

# Automated Reasoning for Planning Railway Infrastructure

**Bjørnar Luteberget**

PhD Defence

18 Oct 2019



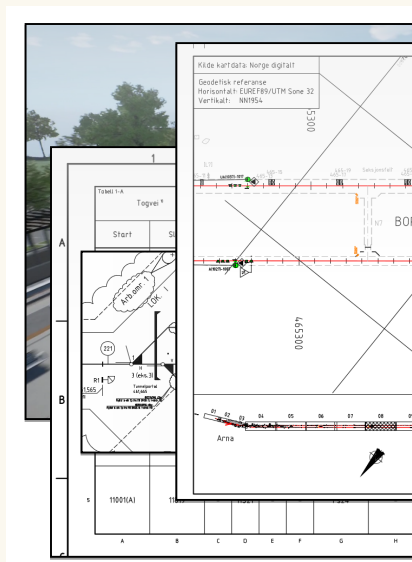
UiO : **University of Oslo**



**RailCOMPLETE**

# Background: railway engineering

- ▶ **Costly** projects with high quality requirements, **complicated** regulations.
- ▶ Produce a lot of tables, drawings, 3D models, specifications, documentation, etc.
- ▶ Evaluation relies on a lot of manual checking of regulations **compliance**.
- ▶ Coordination between **disciplines** require constant **re-evaluation** of designs.

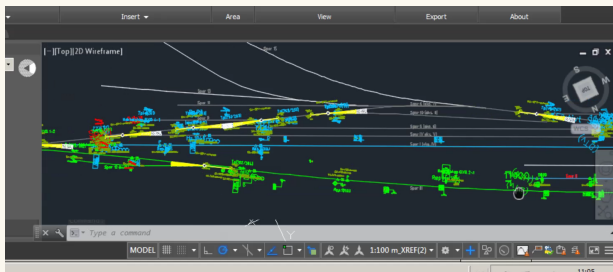


# Railcomplete, RailCons, IFI – project background

- ▶ Claus Feyling launched **Railcomplete AS**:  
*Bringing **BIM** to your railway projects.*
- ▶ Also launched **RailCons**, industry Ph.D. project funded by Norwegian Research Council and Railcomplete AS. In collaboration with **IFI** (Christian Johansen, Martin Steffen).

# RailCons goals

- ▶ Basis: the **RailCOMPLETE** editor for railway modeling
- ▶ RailCOMPLETE has special-purpose analysis features.



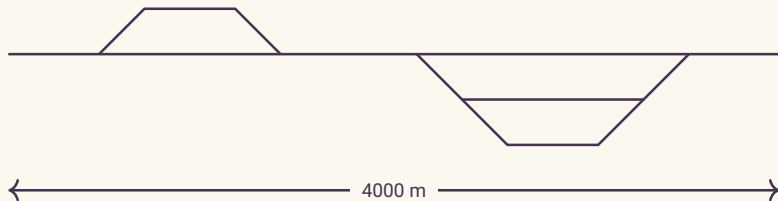
- ▶ RailCons: develop expressive analysis frameworks for **correctness** and **goodness**.
- ▶ The goal is to **verify** design properties, **optimize** and **synthesize** designs.
- ▶ Combine the strengths of IFI/PSY/Formal methods with Railcomplete's vision for construction projects.

# Presentation overview

1. **Local capacity verification** (SAT and simulation)
2. **Static** properties from regulations (Datalog)
3. **Controlled natural language** as a front-end for specifications (Grammatical Framework)
4. Drawing **schematic** views (SAT and numerical)

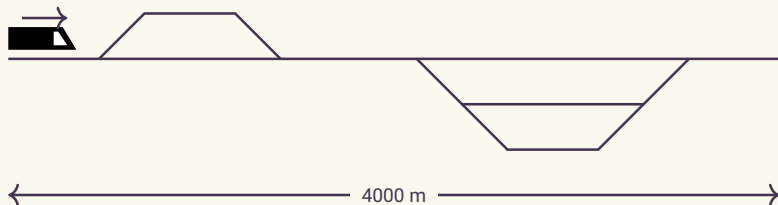
# Railway control systems

Constructing a new railway line starts with a **track plan**:



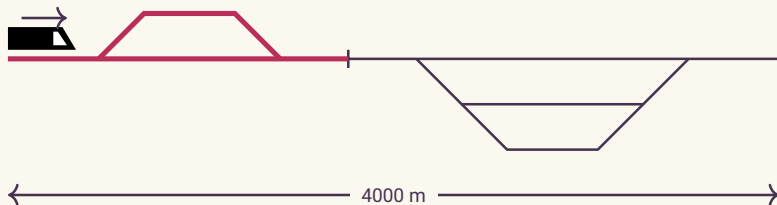
# Railway control systems

Constructing a new railway line starts with a **track plan**:



# Railway control systems

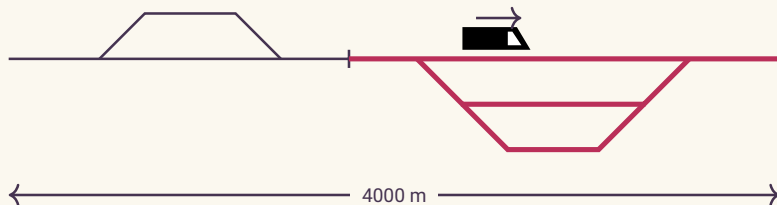
By adding **detectors**, we can allocate smaller pieces of tracks to the train:





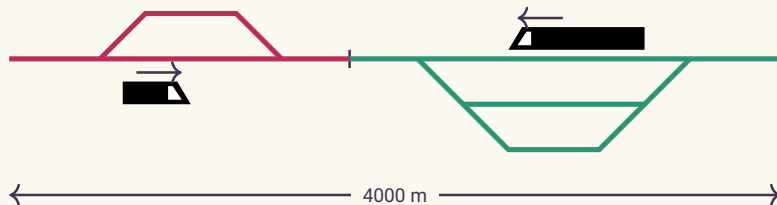
# Railway control systems

By adding **detectors**, we can allocate smaller pieces of tracks to the train:



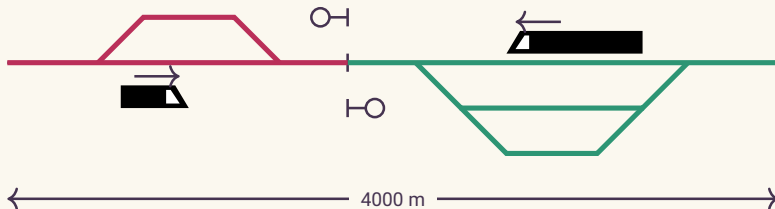
# Railway control systems

Now, **other trains** can occupy different sections.



