RailCons

Tools for static railway infrastructure analysis RailCons

RailCOMPLETE

UiO : University of Oslo

Abstract

In designing safety-critical infrastructures s.a. 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.

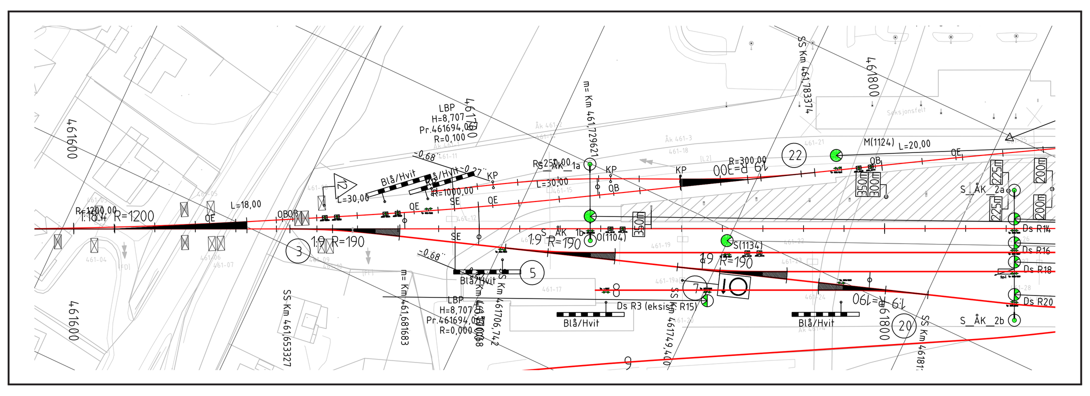
– Our goal is to automate the manual work of the railway engineers through software that is fast enough to do verification on-the-fly, so to include it in the railway design tools, much like a compiler in an IDE.

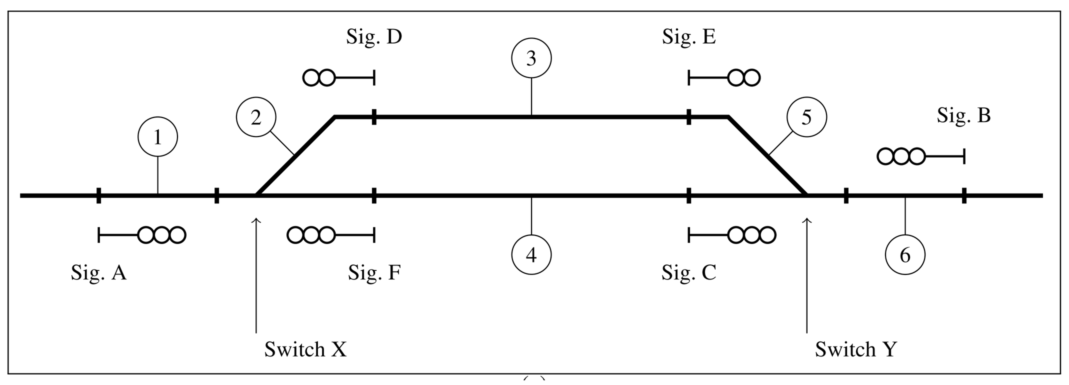
– Usability of the verification is achieved through a seamless integration of a fast engine and using RailCNL to allow engineers to read/write the verified rules.

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 interconnection 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.

Route	Start	End	Sw. pos	Detection sections	Conflicts
AC	A	C	X right	1, 2, 4	AE, BF
AE	A	E	X left	1, 2, 3	AC, BD
BF	B	F	Y left	4, 5, 6	AC, BD
BD	B	D	Y right	3, 5, 6	AE, BF

