Rule-based Consistency Checking of Railway Infrastructure Designs

Bjørnar Luteberget, Christian Johansen,
and Martin Steffen

12th International Conference on
integrated Formal Methods

June 3, 2016

UiO : University of Oslo

RailCOMPLETE

Talk outline

1. Background and motivation
2. Embedding railML in CAD
3. Verification of regulations using a Datalog language
4. Prototype tool integrating this verification into existing engineering tools (RailCOMPLETE)

Railway verification and formal methods

- Railway systems: large-scale, safety-critical infrastructure
- High safety requirements: SIL 4 for passenger transport
- Increasingly computerized components
- Typical use of formal methods in railways: model checking of control systems

Objective

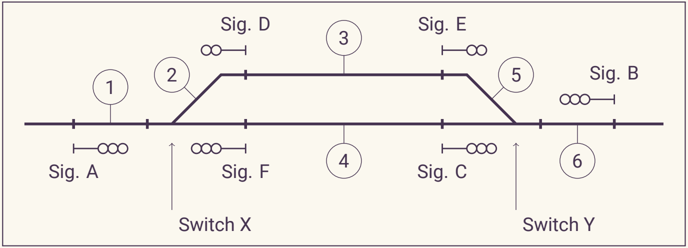
Given a railway signalling and interlocking design,
verify that it complies with regulations.

Secondary objectives:

- Integrate with engineering/design tools
  - On-the-fly verification ("lightweight")
  - Usable for engineers who are not formal methods experts
- Find suitable language for expressing regulations

> "Formal methods will never have a significant impact until they can be used by people that don't understand them."
>
> — (attributed to) Tom Melham

Railway designs for signalling and interlocking



(a) Track and signalling component layout

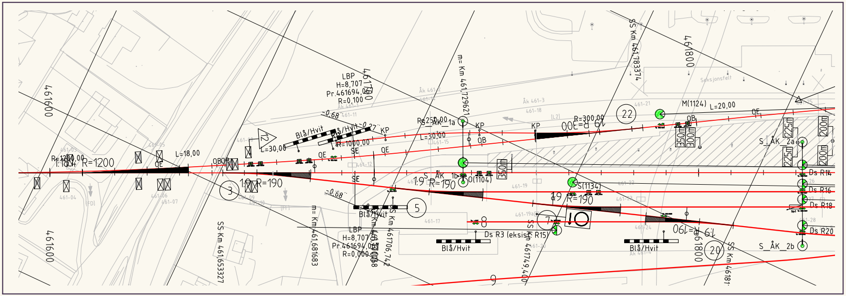
Route	Start	End	Sw. pos	Detection sections	Conflicts
AC	A	C	X right	1, 2, 4	AE, BF
AE	A	E	X left	1, 2, 3	AC, BD
BF	B	F	Y left	4, 5, 6	AC, BD
BD	B	D	Y right	3, 5, 6	AE, BF

(b) Tabular interlocking specification

CAD tools

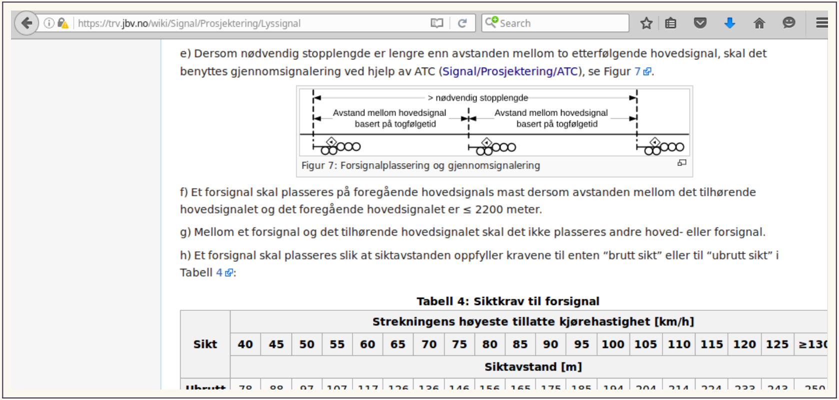
Producing design documentation for construction

- Computer-aided design (CAD) tools are widely used for all types of construction projects
- Originally a software-assisted way of producing paper drawings (now PDFs), but many extensions add structured data to integrate with analysis tools



Technical regulations

- In our case study: Norwegian regulations from infrastructure manager Jernbaneverket
- Static kind of properties, often related to object properties, topology and geometry (examples later)



https://trv.jbv.no/wiki/Signal/Prosjektering/Lyssignal

Search

e) Dersom nødvendig stopplengde er lengre enn avstanden mellom to etterfølgende hovedsignal, skal det benyttes gjennomsignalering ved hjelp av ATC (Signal/Prosjektering/ATC), se Figur 7.