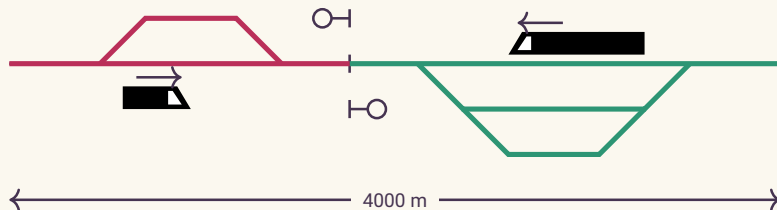
# Railway control systems

We add **signals** to indicate to drivers when they can proceed.



# Railway control systems

This situation is in principle **safe**, but is it a **good design**?



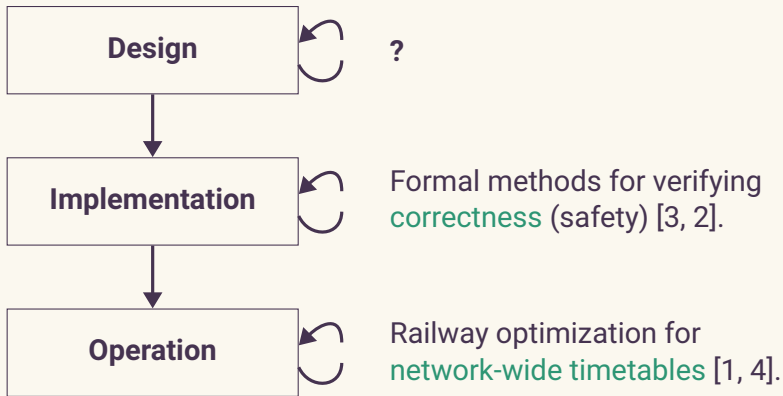
# Requirements

Will my **station design** handle the  
actual **traffic**?

Two methods used in practice:

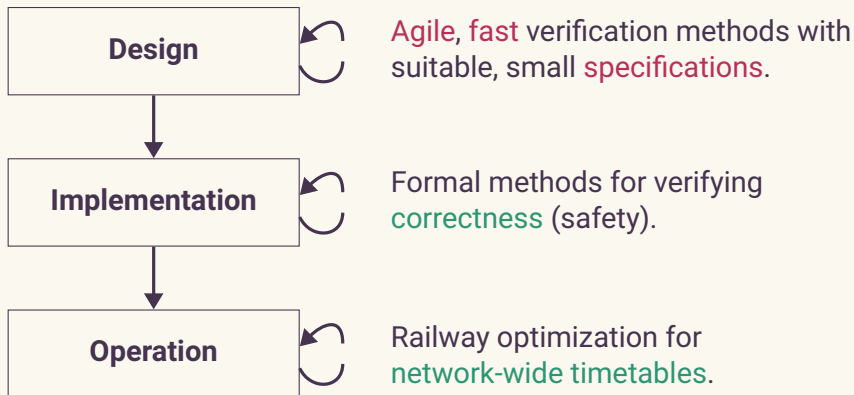
1. Whole-network **time table** analysis: a whole discipline in itself – complicated theory and software
2. Manual, **ad-hoc** analysis: varying quality, little documentation, low repeatability.

# Design-implementation-operation



- [1] M. Abril, F. Barber, L. Ingolotti, M.A. Salido, P. Tormos, and A. Lova. An assessment of railway capacity. *Transportation Research*, 44(5):774 – 806, 2008.
- [2] Arne Borälv and Gunnar Stålmärck. Formal verification in railways. In *Industrial-Strength Formal Methods in Practice*, pages 329–350. Springer, 1999.
- [3] A. Fantechi, W. Fokkink, and A. Morzenti. Some trends in formal methods applications to railway signalling. In *Formal Methods for Industrial Crit Sys.*, 2012.
- [4] Alex Landex. *Methods to est. railway cap. and passenger delays*. PhD thesis, 2008.

# Design-implementation-operation



# Specification capture

Railway engineers gave us examples of **performance properties** that governed their designs.

Typical categories:

1. Running time (get from A to B)
  - Similar to a simulation test, but smaller specification.
2. Frequency (several consecutive trains)
  - Route trains into alternate tracks.
3. Overtaking
4. Crossing
  - Let one train wait on a side track while another train passes.



# Capacity specifications

**Local** requirements suitable for construction projects.

- ▶ Operational scenario  $S = (V, M, C)$ :
- ▶ **Vehicle types**  $V = \{(l_i, v_i^{\max}, a_i, b_i)\}$ , defined by length, max velocity, max accel, max braking.
- ▶ **Movements**  $M = \{(v_i, \langle q_i \rangle)\}$ , defined by vehicle type  $v$  and ordered sequence of visits  $\langle q_i \rangle$ .
  - ▶ Each **visit**  $q_i = (\{l_i\}, t_d)$  is a set of alternative locations  $l_i$  and an optional dwelling time  $t_d$ .
- ▶ **Timing constraints**  $C = \{(q_a, q_b, t_c)\}$  which orders two visits and sets a maximum time from the first to the second  $t_{q_a} < t_{q_b} < t_{q_a} + t_c$ . The maximum time constraint can be omitted ( $t_c = \infty$ ).

# Constraints

**Verification** of these **specifications** would involve finding satisfying **train trajectories** and **control system state**:

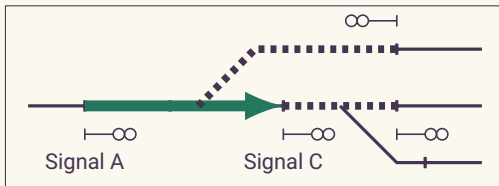
$$\exists p : \text{spec}(p)$$

Also, constrained by:

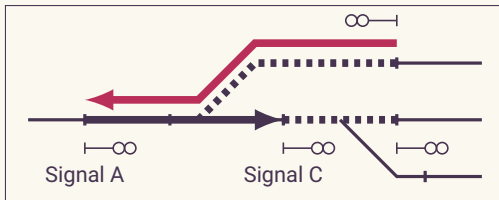
- ▶ 1 - Physical infrastructure
- ▶ 2 - Allocation of resources (collision safety)
- ▶ 3 - Limited communication
- ▶ 4 - Laws of motion

## Constraints (2) Allocation of resources

An **elementary route** is a set of resources allocated together.

