \,..\, n\}$ (see conditions (22) in Section III). Then the safe speed at the start of a route is

$$
safespeed(r, l, n) = \left| \sqrt{2d \sum_{r \in sr(r,l,n-1)} \sum_{t \in \mathrm{elm}(r)} \mathrm{len}(t)} \right|
$$

where $\mathrm{elm}(r) = \mathrm{dom}(r) \cup \mathrm{ran}(r)$ and $\mathrm{len}(t)$ is the length of track t. As expected, when $n = 0$ (that is, a signal attached to r is showing 'red' aspect), $sr(r, l, n) = \varnothing$ and $safespeed(r, l, 0) = 0$.

The value of $safespeed$ depends on the current line l (since a route may be shared among lines, one needs to know l to find the relevant successor routes), the lengths of individual routes in front of the train and the applicable value of d. The current value of $safespeed(r, l, n)$ may be computed by an on-board computer or estimated by a train driver (who must have an intimate knowledge of line l). This illustrates the connection between safety, as defined by signalling rules, and capacity which depends on the speed at which trains progress. When the number of gradations of

[4] SAFECAP DSL does *not* define the notion of a signal. The conventional signal sign is used only as a part of the visual syntax to mark route boundaries.

[5] In our tool, the SAFECAP Platform, signalling rules may also be entered in a tabular form.

proceed aspects is large it may more convenient to report the state of a route as safe speed $safespeed(r, l, n)$. There are signalling procedures operating this way deployed across Europe.

Signalling rules are assigned to points and a pair of routes and lines. The rule of a point defines the condition when the ends of the point may be safely moved (thus avoiding potential derailment); the rule of a route defines the condition when the route is available for reservation and subsequent train movement. Let rule be a syntactic representation of a rule. Point rules are known as S_P and routes rules as S_R:

$$S_P \in \mathsf{P} \to \mathrm{rule} \qquad S_R \in \mathsf{L} \times \mathsf{R} \to \mathrm{rule}$$

In Section III-B we replace the syntactic form with an interpretation yielding a boolean when evaluated on a certain state.

Let's return to the example in Fig. 3. The following are the signalling rules ensuring a safe railway operation. Each route rule protects all the ambits of the route and, in some cases, requests that point P is set in an appropriate direction. Point P may be moved only when ambit BB is clear.

$$\begin{array}{rcl} S_R(AC, AQ) & = & \mathbf{clear}\ AA \\ S_R(AC, QC) & = & \mathbf{clear}\ BB, BC \wedge \mathbf{reverse}\ P \\ S_R(BC, BR) & = & \mathbf{clear}\ BA \\ S_R(BC, RC) & = & \mathbf{clear}\ BB, BC \wedge \mathbf{normal}\ P \\ S_R(CA, CT) & = & \mathbf{clear}\ BC \\ S_R(CB, CT) & = & \mathbf{clear}\ BC \\ S_R(CA, TA) & = & \mathbf{clear}\ BB, AA \wedge \mathbf{reverse}\ P \\ S_R(CB, TB) & = & \mathbf{clear}\ BB, BA \wedge \mathbf{normal}\ P \\ S_P(P) & = & \mathbf{clear}\ BB \end{array}$$

To summarise, the *static* part of a SAFECAP schema is a structure of the following form

$$(\mathsf{N}, \mathsf{T}, \mathsf{A}, \mathsf{P}, \mathsf{R}, \mathsf{L}, S_P, S_R)$$

where N, T, A, P, R, L are the set of nodes, tracks, ambits, points, routes and lines; S_P and S_R are point and route signalling rules. In the following section we discuss conditions identifying valid SAFECAP schemas.

III. SEMANTICS

Having defined the concepts of our language we shall now give a precise formulation of *theory* $\mathcal{SC}$ of SAFECAP schemas. The elements of the theory are axioms defining topological, structuring and safety constraints. To demonstrate that a given schema s is valid we ask to prove that it constitutes a valid model in $\mathcal{SC}$. All $\mathcal{SC}$ axioms are written in first order logic. When instantiated for a particular schema, they turn into propositional formulas in ZF set theory.

There are two major classes of $\mathcal{SC}$ axioms: those dealing with the well-formedness of the static part of a schema; and those defining and constraining what we call the *dynamic properties* of a schema. The former define properties of physical track topology and logical structuring into ambits, routes and lines. The latter address train movement safety.