> nødvendig stopplengde

Avstand mellom hovedsignal basert på togfølgetid

Avstand mellom hovedsignal basert på togfølgetid

Figur 7: Forsignalplassering og gjennomsignalering

f) Et forsignal skal plasseres på foregående hovedsignals mast dersom avstanden mellom det tilhørende hovedsignalet og det foregående hovedsignalet er ≤ 2200 meter.

g) Mellom et forsignal og det tilhørende hovedsignalet skal det ikke plasseres andre hoved- eller forsignal.

h) Et forsignal skal plasseres slik at siktavstanden oppfyller kravene til enten “brutt sikt” eller til “ubrutt sikt” i Tabell 4:

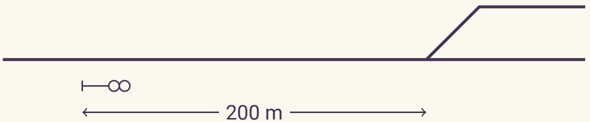
Tabell 4: Siktkrav til forsignal

Sikt	Strekningens høyeste tillatte kjørehastighet [km/h]																	
	40	45	50	55	60	65	70	75	80	85	90	95	100	105	110	115	120	125	≥130
	Siktavstand [m]																		
Ubrutt	78	88	97	107	117	126	136	146	156	165	175	185	194	204	214	224	233	243	250

Technical regulations

Example from regulations:

- A *home main signal* shall be placed at least 200 m in front of the first controlled, facing switch in the entry train path.



- Can be classified as follows:
  - Object properties
  - Topological layout properties
  - Geometrical layout properties
  - Interlocking properties

Related work

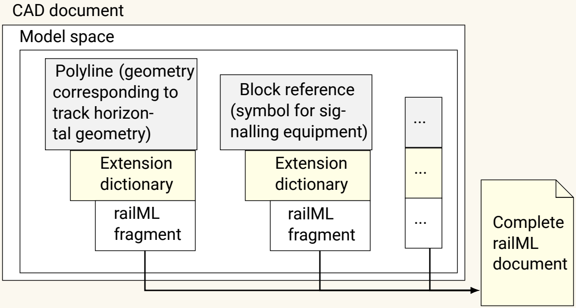
- Safety of interlocking has been extensively studied in the formal methods literature
  - model checking interlocking tables, (Ferrari et al. FORMS/FORMAT 2010)
  - verified code generation, (Borälv & Stålmarck, 1999)
- These works focus on *dynamic* aspects of railway operation
- In contrast, we focus on *static* design properties, less computationally expensive
  - Most close contribution to ours, uses semantic technologies (Lodemann et al. 2013)
- We are concerned with automating manual tasks performed by railway engineers, not directly verifying safety properties

Talk outline

1. Background and motivation
2. Embedding railML in CAD designs
3. Verification of regulations using a Datalog language
4. Prototype tool integrating this verification into existing engineering tools (RailCOMPLETE)

Embedding railML in CAD

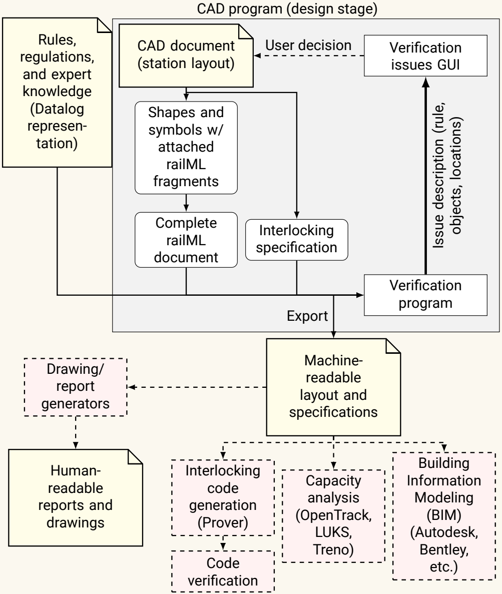
- CAD docs are object databases of geometrical objects
- railML is an XML based language for data exchange of railway designs, developed by an international standardization committee



CAD verification tool and tool chain

- Also, the structured data can be re-used for many other purposes, notably data exchange with other tools:
  - Interlocking code generation and verification
  - Capacity simulation
  - 3D view, Building Information Modeling
- This leads us to the tool chain overview...