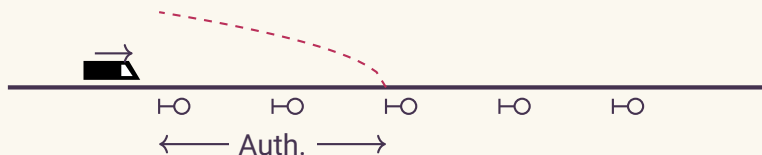
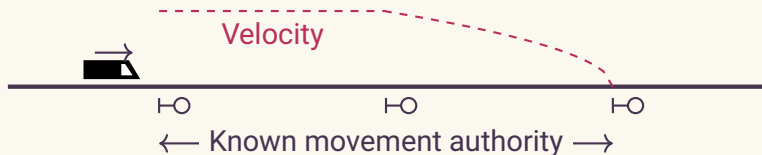


Routes are **conflicting** if they use any of the same resources.



## Constraints (3) Limited communication

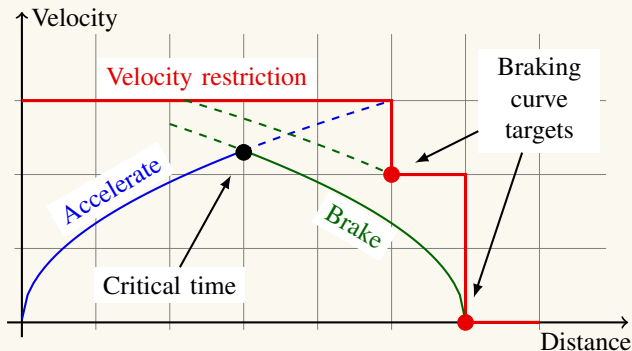
Signal information only carries across two signals ("pre-signalling").



## Constraints (4) Laws of motion

Trains move within the limits of given maximum acceleration and braking power. Train drivers need to plan ahead for braking so that the train respects its given movement authority and speed restrictions at all times.

$$v - v_0 \leq a\Delta t, \quad v^2 - v_i^2 \leq 2bs_i.$$



# Automated verification

Design-time capacity verification amounts to **planning** in a **mixed discrete/continuous** space.

Some suggestions:

- ▶ **PDDL+**, planning domain description language for mixed discrete-continuous planning domains [1].
- ▶ **SMT** with **non-linear real** arithmetic [2, 4].
- ▶ **dReal**:  $\delta$ -complete decision proc. for FOL with reals [3].

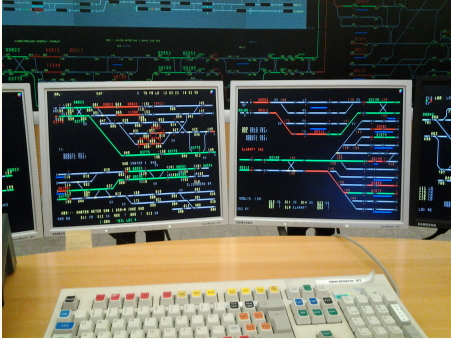
Using these tools/techniques and straight-forward modeling did **not** make our problem manageable on relevant scales.

- [1] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.
- [2] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *J. SAT*, 1:209–236, 2007.
- [3] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. CADE-24 vol. 7898 of *LNCS*, pages 208–214. Springer, 2013.
- [4] D. Jovanovic and L. de Moura. Solving non-linear arithmetic. *ACM Comm. Computer Algebra*, 46(3/4):104–105, 2012.

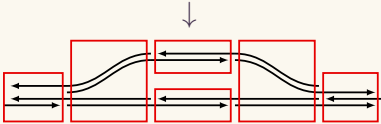
# Dispatch vs. driver

Split the planning work into two separate points of view:

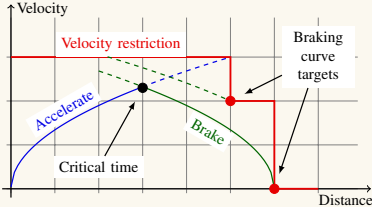
### Dispatcher



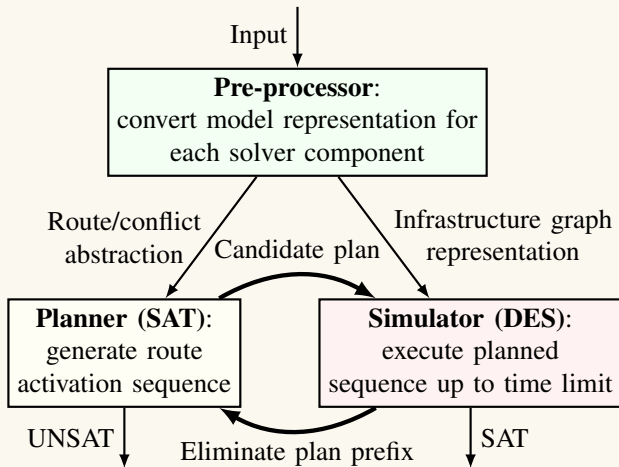
### Train driver



Elementary routes and their conflicts



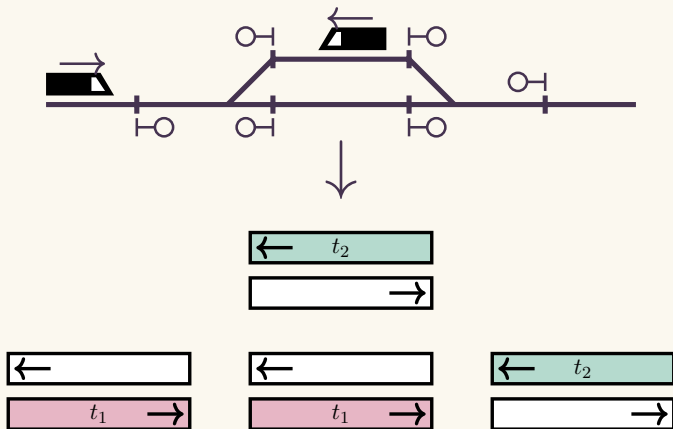
# Local Capacity Solver architecture





# SAT encoding of dispatch planning

General idea: represent **which train** occupies **which elementary route** in each of a sequence of **steps**.