Links and Collaborators

RailCons web page

RailCOMPLETE® web page

Video vimeo.com/221549125

CHALMERS

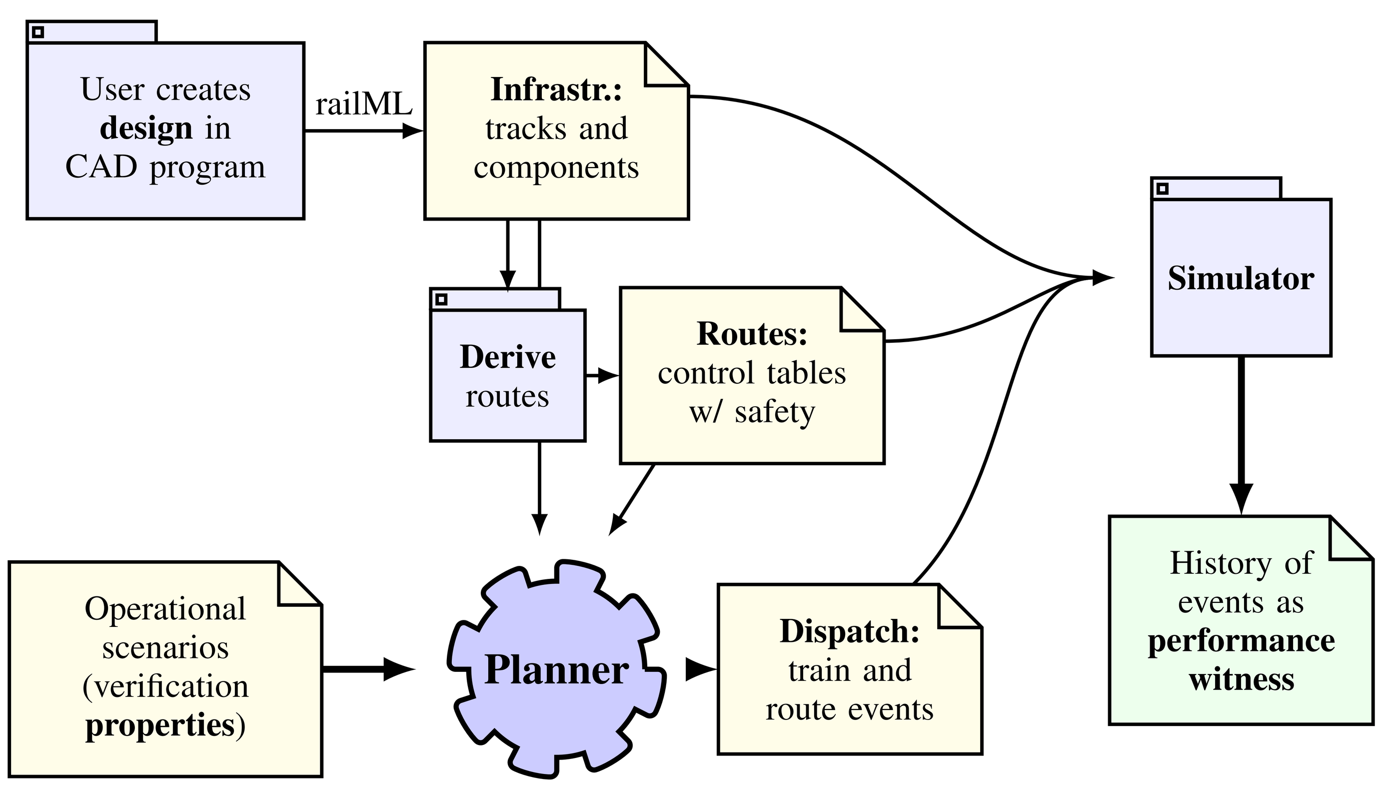
OXFORD

Presenters

Bjørnar S. Luteberget
Claus Feyling
{bjlut,clfey}@railcomplete.no
RailCOMPLETE AS

Christian Johansen
Martin Steffen
{cristi,msteffen}@ifi.uio.no
University of Oslo

Tool chain for railway design



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.

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).

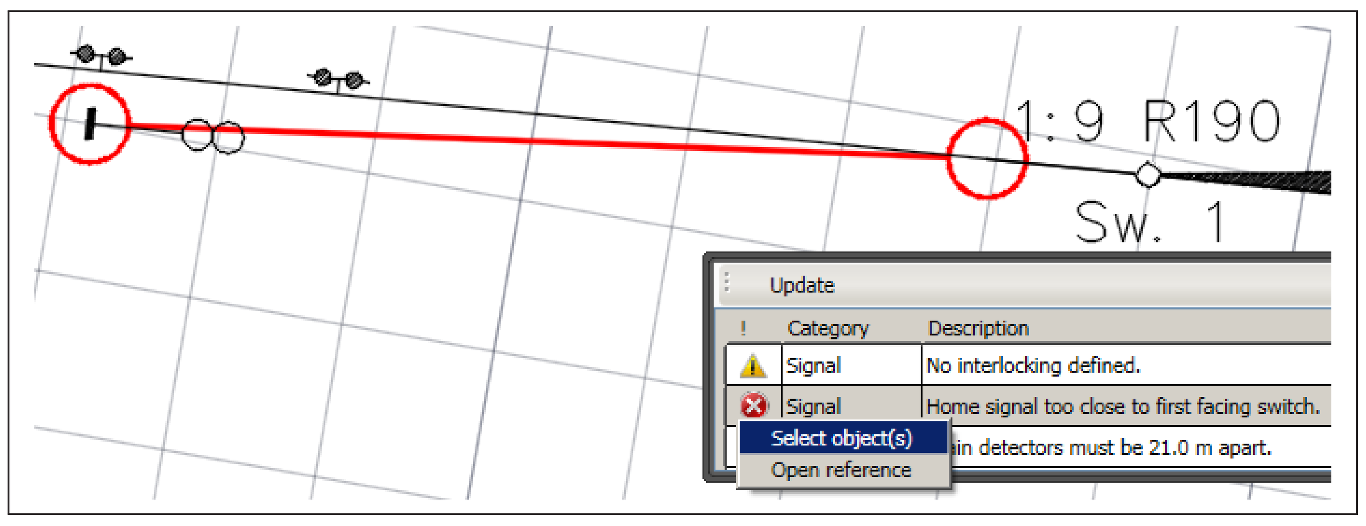
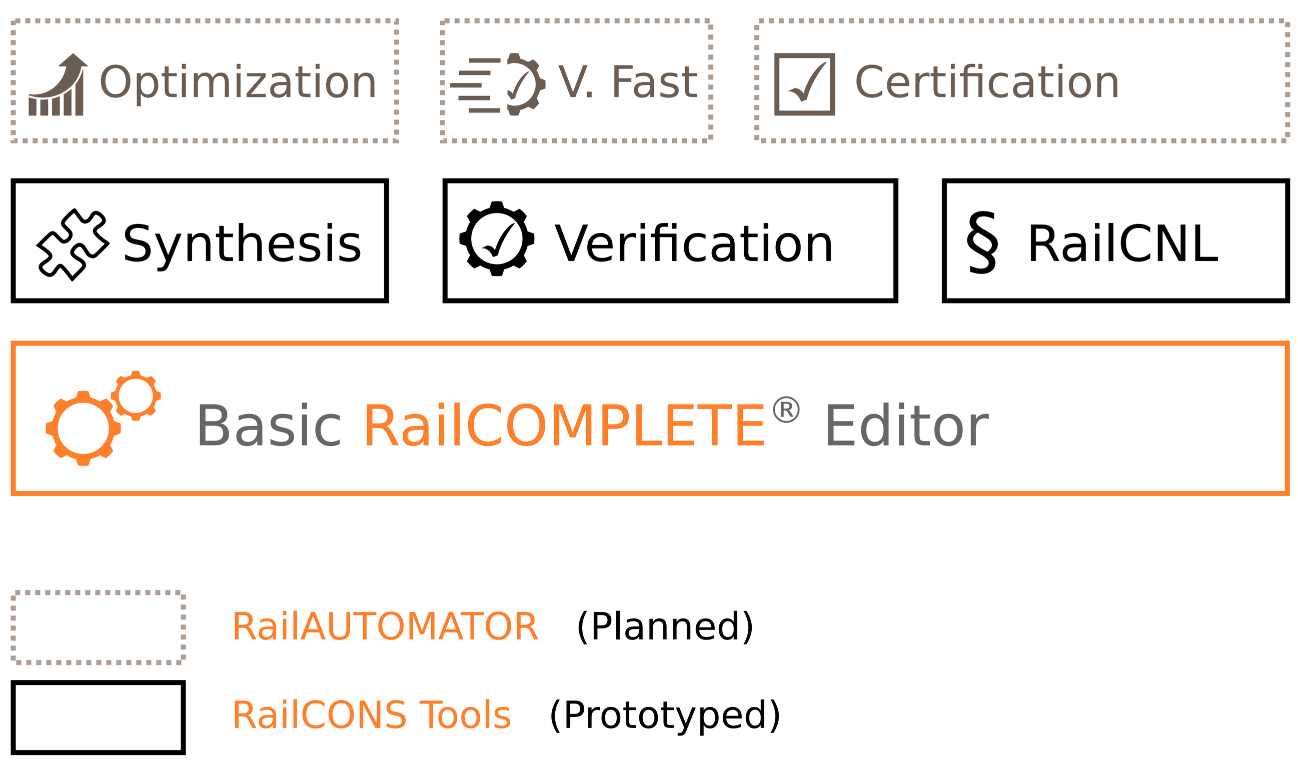


Figure 1: Prototype static verification tool used in AutoCAD

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.