Tool chain overview



- Dotted boxes indicate external programs
- Static verification can discover violations of technical regulations early, as the user is building the model
- More heavy-weight verification, simulation, testing, etc. benefits from machine-readable data exhcange

Talk outline

1. Background and motivation
2. Embedding railML in CAD
3. Verification of regulations using a Datalog language
4. Prototype tool integrating this verification into existing engineering tools (RailCOMPLETE)

Formalization of rule checking

- Formalize the following information
  - The CAD design (extensional information, or facts)
  - The regulations (intensional information, or rules)
- Use a solver which:
  - Is capable of verifying the rules
  - Runs fast enough for on-the-fly verification

Datalog

- Basic Datalog: conjunctive queries with fixed-point operators ("SQL with recursion")
  - Guaranteed termination
  - Polynomial running time (in the number of facts)
- Expressed as logic programs in a Prolog-like syntax:

$$a(X, Y) \text{ :- } b(X, Z), c(Z, Y)$$

$$\Updownarrow$$

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

- We also use:
  - Stratified negation (negation-as-failure semantics)
  - Arithmetic (which is "unsafe")

Encoding facts and rules in Datalog

- The process of formalizing the railway data and rules to Datalog format is divided into three stages:
  1. Railway designs (station data) – facts
  2. Derived concepts (used in several rules) – rules
  3. Technical regulations to be verified – rules
- Now, more details about each stage...

Input documents representation

- Translate the railML XML format into Datalog facts using the ID attribute as key:

$$
\begin{aligned}
\textit{track}(a) &\leftarrow \text{element}_a \text{ is of type } \texttt{track}, \\
\textit{signal}(a) &\leftarrow \text{element}_a \text{ is of type } \texttt{signal}, \\
&\vdots \\
\textit{pos}(a,p) &\leftarrow (\text{element}_a.\texttt{pos} = p), \quad a \in \text{Atoms}, p \in \mathbb{R}, \\
&\vdots \\
\textit{signalType}(a,t) &\leftarrow (\text{element}_a.\texttt{type} = t), \\
&\quad t \in \{\text{main, distant, shunting, combined}\}\,.
\end{aligned}
$$

Input documents representation

- To encode the hierarchical structure of the railML document, a separate predicate encoding the parent/child relationship is added:

$$\textit{belongsTo}(a, b) \leftarrow b \text{ is the closest XML ancestor of } a \text{ whose element type inherits from } \texttt{tElementWithIDAndName}.$$

Derived concepts

- Derived concepts are defined through intermediate rules
- Railway concepts defined independently of the design
- Example:

$$\mathit{directlyConnected}(a,b) \leftarrow \exists t : \mathit{track}(t) \wedge \mathit{belongsTo}(a,t) \wedge \mathit{belongsTo}(b,t),$$

$$\begin{aligned}\mathit{connected}(a,b) \leftarrow\ & \mathit{directlyConnected}(a,b) \vee (\exists c_1, c_2 : \mathit{connection}(c_1,c_2) \wedge \\ & \mathit{directlyConnected}(a,c_1) \wedge \mathit{connected}(c_2,b)).\end{aligned}$$

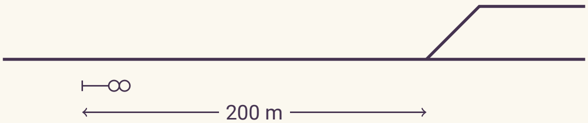
- A library of concepts allows concise expression of technical regulations

Technical regulations as Datalog rules

- Detecting errors in the design corresponds to finding objects involved in a regulation violation
- To *validate* the rules in a given design, we show that there are no satisfiable instances of the *negation* of the rule
- Some examples:
  - Example 1, home signal placement: topological and geometrical layout property for placement of a home signal
  - Example 2, train detector conditions: relates interlocking to topology
  - Example 3, flank protection conditions: relates interlocking to topology
- These are Jernbaneverket regulations which are relevant for automatic verification

Rule: example 1

- A *home main signal* shall be placed at least 200 m in front of the first controlled, facing switch in the entry train path.
- Uses arithmetic and negation

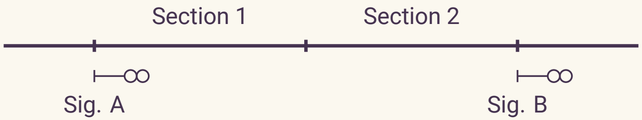


$$\textit{isFirstFacingSwitch}(b, s) \leftarrow \textit{stationBoundary}(b) \wedge \textit{facingSwitch}(s) \wedge \neg(\exists x : \textit{facingSwitch}(x) \wedge \textit{between}(b, x, s)),$$

$$\textit{ruleViolation}(b, s) \leftarrow \textit{isFirstFacingSwitch}(b, s) \wedge (\neg(\exists x : \textit{signalFunction}(x, \text{home}) \wedge \textit{between}(b, x, s)) \vee (\exists x, d, l : \textit{signalFunction}(x, \text{home}) \wedge \wedge \textit{distance}(x, s, d, l) \wedge l < 200).$$