A. Static Well-formedness

In this section, we start with the characterisation of a topology graph and then express how the definitions of points, ambits, routes, lines relate to the topology graph and each other.

Set N of nodes is not empty,

$$\varnothing \subset \mathsf{N} \tag{1}$$

Set T of tracks defines a non-empty relation over nodes,

$$\varnothing \subset \mathsf{T} \subseteq \mathsf{N} \times \mathsf{N} \tag{2}$$

Topology graph (N, T) may not contain self-loops, i.e., relation T is irreflexive,

$$\mathrm{id}(\mathsf{N}) \cap \mathsf{T} = \varnothing \tag{3}$$

Track may be used in either direction and thus graph (N, T) is undirected,

$$\mathsf{T}^{-1} \subseteq \mathsf{T} \tag{4}$$

Topology graph (N, T) must be connected - there should exist a path between any two nodes of the graph. This is shown by proving that from any node n one can reach all other nodes except the starting node,

$$\forall n \cdot n \in \mathsf{N} \Rightarrow \mathsf{T}^*[\{n\}] = \mathsf{N} \setminus \{n\} \tag{5}$$

where T^* is a transitive closure of T. Node degree $\deg(n) = \mathrm{card}(\mathsf{T}[\{n\}] \cup \mathsf{T}^{-1}[\{n\}])$ signifies the number of tracks incident to node n. It must hold for every node that the degree is between 1 and 4,

$$\forall n \cdot n \in \mathsf{N} \Rightarrow 1 \leq \deg(n) \leq 4 \tag{6}$$

It is required that a schema provides a way for trains to enter and leave the schema layout. A train may only appear on a boundary node and disappear on a distinct boundary node,

$$\begin{array}{l} \exists n_1, n_2 \cdot (n_1, n_2) \in \mathsf{N} \times \mathsf{N} \wedge \\ \qquad \deg(n_1) = \deg(n_2) = 1 \wedge n_1 \neq n_2 \end{array} \tag{7}$$

A point is defined by a triplet of nodes,

$$\mathsf{P} \subseteq \mathsf{N} \times \mathsf{N} \times \mathsf{N} \tag{8}$$

A point is positioned on a node of degree 3; its normal and reverse branches are incident to the node where the point is located,

$$\begin{array}{l} \forall p, n, a, b \cdot p \in \mathsf{P} \wedge p = (n, a, b) \wedge \\ \quad \deg(n) = 3 \wedge (n, a) \in \mathsf{T} \wedge (n, b) \in \mathsf{T} \end{array} \tag{9}$$

An ambit is a sub-graph of a topology graph,

$$\mathsf{A} \subseteq \mathbb{P}(\mathsf{N} \times \mathsf{T}) \tag{10}$$

The first component of an ambit must define the set of vertices and the second must be a non-empty set of edges,

$$\forall a, q, p \cdot a \in \mathsf{A} \wedge a = (q, p) \Rightarrow p \subseteq q \times q \wedge \varnothing \subset p \tag{11}$$

Like the overall topology graph, an ambit is made of undirected track,

$$\forall a, q, p \cdot a \in \mathsf{A} \wedge a = (q, p) \Rightarrow p^{-1} \subseteq p \tag{12}$$

A sub-graph defined by an ambit must be connected,

$$\begin{array}{l} \forall a, q, p \cdot a \in \mathsf{A} \wedge a = (q, p) \Rightarrow \\ \quad \forall n \cdot n \in q \Rightarrow p^*[\{n\}] = q \setminus \{n\} \end{array} \tag{13}$$

Every track of a layout must be claimed by an ambit,

$$\bigcup_{a \in \mathsf{A}} \mathrm{prj}_2(a) = \mathsf{T} \tag{14}$$

However, no two ambits are allowed to share a track. In other words, ambits partition the topology graph,

$$\forall a, b \cdot a, b \in \mathsf{A} \wedge a \neq b \Rightarrow \mathrm{prj}_2(a) \cap \mathrm{prj}_2(b) = \varnothing \tag{15}$$

Due to the way train detection circuit hardware is realised, a point may not be positioned on the boundary of an ambit. That is, if an ambit contains the node where a point is placed, the ambit must also contain the lead, reverse and normal tracks of the point,

$$\begin{array}{l} \forall a, p, u, v, w \cdot a \in \mathsf{A} \wedge p \in \mathsf{P} \wedge \\ \quad p = (u, v, w) \wedge u \in \mathrm{prj}_1(a) \Rightarrow \\ \quad\quad (u, v) \in \mathrm{prj}_2(a) \wedge (u, w) \in \mathrm{prj}_2(a) \wedge \\ \quad\quad \{z\} = \mathsf{N} \setminus \{u, v, w\} \wedge (z, u) \in \mathrm{prj}_2(a) \end{array} \tag{16}$$