Railway design automation Modules



Existing, prototyped, and planned modules and software and their dependency.

- RailCOMPLETE® offers a good basis, being built on top of CAD, providing various means for digital engineering of railway infrastructure designs.
- RailCons modules are prototyped and integrated in RailComplete, offering advanced automated techniques to the railway engineer.
- The planned modules would be using and enhancing the existing RailCons.

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, making 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 both to the types of logical statements needed by the verification engine, and to the regulations terminology. This allows concise and readable expressions, increasing naturalness and maintainability.
- The language closely resembles natural language, and can be read by engineers with the required domain knowledge without learning a programming language.

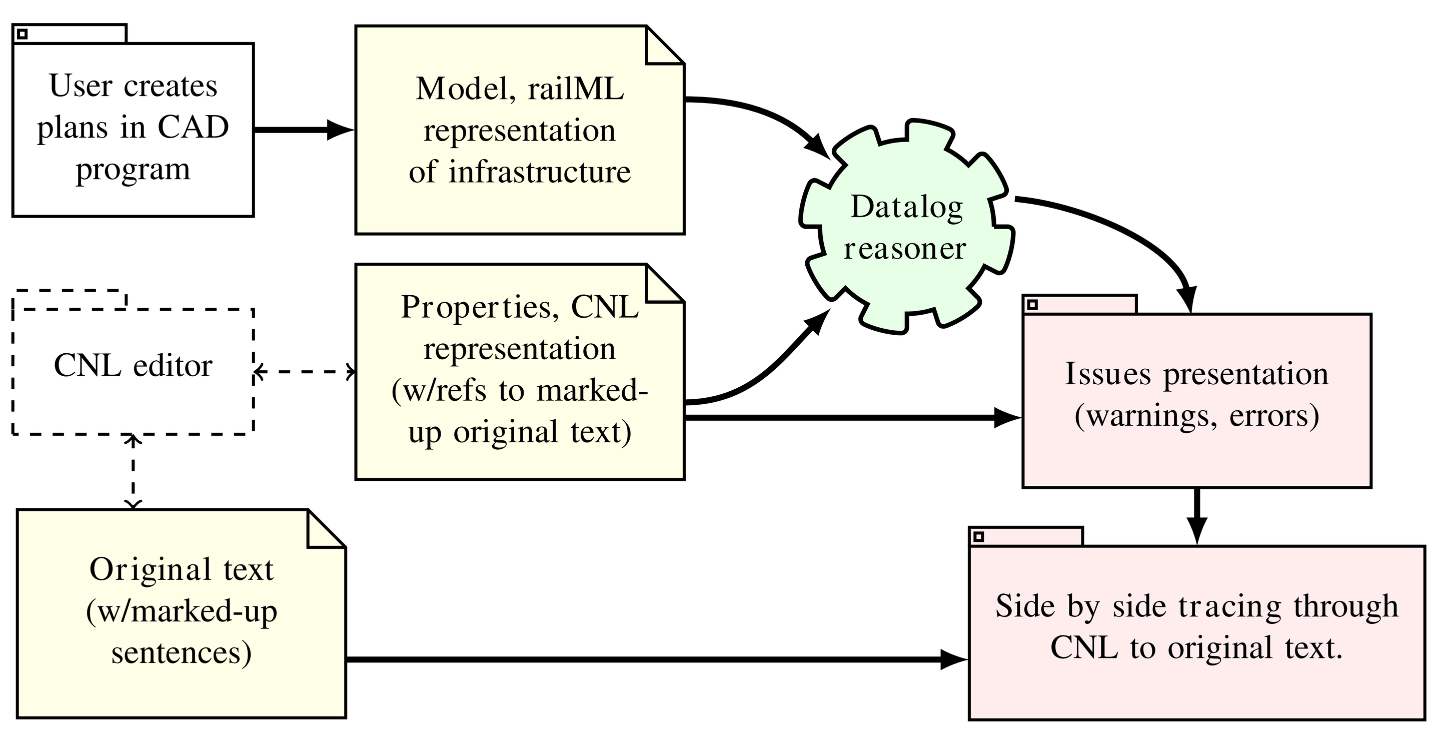


Figure 2: Verification process overview. Models come directly from the CAD program, which engineers are already familiar with. Properties come from paraphrasing the regulations using CNL, which in turn are translated into Datalog. The reasoner outputs issues (warnings and errors) which are presented to the user in the CAD program by highlighting the objects involved in the violation. Issues are traced back to the original text (i.e. the regulations) though identifiers on the marked-up sentences.

- 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 to clarify 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.

See our paper **Efficient verification of railway infrastructure designs against standard regulations**, In *Journal on Formal Methods in System Design*, DOI 10.1007/s10703-017-0281-z