# SAT encoding

Planning as **bounded model checking** (BMC [1,2]). Build planning steps as needed using **incremental** SAT solver interface.

Movement correctness:

- ▶ **Conflicting** routes are not active simultaneously  
 $\text{conflict}(r_1, r_2) \Rightarrow o_{r_1}^i = \text{Free} \vee o_{r_2}^i = \text{Free}.$
- ▶ Elementary route allocation is **consistent** with train movement:  
 $(o_r^i \neq t \wedge o_t^{i+1} = t) \Rightarrow$   
 $\bigvee \{ o_{r_x}^{i+1} = t \mid \text{route}(r_x), \text{entry}(r) = \text{exit}(r_x) \}$

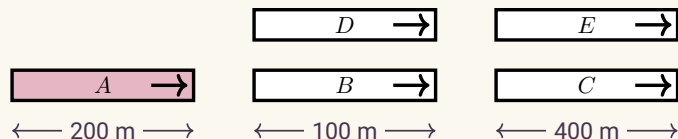
Satisfy specification:

- ▶ Visits happen in order (timing requirement is measured on simulation).

[1] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19:7–34, 2001.

[2] J. F. Groote, S. F. M. van Vlijmen, and J. W. C. Koorn. The safety guaranteeing system at station Hoorn-Kersenboogerd. *COMPASS '95*, p. 57–68. IEEE, 1995.

# Freeing



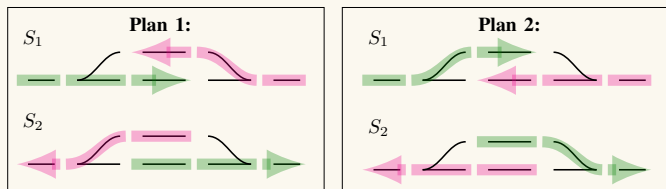
If  $A$  holds a train  $t$  of length 200.0 m, freeing  $A$  is constrained by:

$$A^i \Rightarrow (A^{i+1} \vee (B^i \wedge C^i) \vee (D^i \wedge E^i)).$$

## Eliminate equivalent solutions

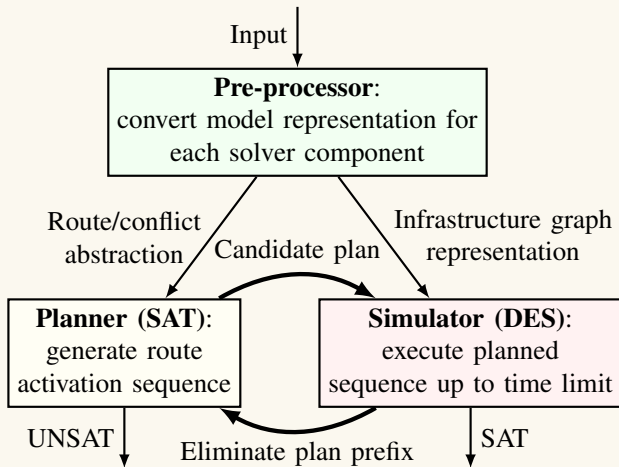
- ▶ Can free  $\Rightarrow$  must free
- ▶ Can allocate  $\Rightarrow$  must allocate
- ▶ Exception to allocation: **deferred progress**  
a train may be waiting for a conflict to be resolved, even if the conflict starts in the future.

Crossing example: **exactly two** solutions:

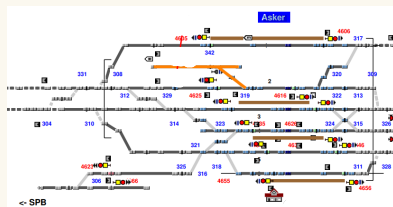


- ▶ Overlaps. Partial release.
- ▶ Loops in the infrastructure / loops in the dispatch.

# Local Capacity Solver architecture



# Case studies



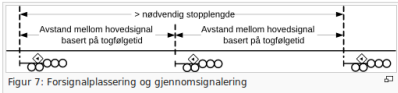
| Infrastructure              | Property     | Result | $n_{DES}$ | $t_{SAT}$ | $t_{DES}$ | $t_{total}$ |
|-----------------------------|--------------|--------|-----------|-----------|-----------|-------------|
| Simple<br>(3 elem.)         | Run.time     | Sat.   | 1         | 0.00      | 0.00      | 0.00        |
|                             | Crossing     | Unsat. | 0         | 0.00      | 0.00      | 0.00        |
| Two track<br>(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        |
| Kolbotn (BN)<br>(56 elem.)  | Crossing 3   | Unsat. | 0         | 0.01      | 0.00      | 0.01        |
|                             | Run.time     | Sat.   | 2         | 0.01      | 0.00      | 0.02        |
|                             | Overtake 4   | Sat.   | 1         | 0.05      | 0.00      | 0.06        |
| Eidsvoll (BN)<br>(64 elem.) | Overtake 3   | Unsat. | 0         | 0.05      | 0.00      | 0.06        |
|                             | 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        |
| Asker (BN)<br>(170 elem.)   | Crossing 4   | Unsat. | 0         | 0.21      | 0.00      | 0.21        |
|                             | Overtaking 2 | Sat.   | 1         | 0.20      | 0.00      | 0.21        |
|                             | Overtaking 3 | Unsat. | 1         | 0.73      | 0.00      | 0.74        |
| Arna (CAD)<br>(258 elem.)   | Crossing 4   | Sat.   | 0         | 0.75      | 0.00      | 0.77        |
|                             | 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        |
| Gen. 3x3<br>(74 elem.)      | Crossing 4   | Sat.   | 1         | 1.73      | 0.00      | 1.74        |
|                             | High time    | Sat.   | 1         | 0.01      | 0.00      | 0.01        |
| Gen. 4x4<br>(196 elem.)     | Low time     | Unsat. | 27        | 0.18      | 0.01      | 0.19        |
|                             | High time    | Sat.   | 1         | 0.01      | 0.00      | 0.03        |
| Gen. 5x5<br>(437 elem.)     | Low time     | Unsat. | 256       | 2.08      | 0.26      | 2.34        |
|                             | 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.*) is shown for each infrastructure to indicate 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.