In (16) we are entitled to write $\{z\} = \mathsf{N} \setminus \{u, v, w\}$ since, by (9), node u has degree 3 and thus set $\mathsf{N} \setminus \{u, v, w\}$ contains exactly one element. For instance, for the schema in Fig. 3, it is not possible to have ambit BC to include tracks (P, T) and (T, P) as this would mean that point at node P spans over two train detection circuits.

Let $E \subseteq \mathsf{A} \times \mathsf{A}$ be a relation such that $(a, b) \in E$ if and only if there is a common node n with degree 1 in both a and b. Ambits a and b are called adjacent and tuple (A, E) defines a graph called *ambit graph*.

Theorem 1: Ambit graph (A, E) is connected and unique.

This theorem allows one to switch between reasoning at the level topology and ambit graphs.

A route is a sequence of tracks. We define it as a successor relation that is an injective and asymmetric function over tracks,

$$\begin{array}{l} \forall r \cdot r \in \mathsf{R} \Rightarrow \\ \quad r \in \mathsf{T} \nrightarrow \mathsf{T} \wedge r^{-1} \in \mathsf{T} \nrightarrow \mathsf{T} \wedge r \cap r^{-1} = \varnothing \end{array} \tag{17}$$

The asymmetry of a route gives it an orientation - every track included in a route is included with a specific orientation. A route may not contain cycles. A cycle in a next relation is present if there is a non-empty subset in the domain of the relation that is projected into itself,

$$\forall r \cdot r \in \mathsf{R} \Rightarrow \neg \exists c \cdot \varnothing \subset c \subseteq \mathrm{dom}(r) \wedge c \subseteq r[c] \tag{18}$$

A route must be non-empty. It is convenient to state this property as the existence of unique first and last elements of a route,

$$\forall r \cdot r \in \mathsf{R} \Rightarrow \mathrm{card}(\mathrm{fst}(r)) = \mathrm{card}(\mathrm{lst}(r)) = 1 \tag{19}$$

where $\mathrm{fst}(r)$ and $\mathrm{lst}(r)$ determine the first and last elements of a successor relation: $\mathrm{fst}(x) = \mathrm{dom}(x) \setminus \mathrm{ran}(x)$ and $\mathrm{lst}(x) = \mathrm{ran}(x) \setminus \mathrm{dom}(x)$.

Let ta map a track to an ambit in which the track is contained. Due to (14) and (15), mapping $ta : \mathsf{T} \rightarrow \mathsf{A}$ is functional and surjective. It is defined as $ta^{-1}(a) = \mathrm{prj}_2(a)$.

A path of a route may not start or end in a middle of an ambit but must rather be delineated by ambit boundaries. If the tracks of a route are mapped into ambits then the tracks of the ambits are exactly the tracks of the route,

$$\forall r \cdot r \in \mathsf{R} \Rightarrow (ta \circ ta^{-1})[\mathrm{elm}(r)] = \mathrm{elm}(r) \tag{20}$$

where $\mathrm{elm}(r)$ are the elements of relation r: $\mathrm{elm}(r) = \mathrm{dom}(r) \cup \mathrm{ran}(r)$.

One may not form a route by traversing a point in a wrong direction, that is, having a route path going from a branch to normal track of a point or vice versa,

$$\begin{array}{l}\forall r \cdot r \in \mathsf{R} \Rightarrow \\ \quad \forall p, u, v, w \cdot p \in \mathsf{P} \wedge p = (u, v, w) \Rightarrow \\ \quad\quad \{(v, u), (u, w)\} \not\subseteq \mathrm{elm}(r) \wedge \\ \quad\quad \{(w, u), (u, v)\} \not\subseteq \mathrm{elm}(r)\end{array} \tag{21}$$