RailCOMPLETE

RailCons

Tools for static railway infrastructure analysis RailCons

UiO : University of Oslo

Datalog logic programming for static railway verification

Datalog

The Datalog language is a first-order conjunctive queries logic extended with *least fixed points*. Datalog uses the Prolog convention of interpreting identifiers starting with a capital letter as variables, and other identifiers as constants, e.g., the clause

$$\mathrm{a}(X,Y) \coloneq \mathrm{b}(X,Z),\ \mathrm{c}(Z,Y)$$

has the meaning of

$$\forall x,y : ((\exists z : (b(x,z) \wedge c(z,y))) \rightarrow a(x,y))\,.$$

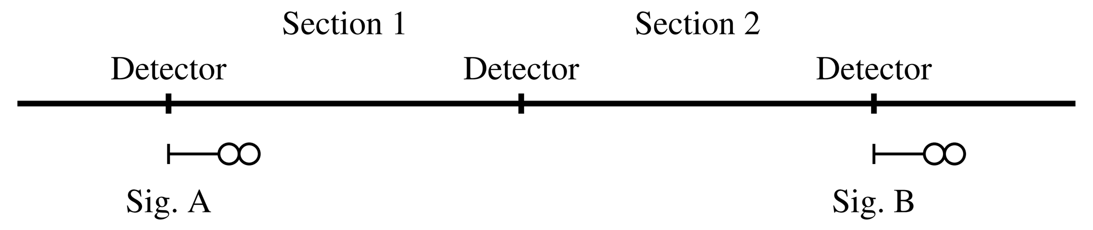
We can define railway objects to be connected through the graph of tracks:

$$\begin{aligned}
\mathrm{directlyConnected}(a,b) &\coloneq \mathrm{track}(t),\ \mathrm{belongsTo}(a,t),\\
&\qquad \mathrm{belongsTo}(b,t).\\
\mathrm{connected}(a,b) &\coloneq \mathrm{directlyConnected}(a,b).\\
\mathrm{connected}(a,b) &\coloneq \mathrm{directlyConnected}(a,x),\\
&\qquad \mathrm{connection}(x,c),\\
&\qquad \mathrm{connected}(c,b).
\end{aligned}$$

Here, the *connection* predicate contains switches and other connection types. Further details of relevant predicates are given in the sections below.

Railway regulations representation

Interlocking: Track clear on route Each pair of adjacent train detectors defines a track detection section. For any track detection sections overlapping the route path, there shall exist a corresponding condition on the activation of the route.



Tabular interlocking:

Route	Start	End	Sections must be clear
AB	A	B	1, 2

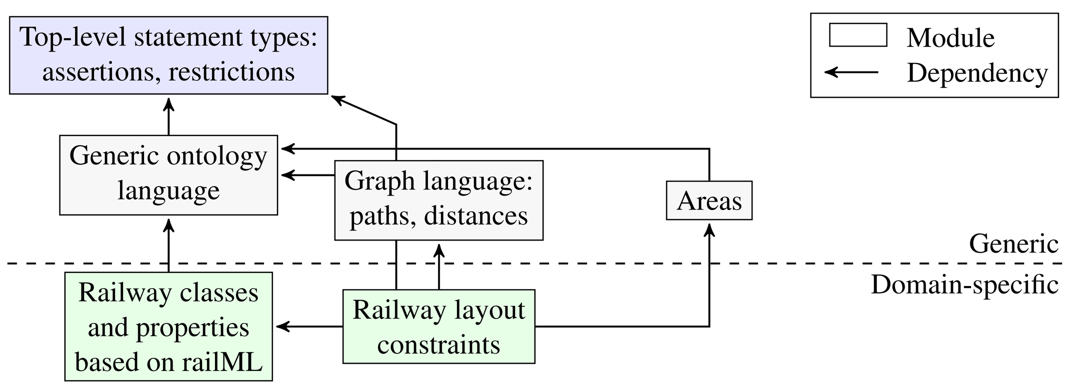
This property can be represented as follows:

$$\begin{aligned}
\mathrm{existsPathWithDetector}(a,b) &\leftarrow \exists d : \mathrm{following}(a,b,d) \wedge\\
&\qquad \mathrm{trainDetector}(x) \wedge\\
&\qquad \mathrm{between}(a,x,b).\\
\mathrm{ruleViolation}(r,d_a,d_b) &\leftarrow\\
&\quad \mathrm{detectionSectionOverlapsRoute}(r,d_a,d_b) \wedge\\
&\quad \neg\mathrm{detectionSectionCondition}(r,d_a,d_b).
\end{aligned}$$

RailCNL details

RailCNL modules

RailCNL has a modular design where domain-specific constructs are separated from generic ones. However, CNL modules are not always trivially composable, and care must be taken to retain naturalness while avoiding ambiguity when increasing the complexity of the language.



RailCNL examples

Example 2 (Parse tree for a railway layout statement.)

CNL: *Distance from an entry signal to first facing switch must be greater than 200.0 m.*

AST:
```
DistanceRestriction Obligation
  (SubjectClass (StringClassAdjective "entry"
    (StringClass "signal")))
  (FirstFound FacingSwitch)
  (Gt (MkValue (StringTerm "200.0m")))
```

Example 3 (Datalog translation of an ontology restriction.)

CNL: *A signal must have height 4.0m or 4.5m.*

AST:
```
OntologyRestriction Obligation
  (SubjectClass
    (StringClassNoAdjective (StringClass "signal")))
  (ConditionPropertyRestriction (MkPropertyRestriction
    (StringProperty "height")
    (OrRestr (Eq (MkValue (StringTerm "4.0m")))
             (Eq (MkValue (StringTerm "4.5m"))))))
```

Datalog:
```
r1_found(Subj0) :- signal(Subj0), height(Subj0, 4.0).
r1_found(Subj0) :- signal(Subj0), height(Subj0, 4.5).
r1_obl(Subj0) :- signal(Subj0), !r1_found(Subj0).
```

Tooling for RailCNL integrated in RailCOMPLETE

Paraphrasing view

Generelle krav

Utførelse

På jernbaneskilt er det naturlig å skille mellom høyest mulig og en lavere refleksevne. Dette fremgår av tegningen for det enkelte skilt.

For å unngå speilrefleks, bør skilt og merker ikke settes opp vinkelrett (90º) på sporet, men dreies 4º ut.

ID: skilt1 — Definisjon.

RailCNL: En skilt har refleksevne høy eller lav.

AST: Constraint (SubjectClass (StringClass "skilt")) (ConditionPropertyRestriction (MkPropertyRestriction (StringProperty "refleksevne") (OrRestr (Eq (MkValue (StringTerm "høy"))) (Eq (MkValue (StringTerm "lav"))))))

ID: skilt2 — Automatisk verifisering.

RailCNL: En skilt bør ha sporvinkel som er større enn 94.