# Static properties: technical regulations

- ▶ In our case study: Norwegian regulations from national railways (Bane NOR)
- ▶ **Static** kind of properties, often related to object properties, topology and geometry (example on next slide)

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 [↗](#).



f) Et forsignal skal plasseres på foregående hovedsignals mast dersom avstanden mellom det tilhørende hovedsignalet og det foregående hovedsignalet er  $\leq 2200$  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 [↗](#):

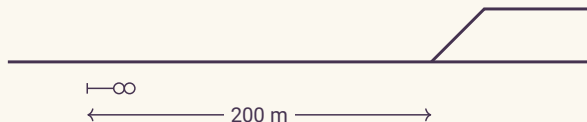
**Tabell 4: Sikt krav til forsignal**

| Sikt       | Strekningens høyeste tillatte kjørehastighet [km/h] |    |    |     |     |     |     |     |     |     |     |     |     | ≥130 |     |     |     |     |     |
|------------|---|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|
|            | 40  | 45 | 50 | 55  | 60  | 65  | 70  | 75  | 80  | 85  | 90  | 95  | 100 |      | 105 | 110 | 115 | 120 | 125 |
| Brutt sikt | 70  | 80 | 87 | 107 | 117 | 126 | 126 | 146 | 156 | 165 | 175 | 185 | 194 | 204  | 214 | 224 | 233 | 243 | 250 |

# Static properties: 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.



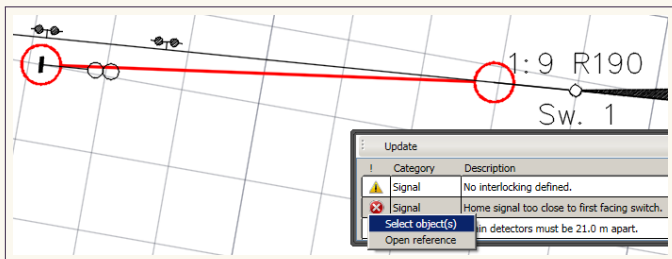
- ▶ Can be classified as follows:
  - Object properties
  - Topological layout properties
  - Geometrical layout properties
  - Interlocking properties



# Datalog verification tool

- ▶ Prototype using **XSB Prolog** tabled predicates, front-end is the **RailCOMPLETE** tool based on Autodesk AutoCAD
- ▶ Rule base in Prolog syntax with **structured comments** giving information about rules

```
%| 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 .
```



## Challenge: participatory verification

Challenge: Users (railway engineers) are not experts in verification techniques, so how can they

- ▶ **build** models of the systems to be verified?
- ▶ **write** properties in the verifier's input language?
- ▶ **interpret** the output of the verifier when violated properties are found?

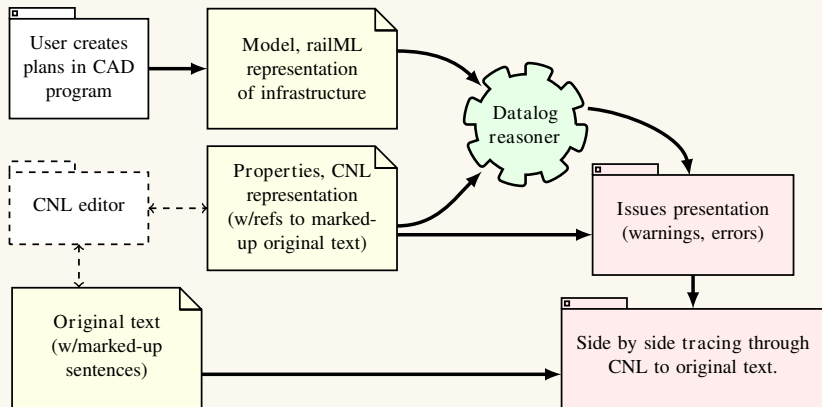
Input to verification:

- ▶ **Models**: CAD extended with structured railway data (familiar to engineers, user-friendly)
- ▶ **Properties**: **Datalog** (unfamiliar to engineers, not user-friendly enough)

... consider another **verification property input language**?

# Overview of approach

- ▶ Define a **Controlled Natural Language** as a high-level **domain-specific** language to write properties.
- ▶ Represent properties as rephrasing of natural language specifications (adds tracability of requirements)



# Issues view

## ► Backwards tracing – explanation of non-compliance

CAD program showing issues in layout plan



CNL debug view paraphrased text and translations



Original text highlighting source of paraphrased text

|   | Category | Description   |
|---|----------|---|
| ⚠ | Signal   | No interlocking defined.                                      |
| ✖ | Signal   | The distance from a train detector to another must be greater |

### **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, End

### **Placement and length**

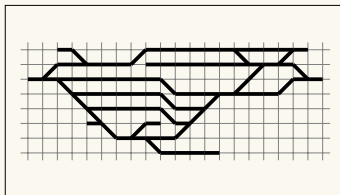
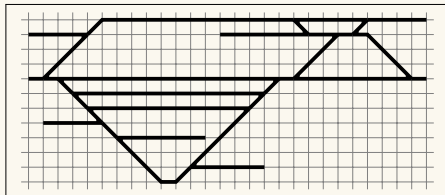
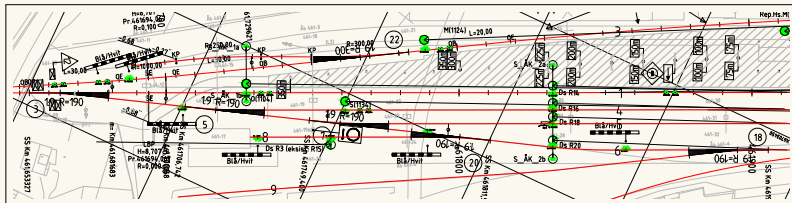
This section gives generalized rules for placement and length for train detection systems and its relationship to other infrastructure components. Detailed requirements are given in appendices.

#### **General**

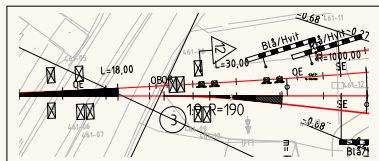
- No detection sections shall be shorter than 21 meters.**
- No dead zone shall be longer than 3 meters.