For some point $p = (u, v, w)$ prohibited paths are sequences of tracks $((v, u), (u, w))$ (going from normal to reverse branch) and $((w, u), (u, v))$ (going from reverse to normal branch). Since a route may not contain cycles (18) it is impossible to form a legitimate route that includes both normal and reverse track of a point in a certain orientation. A line is a sequence of routes. The following condition simply mirror the properties of a route,

$$\begin{array}{l}\forall l \cdot l \in \mathsf{L} \Rightarrow l \in \mathsf{R} \nrightarrow \mathsf{R} \wedge l^{-1} \in \mathsf{R} \nrightarrow \mathsf{R} \wedge l \cap l^{-1} = \varnothing \\ \forall l \cdot l \in \mathsf{L} \Rightarrow \neg \exists c \cdot \varnothing \subset c \subseteq \mathrm{dom}(l) \wedge c \subseteq l[c] \\ \forall l \cdot l \in \mathsf{L} \Rightarrow \mathrm{card}(\mathrm{fst}(l)) = \mathrm{card}(\mathrm{lst}(l)) = 1\end{array} \tag{22}$$

B. Dynamic properties

A number of important concepts such as, route and track reservation, train derailment and collision, become meaningful only when one considers the dynamics of a railway in terms moving trains, switching points and signals and so on. It is a fairly extensive subject and, in the scope of this paper, we only address a subset of dynamic properties that do involve the notion of a train. The properties we present in this section give the first approximation of actual phenomena. We do not consider the fact that many actions affecting schema properties have a considerable extent in time.

In addition to static parts of a schema such as tracks and nodes, we introduce are a number of *stateful* entities. These change their properties over time and especially due to the movement of trains across the layout of a schema. The stateful entities we discuss in this section are ambits, points and routes.

An ambit state reflects the current knowledge about the ambit occupation by a train,

$$\Omega_A \in \mathsf{A} \rightarrow \mathrm{BOOL}$$

For some ambit $a \in \mathsf{A}$, $\Omega_A(a) = \top$ signifies that a is occupied and $\Omega_A(a) = \bot$ that a is not occupied. The state of a route/line pair is the current aspect associated with the route and line. An aspect is encoded as a natural number with 0 indicating that the route is unavailable (i.e., 'red' signal) and > 0 for various grades of permissive signals,

$$\Omega_R \in \mathsf{L} \times \mathsf{R} \rightarrow \mathbb{N}$$

Finally, the state of a point determines which branch of the point is currently available for the formation of a route,

$$\Omega_P \in \mathsf{P} \rightarrow \mathrm{BOOL}$$

$\Omega_P(p) = \top$ signifies that the normal branch is currently available.

To reason about signalling rules attached to routes and points we need to give a concrete interpretation of their meaning. The following rules defines the conversion of statements in the language of signalling rules into boolean values. Let $\Omega = (\Omega_A, \Omega_R, \Omega_P)$. Then $[\![\ldots]\!]_\Omega \in \mathbb{B}$ such that

$$\begin{array}{rcl}[\![\mathbf{clear}\ al]\!]_\Omega & := & \forall a \cdot a \in al \Rightarrow \neg\, \Omega_A(a) \\ [\![\mathbf{occupied}\ al]\!]_\Omega & := & \forall a \cdot a \in al \Rightarrow \Omega_A(a) \\ [\![\mathbf{normal}\ pl]\!]_\Omega & := & \forall p \cdot p \in pl \Rightarrow \Omega_P(p) \\ [\![\mathbf{reverse}\ pl]\!]_\Omega & := & \forall p \cdot p \in pl \Rightarrow \neg\, \Omega_P(p) \\ [\![\mathbf{state}\ rl\ \mathbf{is}\ rs]\!]_\Omega & := & \forall l, r \cdot (l, r) \in rl \Rightarrow \Omega_R(l, r) \geq rs \\ [\![r \wedge q]\!]_\Omega & := & [\![r]\!]_\Omega \wedge [\![q]\!]_\Omega\end{array}$$

defines an interpretation of signalling rules where $al \subseteq \mathsf{A}$ is a set of ambits, $pl \subseteq \mathsf{P}$ is a set of points, $rl \subseteq \mathsf{L} \times \mathsf{R}$ is a set of (line, route) pairs.

Ω_R may be linked to syntactic rule definitions S_R. The state of a route indicates a permissive apsect only if the signalling rule interpretation yields truth,

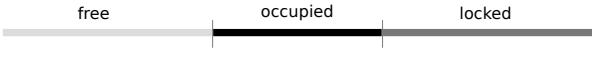
$$\forall l, r \cdot l \in \mathsf{L} \wedge r \in \mathrm{elm}(l) \Rightarrow (\Omega_R(l, r) > 0 \Leftrightarrow [\![S_R(l, r)]\!])$$

In case of a point, there is no dependency between condition $[\![S_P(p)]\!]$ thet defines *when* point p may be moved and condition $\Omega_P(p)$ giving the *current branch* of point p.