Datalog:

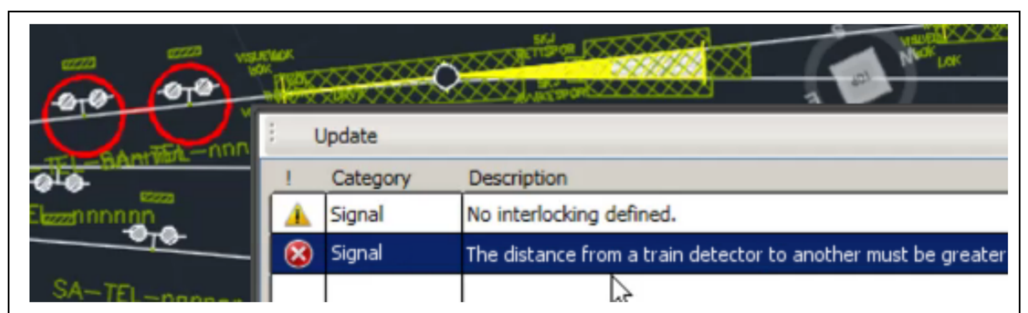
- skilt2_found(Subj0) :- skilt(Subj0), sporvinkel(Subj0, Val2), Val2 > 94.
- skilt2_recommendation(Subj0) :- skilt(Subj0), !skilt2_found(Subj0).

Requirements tracing – informal text and formalized paraphrases are linked together, so that experienced engineers can examine the correctness of corresponding formulations, and inspect the coverage of regulations.

- Defintion (gray), not a normative phrase, but still formalized by using it as a definition for a new predicate.
- Uncovered (red), normative phrase which is not covered by RailCNL.
- Partially covered (yellow), normative phrase which is represented by RailCNL but not covered by automatic verification.
- Covered (green), normative phrases for which CAD models are automatically checked.

Tracing view

CAD program showing issues in layout plan



↕

CNL debug view paraphrased text and translations

ID: detector_1

RailCNL: The distance from an axle counter to another must be larger than 21.0m.

AST: DistanceRestriction Obligation (SubjectClass (StringClassNoAdjective (String "axle_counter"))) (AnyFound (AnyDirectionObject SubjectOtherImplied)) (Gt (MkVal

Datalog: detector_1_start(Subj0, End, Dist) :- trainDetector(Subj0), next(Subj0, En

↕

Original text highlighting source of paraphrased text

Placement and length

This section gives generalized rules for placement and length for train detection systems and its relationship to other infrastructure components. Detailed requirements are given in appendices.

General

a) No detection sections shall be shorter than 21 meters.

b) No dead zone shall be longer than 3 meters.

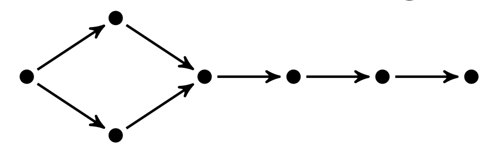
Requirements tracing – regulations violations detected in the CAD model can be traced back to RailCNL phases and their corresponding original informal texts. Each rule violation in the CAD program has a corresponding menu which can open a RailCNL debug view or the paraphrasing view.

– Experienced engineers can trace back from errors to check the correctness of each step of process from original text to CAD program warnings and debug the verification itself.

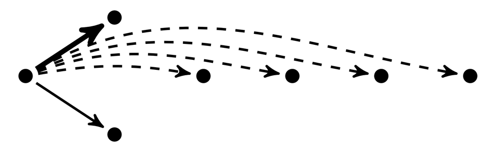
– Inexperienced engineers can trace back from errors in their design to the regulations, which give context and justification for requirements.

Performance – incremental Datalog

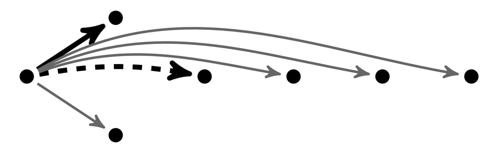
The common use case for running the railway design CAD tool in general is that one performs a series of small changes. This requires lowering the running time of the verification, hopefully to less than one second, while keeping in mind that our prototype verification tool should eventually be able to 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 Datalog programs incrementally seems to be a promising solution to this challenge.



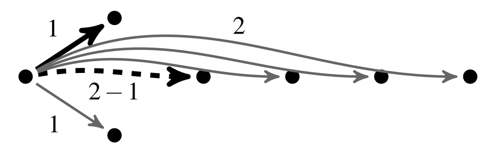
(a) Edge relation visualized as arrows between objects (each element is an arrow $e(a,b)$).



(b) DRed algorithm: removing one edge (thick line) triggers re-evaluation of many dependent edges (dashed lines)



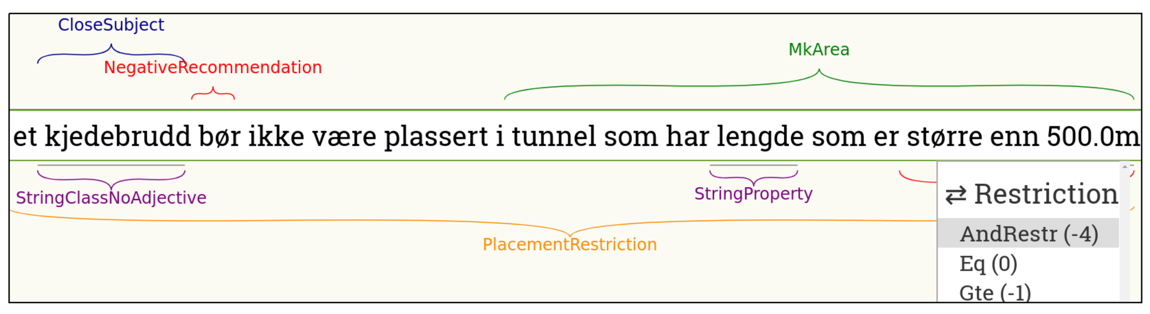
(c) FBF algorithm: removing one edge (thick line) causes re-evaluation of dependent edge (thick dashed line), but confirmation that this edge is still valid stops further propagation.



(d) Counting approach: removing one edge (thick line) causes re-evaluation of dependent edge (thick dashed line), but because this edge has multiple derivations, it is still valid, and propagation can stop. Note that a pure counting approach is not sufficient in this case because of the recursive reachability rule.