# Schematic drawings

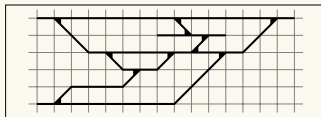
- ▶ Incremental SAT with numerical constraints: unary encoding vs. SMT difference constraints.
- ▶ Choose criteria bends vs. size.



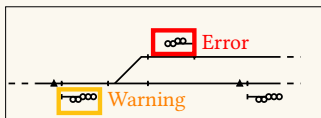
# RailCons results overview (1/2)



**Infrastructure models**, edited in a graphical interactive editor (CAD program) extended with railway semantic data and translated into railML for analysis (Ch. 7).



**Schematic drawings**, automatically created from the topological data in an infrastructure model using a linear track referencing system (Ch. 6).



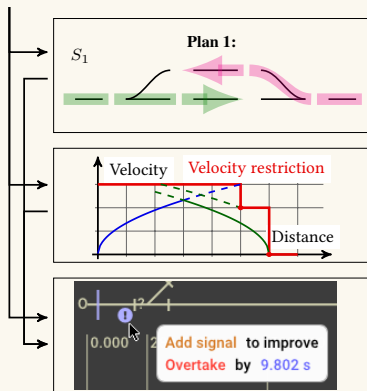
**Static verification**, analysis of infrastructure and interlocking models according to specifications given as Datalog logic programs (Ch. 2).

## **ID: detector 1**

**RailCNL:** The distance from an axle counter to a  
**AST:** DistanceRestriction Obligation (SubjectCla  
"axle\_counter")) (AnyFound (AnyDirectionObjec  
**Datalog:** detector\_1\_start(Subj0, End, Dist) :- tr

**Controlled natural language**, specifying properties of infrastructure using a natural language-like syntax, with editor support (Ch. 5).

## RailCons results overview (2/2)



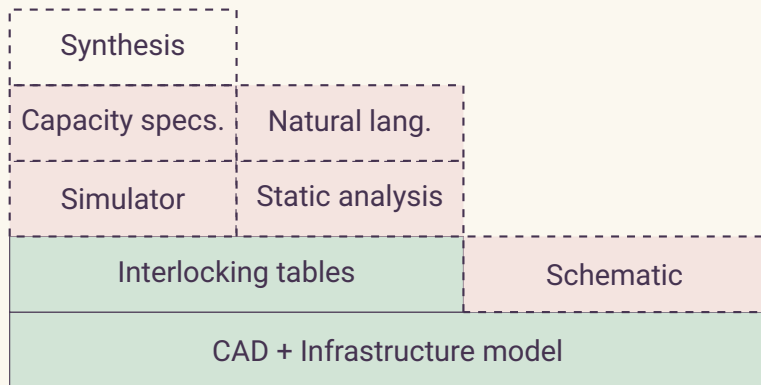
**Planning of operations**, using SAT for capacity verification, with special-purpose specifications suited to construction projects (Ch. 3).

**Simulation**, implemented by established methods and used as a timing measurement component in capacity verification (Ch. 3).

**Synthesis and optimization**, creating a signalling design from scratch or suggesting improvements to existing designs. (Ch. 4).

# Into the future

- ▶ A main goal was to provide engineers with tools.
- ▶ Many remaining challenges in **representation**, **interfaces**, **domain complexity(!)**. **Railcomplete AS** is progressing.
- ▶ Engineer+developer **collaboration** is essential.

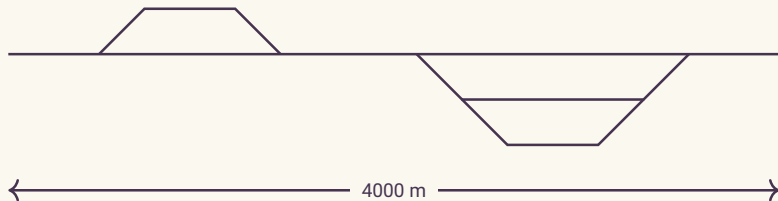






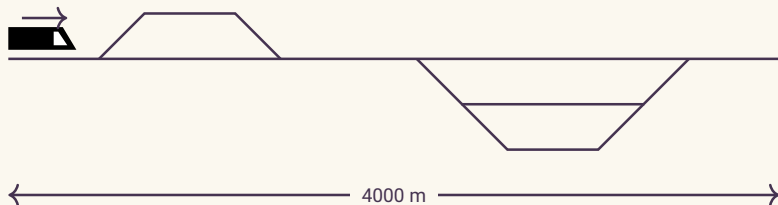
# Railway control systems

Constructing a new railway line starts with a **track plan**:



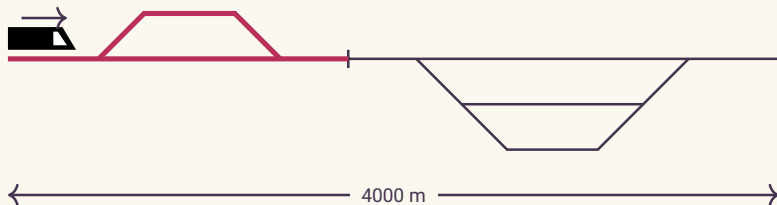
# Railway control systems

Constructing a new railway line starts with a **track plan**:



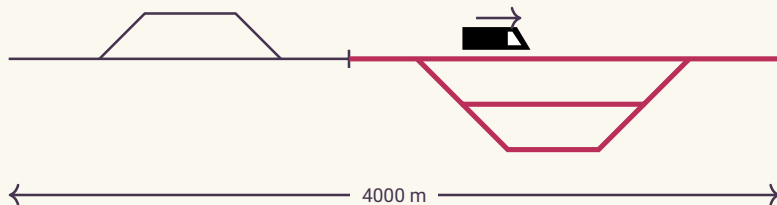
# Railway control systems

By adding **detectors**, we can allocate smaller pieces of tracks to the train:



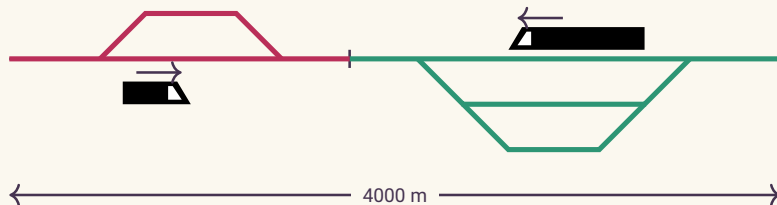
# Railway control systems

By adding **detectors**, we can allocate smaller pieces of tracks to the train:



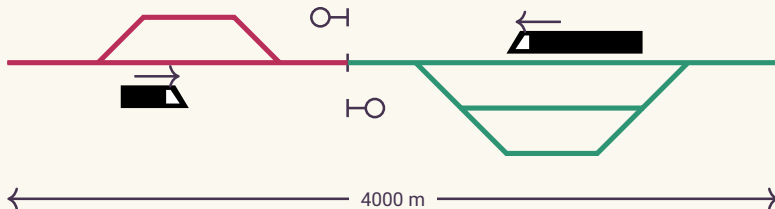
# Railway control systems

Now, **other trains** can occupy different sections.