When we consider the dynamic part of a schema, a route reveals a finer structure. Assuming that a train occupies some proportion of a route, the route is split into three parts, namely:

- free ambits behind the train tail;
- currently occupied ambits;
- ambits locked in front of a train.

The diagram illustrates the situation.



While free and occupied ambits of a route may be characterised by Ω_A, locking of ambits is an entirely new concept that happens to be essential for the demonstration of absence of train collisions (see Section III-C). Ambit reservations AR are defined by a relation associating an ambit to a line,

$$\mathsf{AR} \subseteq \mathsf{A} \times \mathsf{L}$$

One can also define the notion of route reservations, i.e., $\mathsf{RR} \subseteq \mathsf{R} \times \mathsf{L}$, to explain how routes are reserved in front of train. The way such reservations are handled critically affects the estimation of capacity. A detailed discussion of this subject is outside of the paper scope.

For some $r \in \mathsf{R}$ from line l the following define the three parts of a route, as given by the diagram above,

- $ta[\mathrm{elm}(r)] \cap (\Omega_A^{-1}[\{\bot\}] \setminus \mathsf{AR}^{-1}[\{l\}])$ (free ambits of r);
- $ta[\mathrm{elm}(r)] \cap \Omega_A^{-1}[\{\top\}]$ (occupied ambits in r);
- $ta[\mathrm{elm}(r)] \cap \mathsf{AR}^{-1}[\{l\}]$ (locked ambits in r).

These parts do not overlap yet exhaust all the route ambits. The former is apparent for the first and second and also first and third parts. Occupied and locked ambits also do not overlap thanks to the following property demanding that a locked track is not occupied,

$$\Omega_A[\mathrm{prj}_1(\mathsf{AR})] = 0 \tag{23}$$

A locked ambit may only be found on an route currently occupied by a train. Hence, for every ambit a locked for line l, $(a,l) \in \mathsf{AR}$, there must exist an unavailable route from line l containing a,

$$\begin{array}{l} \forall l, a \cdot (a,l) \in \mathsf{AR} \Rightarrow \\ \quad \exists r \cdot r \in \mathrm{elm}(l) \wedge a \in ta[\mathrm{elm}(r)] \Rightarrow \Omega_R(l,r) = 0 \end{array} \tag{24}$$

Ambit reservation forms a continuous path from the front of train till the end of a route. Considering some locked ambit, any next ambit is either the last ambit of a current route or is locked. Symmetrically, a previous ambit is either locked or is the front of a train occupying the route. For the latter, we simply state that the ambit is occupied,

$$\begin{array}{l} \forall l, a, r \cdot (a,l) \in \mathsf{AR} \wedge r \in \mathrm{elm}(l) \wedge a \in ta[\mathrm{elm}(r)] \Rightarrow \\ \quad ((\exists z \cdot nxt(a,r,z) \wedge z \in \mathsf{AR}^{-1}[\{l\}]) \vee a = ta(\mathrm{lst}(r))) \wedge \\ \quad (\exists z \cdot prv(a,r,z) \wedge z \in \mathsf{AR}^{-1}[\{l\}] \vee \Omega_A(z)) \end{array} \tag{25}$$

where $nxt(a,r,z)$ binds n to the next successor, as defined by route r, of ambit a: $nxt(a,r,z) \equiv \{n\} = (ta^{-1} \circ r \circ ta)[\{a\}]$. Correspondingly, $prv(a,r,z) \equiv \{n\} = (ta^{-1} \circ r^{-1} \circ ta)[\{a\}]$.