			Testing station	Arna phase *A*	Arna phase *B*
Relevant components			15	152	231
Interlocking routes			2	23	42
Datalog input facts			85	8283	9159
XSB:					
Non-incr.:	Time:	(*s*)	0.015	2.31	4.59
	Memory	(MB)	20	104	190
Incr. baseline:	Time	(*s*)	0.016	5.87	12.25
	Memory	(MB)	21	1110	2195
Incr. update:	Time	(*s*)	0.014	0.54	**0.61**
	Memory	(MB)	22	1165	2267

Predictive text editor



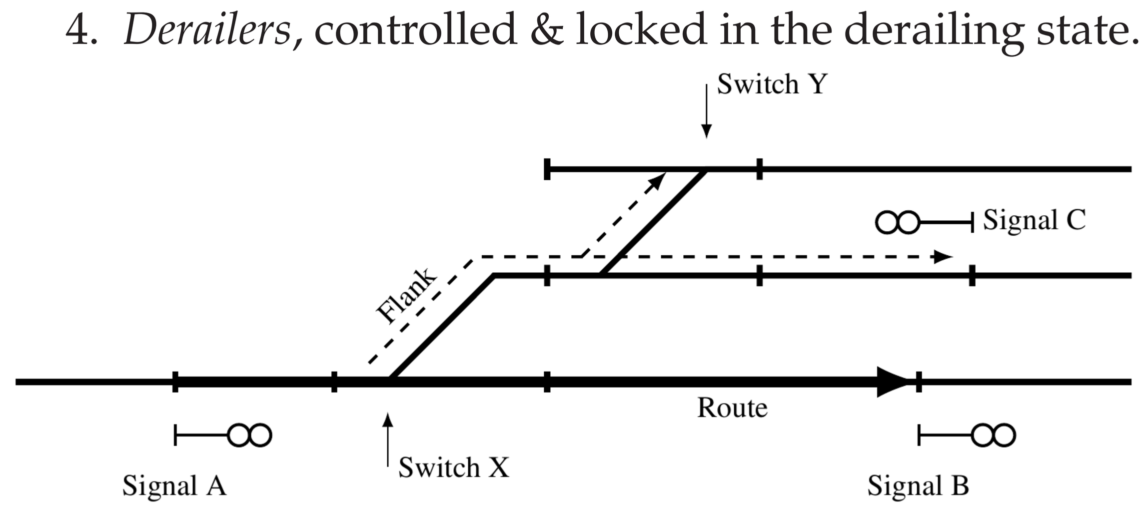
Rule authoring tool with free-form input using the structure of the grammar to provide:

- Syntax checks – parsing status of phrase.
- Predictive – suggestions for completing a phrase.
- Chunked parsing – identifying partially well-formed phrases and suggestions for combining chunks.
- Syntax highlighting and structural information – partial structure of parse tree is displayed over the text.
- Language exploration – a menu provides alternative structures, allowing users to learn more about the language from modifying examples.

Properties / Regulations

Interlocking: Flank protection. A train route shall have flank protection, i.e., for each switch in the route path and its position, the paths starting in the opposite switch position defines the *flank*. Each flank path is terminated by the first flank protection object encountered along the path. The following objects can give flank protection:

1. *Main signals*, by showing the *stop* aspect.
2. *Shunting signals*, by showing the *stop* aspect.
3. *Switches*, controlled and locked in the position which does not lead into the path to be protected.
4. *Derailers*, controlled & locked in the derailing state.



While the indicated route is active (A to B), switch X needs flank protection for its left track. Flank protection is given by setting switch Y in right position and setting signal C to *stop*.

Presented at **SEFM 2017: Software Engineering and Formal Methods**, LNCS 10469, DOI 10.1007/978-3-319-66197-1_6; and at **RSSRail 2017**.

Co-authors: John J. Camilleri and Gerardo Schneider from Chalmers University of Technology.

RailCons

Design-time capacity – SAT modulo Discrete Event Simulation

RailCOMPLETE — UiO : University of Oslo

Capacity

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?

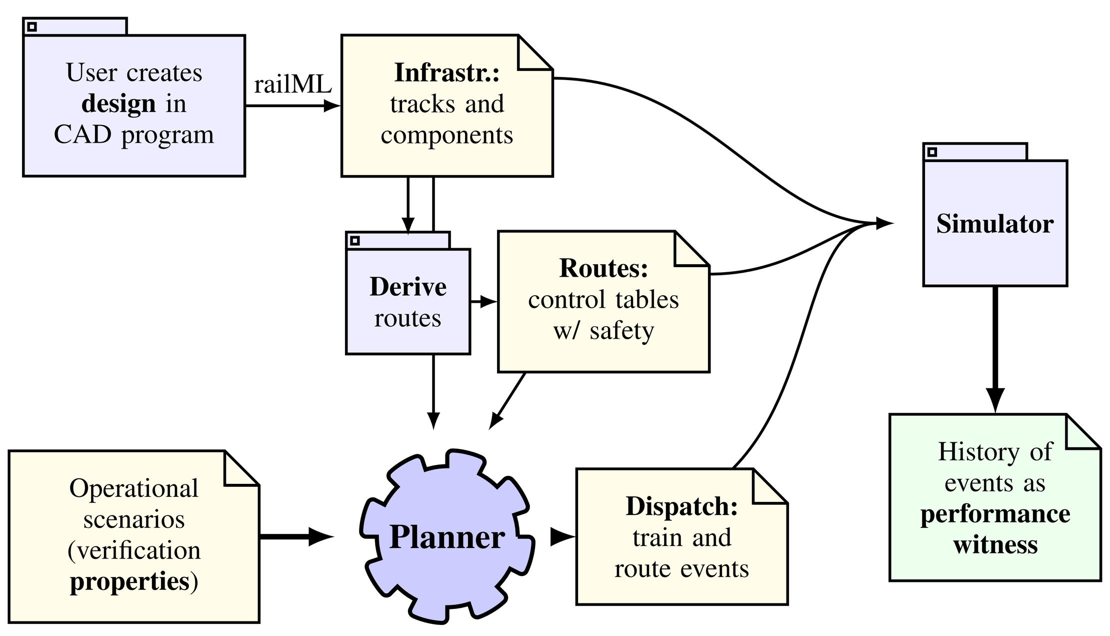
We consider **the low-level railway infrastructure capacity verification problem**, defined as follows:

> 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.