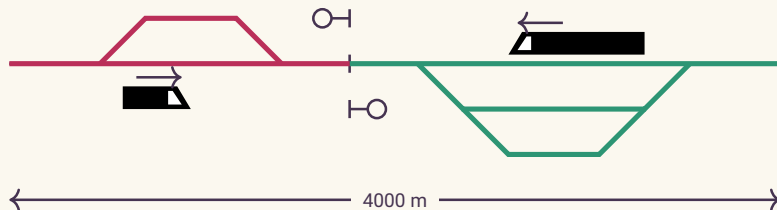
# Railway control systems

We add **signals** to indicate to drivers when they can proceed.



# Railway control systems

This situation is in principle **safe**, but is it a **good design**?

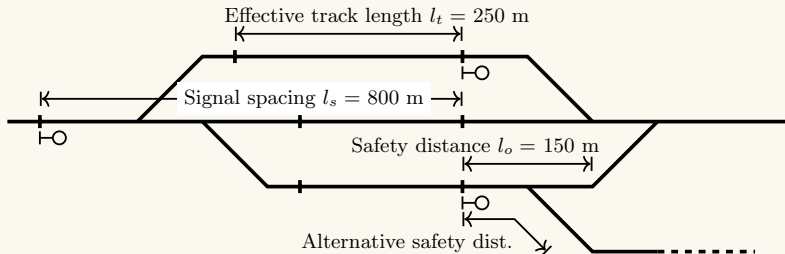




# Two views on capacity: schematic track plan

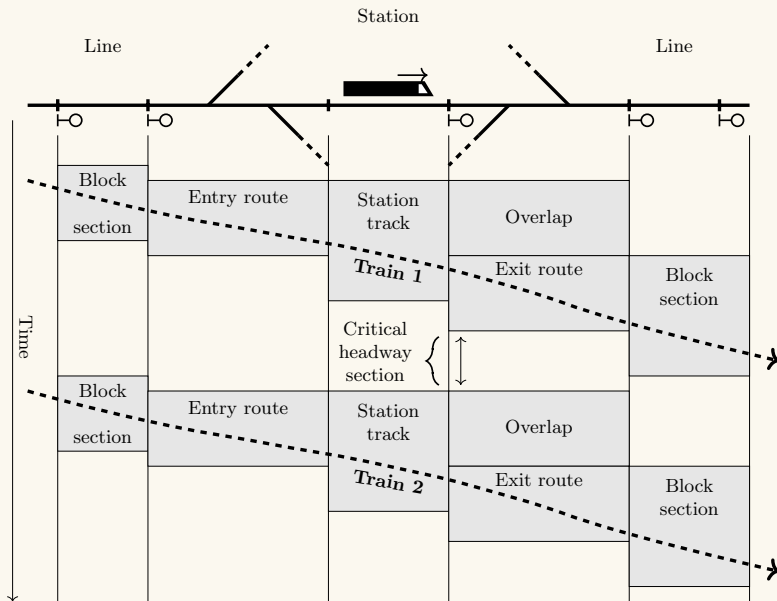
The **schematic track plan** is a map of **tracks and components**, such as signals, detectors, etc.

Distance margins determine allowable **simultaneous** movements.



# Two views on capacity: blocking diagram

A **single path**, or related paths mapped to a linear axis.



# Specification capture

Railway engineers gave us examples of **performance properties** that governed their designs.

Typical categories:

1. Running time (get from A to B)
  - Similar to a simulation test, but smaller specification.
2. Frequency (several consecutive trains)
  - Route trains into alternate tracks.
3. Overtaking
4. Crossing
  - Let one train wait on a side track while another train passes.

# Capacity specifications

**Local** requirements suitable for construction projects.

- ▶ Operational scenario  $S = (V, M, C)$ :
- ▶ **Vehicle types**  $V = \{(l_i, v_i^{\max}, a_i, b_i)\}$ , defined by length, max velocity, max accel, max braking.
- ▶ **Movements**  $M = \{(v_i, \langle q_i \rangle)\}$ , defined by vehicle type  $v$  and ordered sequence of visits  $\langle q_i \rangle$ .
  - ▶ Each **visit**  $q_i = (\{l_i\}, t_d)$  is a set of alternative locations  $l_i$  and an optional dwelling time  $t_d$ .
- ▶ **Timing constraints**  $C = \{(q_a, q_b, t_c)\}$  which orders two visits and sets a maximum time from the first to the second  $t_{q_a} < t_{q_b} < t_{q_a} + t_c$ . The maximum time constraint can be omitted ( $t_c = \infty$ ).

# Advantages of capacity specification

Can be specified for a single **construction project**, not dependent on whole-network timetables.

This can give us:

- ▶ Improved **communication** about specifications between contractual parties.
- ▶ Automated analysis
  - Early-stage, lower-effort capacity verification
  - Regression testing after changes in design
  - Unifies ad-hoc methods in use today
- ▶ Better understanding and communication between construction engineers and timetable planners.

# Verification of local capacity specifications

**Verification** of these **specifications** would involve finding satisfying **train trajectories** and **control system state**:

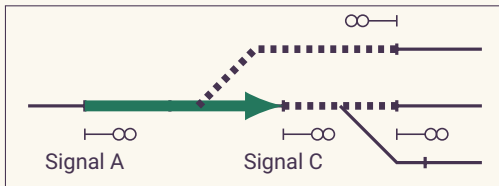
$$\exists p : \text{spec}(p)$$

Also, constrained by:

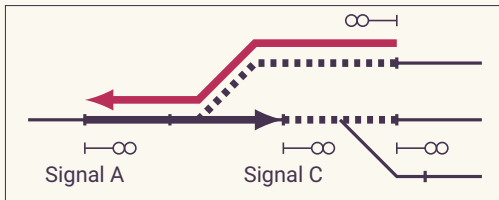
- ▶ 1 - Physical infrastructure
- ▶ 2 - Allocation of resources (collision safety)
- ▶ 3 - Limited communication
- ▶ 4 - Laws of motion

## Constraints (2) Allocation of resources

An **elementary route** is a set of resources allocated together.