Rule: example 2

- 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

Rule: example 2

$$\textit{adjacentDetectors}(a,b) \leftarrow \textit{trainDetector}(a) \wedge \textit{trainDetector}(b) \wedge \neg \textit{existsPathWithDetector}(a,b),$$

$$\textit{detectionSectionOverlapsRoute}(r,d_a,d_b) \leftarrow \textit{trainRoute}(r) \wedge \textit{start}(r,s_a) \wedge \textit{end}(r,s_b) \wedge \textit{adjacentDetectors}(d_a,d_b) \wedge \textit{overlap}(s_a,s_b,d_a,d_b),$$

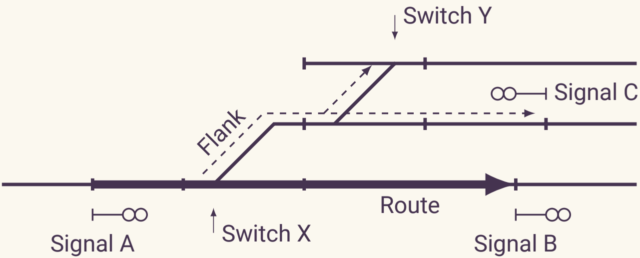
$$\textit{detectionSectionCondition}(r,d_a,d_b) \leftarrow \textit{detectionSectionCondition}(c) \wedge \textit{belongsTo}(c,r) \wedge \textit{belongsTo}(d_a,c) \wedge \textit{belongsTo}(d_b,c).$$

$$\textit{ruleViolation}(r,d_a,d_b) \leftarrow \textit{detectionSectionOverlapsRoute}(r,d_a,d_b) \wedge \neg \textit{detectionSectionCondition}(r,d_a,d_b).$$

- Rule needs negation

Rule: example 3

- For each switch in the route path and its associated 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.



- Declarative program helps conceptual clarity, good for maintenance and understanding
- Simple language encourages definition of auxiliary concepts

Talk outline

1. Background and motivation
2. Embedding railML in CAD
3. Verification of regulations using a Datalog language
4. Prototype tool integrating this verification into existing engineering tools (RailCOMPLETE)

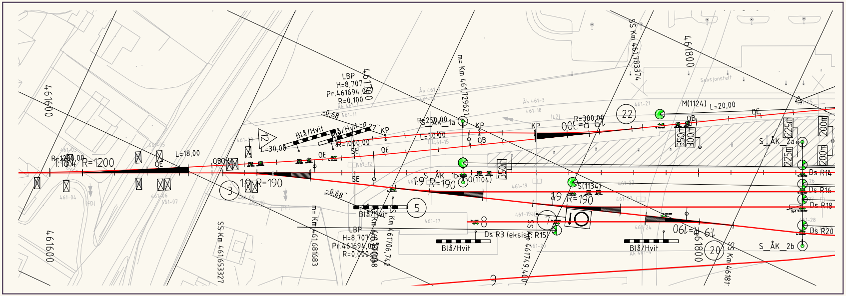
Prototype tool implementation

- Prototype tool implemented in XSB Prolog, which has tabled predicates
- Interfaces with the RailCOMPLETE tool which is based on Autodesk AutoCAD
- Rule base in Prolog syntax with structured comments giving information about rules
- Our example regulation (1) has the following code:

```
%| rule: Home signal too close to first facing switch.
%| type: technical
%| severity: error
homeSignalBeforeFacingSwitchError(S,SW) :-
    firstFacingSwitch(B,SW,DIR),
    homeSignalBetween(S,B,SW),
    distance(S,SW,DIR,L), L < 200.
```

Case study

- Railway engineers working on CAD model of Arna station, have thoroughly modeled using railML attributes
- After initial design phase, smaller changes are often made in response to changing requirements, etc. Fast and easy verification ensures consistency after such changes



Running time

	Testing station	Arna phase A	Arna phase B
Relevant components	15	152	231
Interlocking routes	2	23	42
Datalog facts	85	8283	9159
Running time (s)	0.1	4.4	9.4

- Running time for verification of a few properties: $\approx$1 – 10 s
  - More optimization needed for truly on-the-fly verification

Summary

- We have demonstrated a way to automate checking of regulations compliance for railway signalling and interlocking designs
- Our tools have been integrated in an existing CAD design environment
- Datalog allowed us to express technical regulations concisely and perform efficient verification
- Advantages:
  - eliminate tedious tasks, like filling out check-lists
  - get instant feedback on design quality while editing
  - make use of railML, a standard for describing railway designs

Future work

- Incremental updates (view maintenance)
  - Changes in the CAD design causes the whole verification to start over
  - More efficient: recompute only the parts that are affected by the changes
- Not much progress has happened since the DRed algorithm (Gupta et al. ’93), recent development (Boris Motik et al. ’15)
- RDFox tool (from Oxford) used in semantic web for OWL/SWRL has a recent implementation of updates