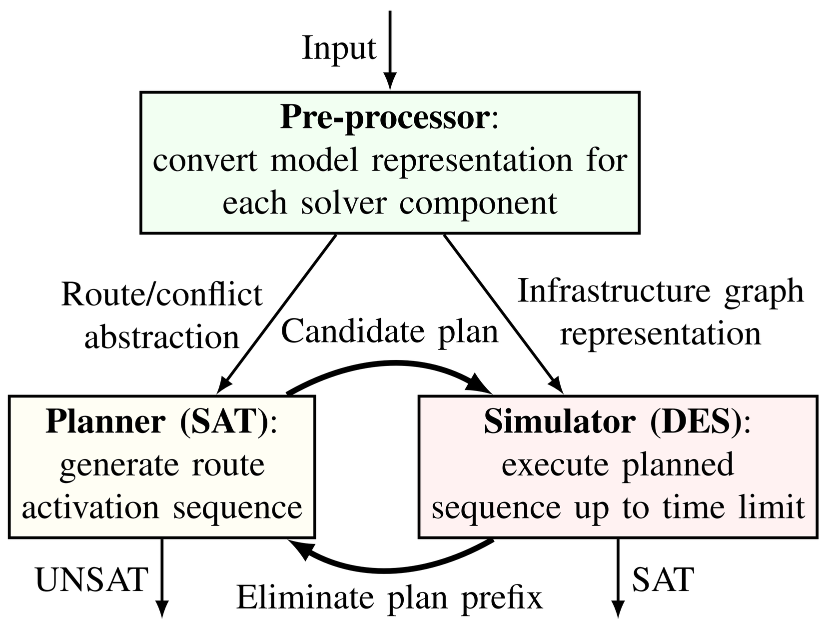
Solving this problem subsumes the following railway infrastructure design activities:

- 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.

Solver architecture

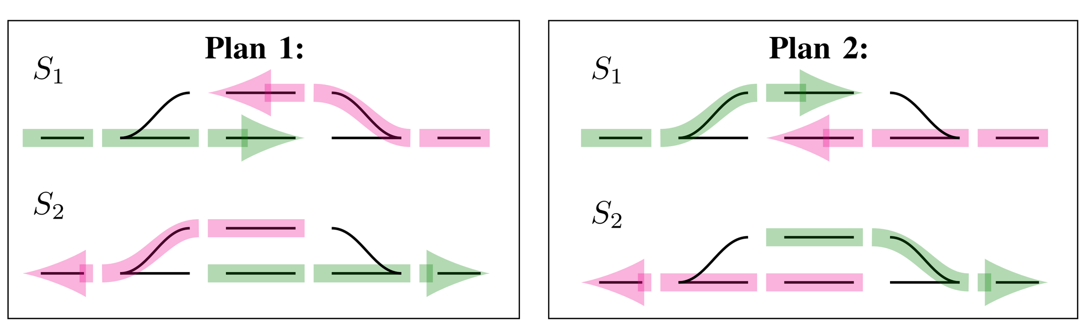


The planner part of the tool chain is implemented in a CEGAR loop:



Synthesis / optimization

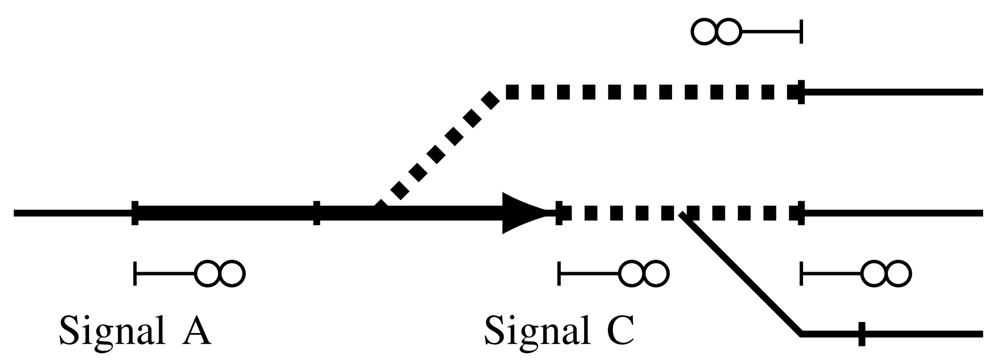
1. **Redundancy:** The planner can be used to detect whether some equipment in the design is redundant. If a plan can be found which does require any use of certain pieces of signalling equipment, these pieces can be considered for removal from the design.
2. **Maximal design:** we can find all relevant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

Physical infrastructure

Trains travel on a network of railway tracks which have **(1) physical properties** such as length, gradient, curvature, etc., **(2) topology** determined by the location of switches (branches), **(3) equipment** such as signals and detectors, and **(4) sight** information showing from which parts of tracks a signal is visible.



Communication constraints

After movement has been allowed by the control system, the driver must be informed of this fact.

- Communication is limited by how many different aspects the lamps can show. To avoid high-speed trains slowing down at every signal, several consecutive elementary routes can be signaled in advance using so-called distant signals.
- Automatic train protection systems (ATP)
- European Rail Traffic Management System (ERTMS) uses long-range radio for communication, effectively removing the communication constraint.

Allocation of resources

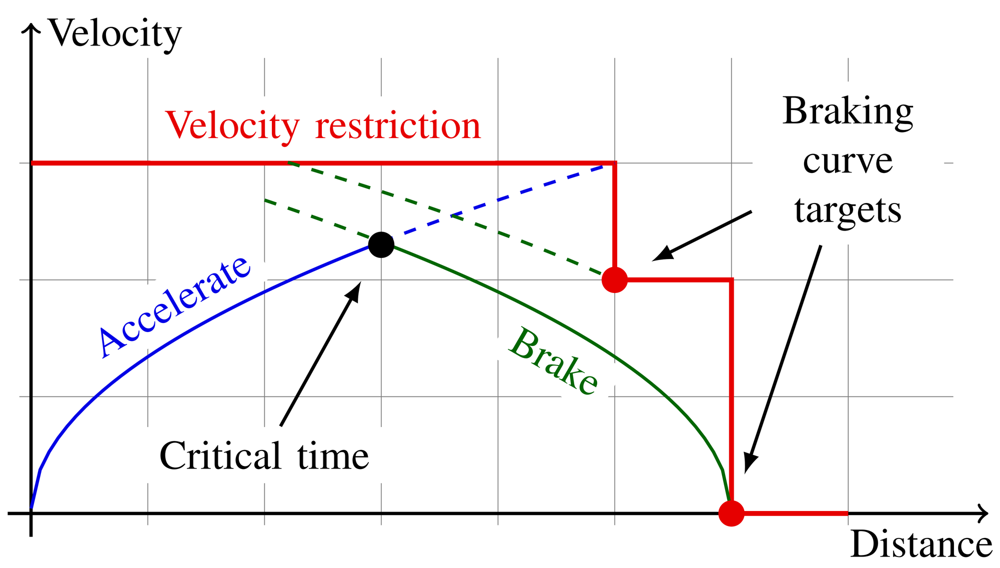
Avoiding collisions by exclusive use of resources is the responsibility of the interlocking, which takes requests from the dispatcher for activating **elementary routes**.

