Abstract

In designing safety-critical infrastructures such as railway systems, engineers often have to deal with complex and large-scale designs. Formal methods can play an important role in helping automate or check various tasks.

– We focus on static infrastructure models and are interested in checking requirements coming from design guidelines and regulations.

– Our goal is to automate the manual work of the railway engineers through software that is fast enough to do verification-on-the-fly, so to include it in the railway design tools, much like a compiler in an IDE.

– Usability of the verification is achieved through a seamless integration of a fast engine and using RailCNL to allow engineers to read/write the verified rules.

Railway signalling design process

Track and signalling component layout

Railway construction projects rely heavily on computer-aided design (CAD) tools to map out railway station layouts. The various disciplines within a project, such as civil works, track works, signalling, or catenary power lines, work with coordinated CAD models.

Interlocking specification

An interlocking is an intersection of signals and switches to ensure that train movements are performed in a safe sequence. The main purpose of the interlocking specification is to tabulate all possible routes and set conditions for their use. Typical conditions are:

• Switches must be positioned to guide the train to a specified route exit signal.

• Train detectors must show that the route is free of any other trains.

• Conflicting routes, i.e., overlapping routes (or safety zones), must not be in use.

CAD tool integration of RailCons verification engine

A prototype tool was implemented using Autodesk AutoCAD, and XSB Prolog as the Datalog backend. When rule violations are found, the railway engineer will benefit from information about the following:

• Which rule was violated (textual message containing a reference to the source of the rule or a justification in the case of expert knowledge rules).

• Where the rule was violated (identity of objects involved).

Also, classification of rules based on e.g. discipline and severity may be useful in many cases. In the rule databases, this may be accomplished through the use of structured comments. Any violations found are associated with the information in the comments, so that the combination can be used to present a helpful message to the user.

RailCNL – a controlled natural language for railway regulations

To allow the engineers to participate in the verification process, we use the controlled natural language RailCNL for representing properties on a higher level of abstraction, make them closer to the original text while still retaining the possibility for automatic translation into Datalog. This approach has the following advantages:

• RailCNL is domain-specific, i.e., tailored to the types of logical statements needed by the verification engine, and to the regulations terminology. This allows concise and readable expressions, increasing naturality and maintainability.

• The language closely resembles natural language, and can be read by engineers with the required domain knowledge without learning a programming language.

• A separate textual explanation (such as comments used in programming) is not needed for presenting violations textually, as the properties are now directly readable as natural text. Comments could still be used, e.g., to clarify edge cases or for clarifying semantics, as is done in the original texts.

• Statements in RailCNL can be linked to statements in the original text, so that reading them side by side reveals to domain experts whether the CNL paraphrasing of the natural text is valid. If not, they can edit the CNL text.

RailCons software allows checking rules and regulations of static infrastructure inside the CAD environment, while more comprehensive verification and quality assurance can be performed by special-purpose software for other design and analysis activities.

– We limit the verification inside the design environment to static rules and expert knowledge, as these rules require less dynamic information (timetables, rolling stock, etc.) and less computational effort, while still offering valuable insights.

– This situation may be compared to the tool chain for writing computer programs. Static analysis can be used at the detailed design stage (writing the code), but can only verify a limited set of properties. It cannot fully replace testing, simulation and other types of analysis, and must as such be seen as a part of a larger tool chain.
**RailCNS**

**Tools for static railway infrastructure analysis**

**Performance – incremental Datalog**

The common use case for running the railway design CAD tool in general is that one performs a series of small changes. This requires lowering the running time of the verification, hopefully to less than one second, while keeping in mind that our prototype verification tool should eventually be able to scale up to much larger stations, projects spanning several stations, and significantly larger knowledge bases. Exploiting the fact that the design work is incremental, also evaluating the Dat-alog programs incrementally seems to be a promising solution to this challenge.

**RailCNS details**

**RailCNS examples**

**Example 2: (Protocol for railway signal operation)**

Datalog: 

\[ \text{defaultRoute}(a, b, d) \leftarrow \text{signalType}(a, \text{main}) \lor \text{signalType}(b, \text{main}) \lor \text{signalType}(d, \text{main}) \]

**Example 3: (Tunnel interference detection)**

Datalog: 

\[ \text{existsPathWithPath}(\text{Detector}(a, b)) \leftarrow \text{existsPath}(a, b, \text{direction}) \]

**Example 4: (Optimization of maintenance inspection)**

Datalog: 

\[ \text{C} \cdot \text{J} \]
This work addresses a central problem that occurs when designing the layout and control systems for railway stations: Does the station infrastructure have the capacity to handle the amount of trains and the desired traveling times?