A proceedable aspect of a route requires that all the route ambits are clear,

$$\begin{array}{l} \forall l \cdot l \in \mathsf{L} \Rightarrow \\ \quad \forall r \cdot r \in \mathrm{elm}(l) \wedge \Omega_R(l,r) > 0 \Rightarrow \\ \quad\quad \Omega_A[\mathrm{elm}(r)] = \{\bot\} \end{array} \tag{26}$$

An advanced aspect $rs > 1$ requires that the successor route on the same line has the state with the same or immediately preceding aspect,

$$\begin{array}{l} \forall l \cdot l \in \mathsf{L} \Rightarrow \\ \quad \forall r, a \cdot r \in \mathrm{dom}(l) \wedge a > 0 \wedge \\ \quad\quad \Omega_R(l,r) = a \Rightarrow \Omega_R(l, l(r)) \geq a - 1 \end{array} \tag{27}$$

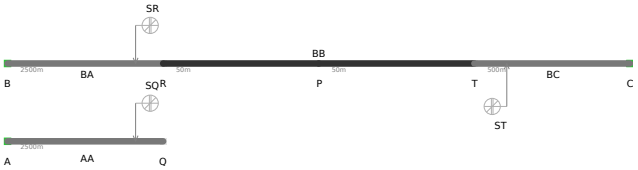
The condition says that if some current signal of line l at route r is showing a proceed aspect $as > 0$ then the successor route on the same line must show an aspect of at most one step lower. The property is not checked for the last route of a line (i.e., some $r = \mathrm{lst}(l)$) that, when not occupied, may show any aspect. We assume, for the benefit of capacity assessment that the layout outside of the considered part does not introduce capacity bottleneck and thus, effectively, has infinite capacity. For this reason, for end routes of a line we enforce the maximum proceed aspect as soon the route becomes proceedable.

The possibility of a dynamic change of point orientation means that at different times differing sets of routes are available. To convert the current state Ω_P of points into a set of available tracks we define helper function $C \in \mathsf{P} \to \mathsf{T}$,

$$C = \{\top \mapsto (u,w), \bot \mapsto (u,v)\} \circ \Omega_P$$

$C(p)$ returns a track that is either normal or reverse branch of point p and is *disabled* by the current configuration of point p. The main role of this construct is to alter the topology graph to determine which routes are available at the moment. Graph $(\mathsf{N}, \mathsf{T} \setminus \{C(p) \mid p \in \mathsf{P}\})$ is called the *actual topology graph* of a schema.

For the example in Fig. 3, setting point P to normal configuration ($\Omega_P(P) = \top$) results in the following actual topology graph:



As the diagram suggest, routes QC and TA are temporarily disabled by point P.

A point rule must give a condition when all the routes containing the point are blocked. This ensures that when a point indicates that it is safe to move the point there are no trains in the vicinity of the point and that a point is not moved if it is a part of a prepared route (a route with a preceedable aspect),

$$\begin{array}{l} \forall p, u, v, w \cdot p \in \mathsf{P} \wedge p(u,v,w) \wedge [\![S_P(p)]\!]_\Omega \Rightarrow \\ \quad \forall l \cdot l \in \mathsf{L} \Rightarrow \\ \quad\quad \forall r \cdot r \in \mathrm{elm}(l) \wedge (u,v) \in \mathrm{elm}(r) \Rightarrow \\ \quad\quad\quad \Omega_R(l,r) \geq 0 \end{array} \tag{28}$$

When a route shows a proceedable aspect all of its points must be set to conform with the path of the route,

$$\begin{array}{l} \forall l \cdot l \in \mathsf{L} \Rightarrow \\ \quad \forall r \cdot r \in \mathrm{elm}(l) \wedge S_R(l,r) > 0 \Rightarrow \\ \quad\quad \mathrm{elm}(r) \subseteq \mathsf{T} \setminus \{C(p) \mid p \in \mathsf{P}\} \end{array} \tag{29}$$

Statement $\mathrm{elm}(r) \subseteq \mathsf{T} \setminus \{C(p) \mid p \in \mathsf{P}\}$ checks that all the tracks of route r are contained in the actual topology graph. If a point from r were set incorrectly then some tracks from $\mathrm{elm}(r)$ would be missing in $\mathsf{T} \setminus \{C(p) \mid p \in \mathsf{P}\}$.

C. Safety Properties

We briefly discuss how the two central safety conditions of railways are addressed in SAFECAP DSL.

A schema must be free from collisions. A collision happens when two trains occupy the same part of a track. In terms of our DSL, we can only speak about potential collisions due to simultaneous occupation of the same ambit by two trains.