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:

$$v - v_0 \leq a\Delta t \qquad v^2 - v_i^2 \leq 2bs_i$$



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 $v_{\max}$, 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 q 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 q_a, q_b, and an optional numerical constraint t_c on the minimum time between visit q_a and q_b. The two visits can come from different movements. If the time constraint t_c is omitted, the visits are only required to be ordered, so that $t_{q_a} < t_{q_b}$.

Running time

An expectation of how long it should take for a train to travel between two locations.

```
movement passengertrain {
  visit #a [b1]; visit #b [b2] }
timing a <90.0 b
```

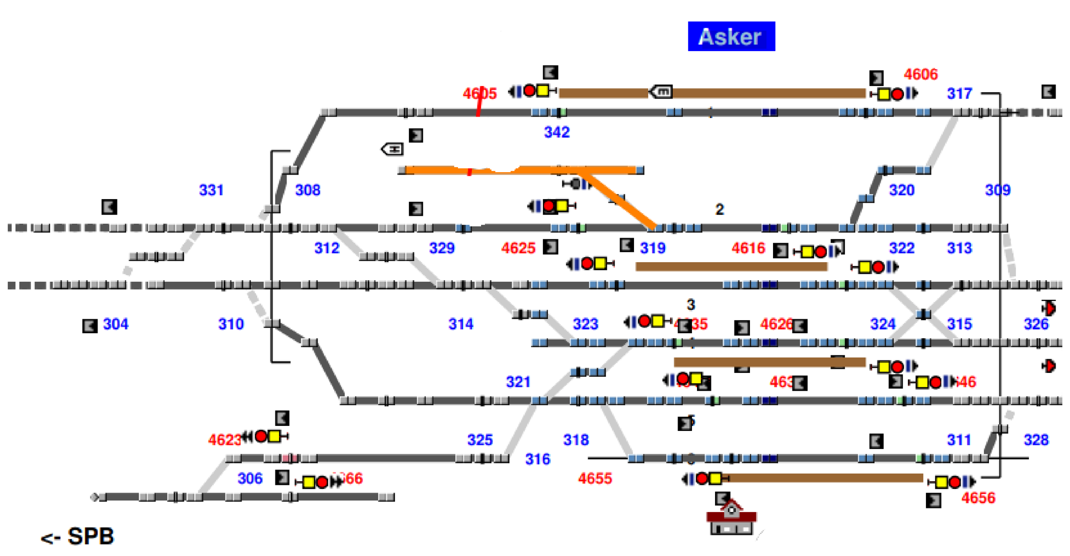
Crossing

Trains traveling in *opposite directions* can visit a station simultaneously.

```
movement passengertrain {
  visit #p_in [b1]; visit #p_out [b2] }
movement goodstrain {
  visit #g_in [b2]; visit #g_out [b1] }
timing p_in < g_out
timing g_in < p_out
```

Case studies

Infrastructure



Source: Bane NOR SF, Norway.

Case studies were performed using:

1. an infrastructure model from the Arna construction project made using the RailCOMPLETE® railway signalling CAD software.
2. the Norwegian railway infrastructure manager Bane NOR supplies a railML infrastructure model of the whole national railway network from which we have extracted examples.

Performance table

Infrastructure	Property	Result	n_{DES}	t_{SAT}	t_{DES}	t_{total}
Two track (14 elem.)	Run.time	Sat.	1	0.01	0.00	0.01
	Frequency	Sat.	1	0.01	0.00	0.01
	Overtaking 2	Sat.	1	0.00	0.00	0.01
	Overtaking 3	Unsat.	0	0.01	0.00	0.01
	Crossing 3	Unsat.	0	0.01	0.00	0.01
Kolbotn (BN) (56 elem.)	Run. time	Sat.	2	0.01	0.00	0.02
	Overtake 4	Sat.	1	0.05	0.00	0.06
	Overtake 3	Unsat.	0	0.05	0.00	0.06
Eidsvoll (BN) (64 elem.)	Run. time	Sat.	2	0.01	0.00	0.02
	Overtake 2	Sat.	1	0.08	0.00	0.08
	Crossing 3	Sat.	1	0.04	0.00	0.04
	Crossing 4	Unsat.	0	0.21	0.00	0.21
Asker (BN) (170 elem.)	Overtaking 2	Sat.	1	0.20	0.00	0.21
	Overtaking 3	Unsat.	1	0.73	0.00	0.74
	Crossing 4	Sat.	0	0.75	0.00	0.77
Arna (CAD) (258 elem.)	Run. time	Sat.	1	0.02	0.00	0.04
	Overtaking 2	Sat.	1	0.50	0.00	0.51
	Overtaking 3	Sat.	1	1.43	0.00	1.45
	Crossing 4	Sat.	1	1.73	0.00	1.74
Gen. 3x3 (74 elem.)	High time	Sat.	1	0.01	0.00	0.01
	Low time	Unsat.	27	0.18	0.01	0.19
Gen. 4x4 (196 elem.)	High time	Sat.	1	0.01	0.00	0.03
	Low time	Unsat.	256	2.08	0.26	2.34
Gen. 5x5 (437 elem.)	High time	Sat.	1	0.06	0.00	0.09
	Low time	Unsat.	3125	38.89	4.35	43.24

TABLE I: Verification performance on test cases, including Bane NOR (BN) and RailCOMPLETE (CAD) infrastructure models. The number of elementary routes (*elem.*) indicates the model's size. n_{DES} is the number simulator runs, t_{SAT} the time in seconds spent in SAT solver, t_{DES} the time in seconds spent in DES, and t_{total} the total calculation time in seconds.

Presented at **FMCAD 2018: Formal Methods in Computer-Aided Design** in Austin, Texas, DOI 10.23919/FMCAD.2018.8603003. ***Co-author: Koen Claessen*** from Chalmers.