We consider the low-level railway infrastructure capacity verification problem, defined as follows:

Given a railway station track plan including signaling components, rolling stock dynamic characteristics, and a performance/capacity specification, verify whether the specification can be satisfied and find a dispatch plan as a witness to prove it.

Solving this problem subsumes the following railway infrastructure design activities:

- **Low-level running time analysis** – verify the time required for getting from point A to point B.
- **Low-level schedulability analysis** – verify frequency of trains arriving at a station, and simultaneous opportunities for crossing, parking, loading, etc.
- **Combinations** – verify running time requirements on schedulable operations.

The planer part of the tool chain is implemented in a CEGAR loop:

![Diagram of CEGAR loop](image)

The planner part of the tool chain is implemented in a CEGAR loop:

1. **Redundancy**: The planner can be used to detect whether some equipment in the design is redundant. If a plan can be found which does require any use of certain pieces of signaling equipment, these pieces can be considered for removal from the design.

2. **Maximal design**: We can find all relevant locations to place signals (maximum schedulability) by placing signals near every switch/branch, turning the signal placement synthesis problem into optimization.

3. **Running time optimization**: Starting from a design schedulable which satisfies schedulability requirements, signals placement can be adjusted locally to achieve timing constraints.

### Constraints

**Allocation of resources**

- Avoiding collisions by exclusive use of resources is the responsibility of the interlocking, which takes requests from the dispatcher for activating *elementary routes*.
  1. **Wait for resources**: track segments and switches in the route path must be free.
  2. **Movable elements**: set switches into position.
  3. **Signals**: show proceed aspect until train has passed.
  4. **Release**: wait for the train to leave, then deallocate.

**Laws of motion**

Trains move according to the laws of motion, accelerating towards the current maximum speed, while also braking in time to meet all speed restrictions ahead $v_i$:

$$v - v_0 \leq a \Delta t \quad v^2 - v_i^2 \leq 2b \Delta s$$

### Specifications

**Operational scenario**

To capture typical performance and capacity requirements in construction projects, we define an operational scenario $S = (V, M, C)$ as follows:

1. A set of *vehicle types* $V$, each defined by a length $l$, a maximum velocity $v_{\text{max}}$, a maximum acceleration $a$, and a maximum braking retardation $b$.
2. A set of *movements* $M$, each defined by a vehicle type and an ordered sequence of visits. Each visit is a set of alternative locations $l_i$ and an optional minimum dwelling time $t_d$.
3. A set of *timing constraints* $C$, which are two visits $g_i, g_j$ and an optional numerical constraint $t_k$ on the minimum time between visit $g_i$ and $g_j$. The two visits can come from different movements. If the time constraint $t_k$ is omitted, the visits are only required to be ordered, so that $t_k < t_{k'}$.

### Case studies

**Infrastructure**

- A Norwegian railway infrastructure manager Bane NOR supplies a rolling infrastructure model of the whole national railway network from which we have extracted examples.

Source: Bane NOR SE, Norway.

**Performance table**

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>Property</th>
<th>Result</th>
<th>$t_{\text{run}}$</th>
<th>$t_{\text{inst}}$</th>
<th>$t_{\text{total}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolling stock</td>
<td>Two track (14 elem.)</td>
<td>Sat.</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Kolmøen (119 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Arna-CAD (295 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Geir 723 (17 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Geir 414 (198 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Geir 523 (479 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train type</td>
<td>Train type</td>
<td>Performance</td>
<td>$t_{\text{run}}$</td>
<td>$t_{\text{inst}}$</td>
<td>$t_{\text{total}}$</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Two track (14 elem.)</td>
<td>Sat.</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Kolmøen (119 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Arna-CAD (295 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Geir 723 (17 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Geir 414 (198 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Geir 523 (479 elem.)</td>
<td>Sat.</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**TABLE 1**: Verification performance on test cases, including Bane NOR (BN) and RailCOMPLETE (CAD) infrastructure models. The number of elementary routes (elements) indicates the model’s size. $t_{\text{run}}$ is the number simulator runs, $t_{\text{inst}}$ the time in seconds spent in SAT solver, $t_{\text{total}}$ the time in seconds spent in DES, and $t_{\text{total}}$ the total calculation time in seconds.

---