It is easy to see that one cannot show the absence of collisions without introducing an explicit train notion and explaining how a train moves around a railway layout, i.e., driving rules. If trains choose to ignore whatever means of indication of route states are available to them (e.g., track side signals) there is nothing preventing two trains from colliding. Hence, the absence of collisions is not a property of a schema per se but that of a combination of a schema and train driving rules (and, possibly, interlocking). Of course, it is essential to establish that a schema is safe under some sensible driving rules. We have done this with one specific encoding of human driving and certain ATP principles in an Event-B model based on the SAFECAP railway schema and proved the absence of collisions. The proof is inductive: starting from a collision free state we show that train movement rules preserve the invariant condition that all ambits are occupied by at most one train [6]

A schema must be free from derailments. A derailment may happen when a train moves over a point changing its configuration.

There is an explicit rule attached to each point defining when a point reconfiguration may happen (S_P from Section II-C). Condition (28) states that a point may be moved only when all the routes containing the point are not occupied. It is trivial to see that this is sufficient to ensure that none of track points are occupied when a point moves, provided that trains are driven in compliance with route states.

IV. TOOLING

SAFECAP DSL is the central component of our modelling framework - SAFECAP Platform [7]. The purpose of the 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. The overall motivation in the development of the Platform is to offer a range of techniques for assessing and improving the ability of layouts and signalling rules to efficiently accommodate railway traffic. Depending upon objectives, capacity assessment may be done as a calculation of a single value according to one of predefined formulae or by running a detailed simulation of train movements. The main instrument of capacity improvement is a library of patterns transforming topology graphs and signalling. Patterns are mechanically instantiated and are encoded in the Epsilon transformation language.

As an implementation platform, we have decided to use the Eclipse Integrated Environment and Eclipse Modelling framework (EMF) to realise our DSL. One important consideration was the ability to benefit from the extensive EMF ecosystem which offers toolkits for the model manipulation and the construction of graphical and textual editing tools. A screenshot of the running platform is given in Fig. 4.

The list of axioms presented in this paper is employed by the Platform to automatically check safety of schemas. The primary verification mechanism is constraint solving. An automated tool derives the definition of $(\mathsf{N}, \mathsf{T}, \mathsf{A}, \mathsf{P}, \mathsf{R}, \mathsf{L}, S_P, S_R)$ structure from a track diagram and translates it into an input notation of an SMT-LIB compliant SMT solver. By the standards of SMT solvers, our models are relatively small so a result is reported nearly instantaneously. One downside of applying constrain solving is the absence of any detailed feedback that may be reported to a user should an error be discovered. To compensate for this, whenever an SMT solver detects a problem, the Platform also runs a model checker on the same schema (in our case, ProB [8] that takes Classical B [2] as an input notation) to identify a trace leading to a problem. In most cases, such a trace may be visually replayed by a user to help with the debugging of a schema.

V. CONCLUSION

One decisions we made early in the project was to develop a railway domain specific language (DSL) to allow the engineers to operate in terms of their domain and to hide the formal verification from them as much as possible. The definition of this DSL is in the core of our method. The paper introduces the DSL and shows the way it is used to allow engineers to reason about railway safety properties.

We have developed a compact and, in our opinion, simple domain language for describing railway topologies and signalling rules. The language is strictly defined making it possible to benefit from widely available verification tools. In the scope of this paper we were not able to discuss train movement rules that constitute an important part of the domain language semantics.

Our plans are to further develop the Platform, the SAFECAP DSL and the methods of analysing and improving capacity. Route-based signalling discussed in this paper is only one the several semantics that we plan to support in

[6]The formulation of verification conditions requires a temporary extension of ambit detection capabilities, e.g., $\Omega_A \in \mathsf{A} \rightarrow \mathbb{N}$. Once it is proven that $\forall a \cdot a \in \mathsf{A}\, \Omega_A(a) \leq 1$ one can revert back to normal ambits.

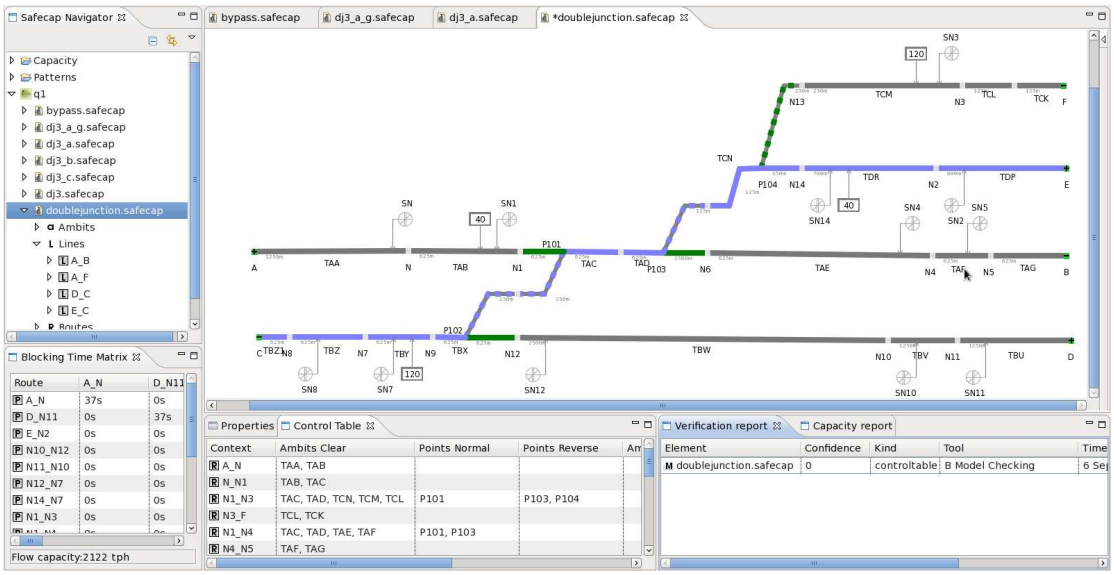


Figure 4. SAFECAP Platform screenshot.

the Platform. An interesting challenge, from the viewpoint of capacity assessment, is modelling the dynamic definition of ambit boundaries that adapt to train progress and speed.

The railway domain has always been one of the areas in which formal methods are successfully deployed and used in a substantial way. For example, in France RATP (a major rail operator) with a considerable experience in formal methods, looks favourably on using formal methods to conform to the development standards that they require. From the mid 90s, in France, RATP, the main rail operator with considerable experience of formal methods, has been approving various developments that use the B method as meeting the development standards RATP require [4]. There are now several tool development companies supporting the use of the B [2] and Event-B [9], [10] methods in the railway domain: ClearSy, Systerel, and Formal Mind. In addition to the B method model-checking has been successfully used by various railway companies working togtehr with universities (see, for example, [11], [12].

ACKNOWLEDGMENT

This research is supported by the EPSRC/RSSB SafeCap project. We are grateful to Simon Chadwick, Markus Roggenbach and Dominic Taylor for their feedback on the earlier versions of this work.

REFERENCES

[1] SAFECAP Project, "Project website," 2012, available at http://safecap.cs.ncl.ac.uk.

[2] J.-R. Abrial, *The B-Book*. Cambridge University Press, 1996.

[3] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier, "Météor: A Successful Application of B in a Large Project," in *Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems-Volume I - Volume I*, ser. FM '99. London, UK, UK: Springer-Verlag, 1999, pp. 369–387.

[4] D. Essamé and D. Dollé, "B in Large-Scale Projects: The Canarsie Line CBTC Experience." in *B*, ser. Lecture Notes in Computer Science, J. Julliand and O. Kouchnarenko, Eds., vol. 4355. Springer, 2007, pp. 252–254.

[5] D. Bjørner, C. George, and S. Prehn, "Scheduling and rescheduling of trains," 1995.

[6] D. Bjørner, "Formal Software Techniques in Railway Systems." pp. 1–12, 2000.

[7] SAFECAP Project, "SAFECAP Platfrom website," 2012, available at http://sf.net/projects/safecap.

[8] M. Leuschel and M. Butler, "ProB: A Model Checker for B," in *Formal Methods Europe 2003*, A. Keijiro, S. Gnesi, and M. Dino, Eds., vol. 2805. Springer-Verlag, LNCS, 2003, pp. 855–874.

[9] J.-R. Abrial, *Modelling in Event-B*. Cambridge University Press, 2010.

[10] ——, "Rigorous development of complex fault-tolerant systems." Springer-Verlag, 2006, ch. Train systems, pp. 1–36.

[11] M. Leuschel, J. Falampin, F. Fritz, and D. Plagge, "Automated property verification for large scale B models with ProB." *Formal Asp. Comput.*, vol. 23, no. 6, pp. 683–709, 2011.

[12] A. Ferrari, G. Magnani, D. Grasso, and A. Fantechi, "Model checking interlocking control tables." in *FORMS/FORMAT*, E. Schnieder and G. Tarnai, Eds. Springer, 2010, pp. 107–115.