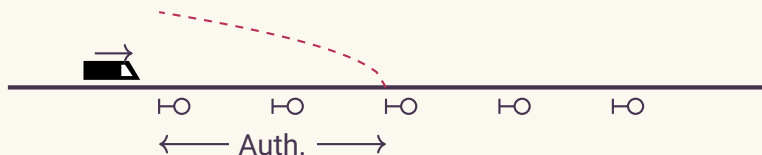
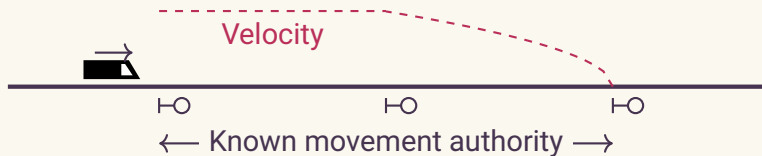


Routes are **conflicting** if they use any of the same resources.



## Constraints (3) Limited communication

Signal information only carries across two signals ("pre-signalling").

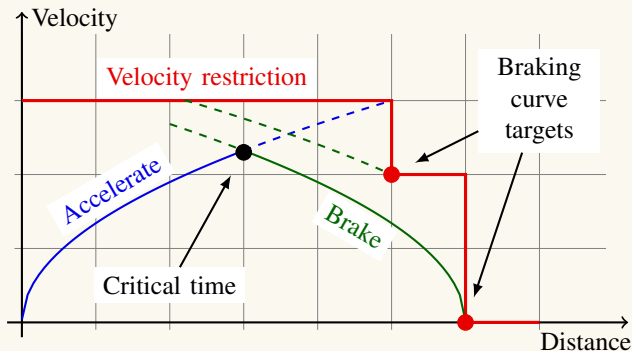




## Constraints (4) Laws of motion

Trains move within the limits of given maximum acceleration and braking power. Train drivers need to plan ahead for braking so that the train respects its given movement authority and speed restrictions at all times.

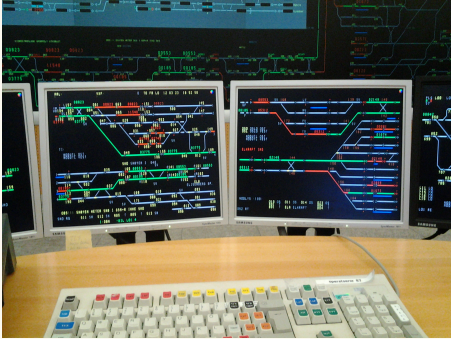
$$v - v_0 \leq a\Delta t, \quad v^2 - v_i^2 \leq 2bs_i.$$



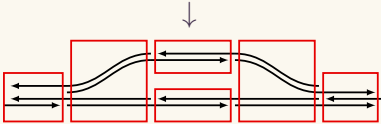
# Dispatch vs. driver

Split the planning work into two separate points of view:

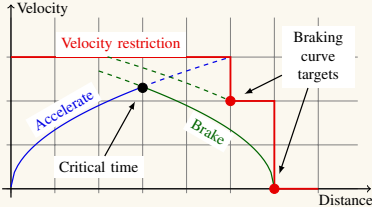
### Dispatcher



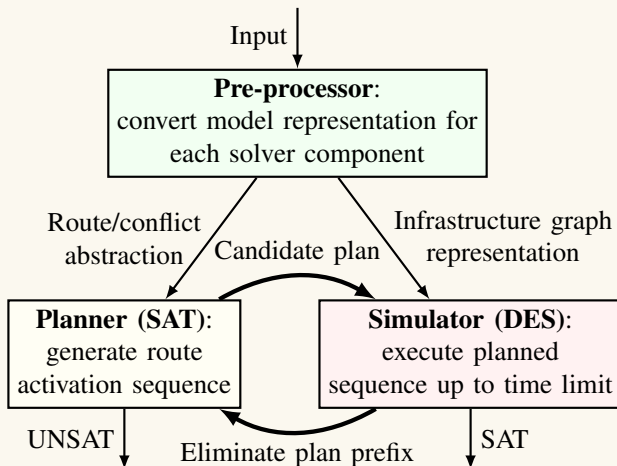
### Train driver



Elementary routes and their conflicts

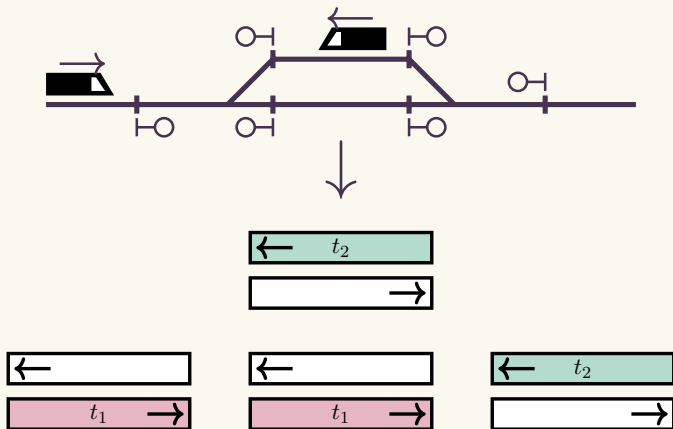


# Verification architecture



# SAT encoding of dispatch planning

General idea: represent **which train** occupies **which elementary route** in each of a sequence of **steps**.



# SAT encoding

Planning as **bounded model checking** (BMC). Build planning steps as needed using **incremental** SAT solver interface.

Movement correctness:

- ▶ **Conflicting** routes are not active simultaneously  
 $\text{conflict}(r_1, r_2) \Rightarrow o_{r_1}^i = \text{Free} \vee o_{r_2}^i = \text{Free}.$
- ▶ Elementary route allocation is **consistent** with train movement:  $(o_r^i \neq t \wedge o_t^{i+1} = t) \Rightarrow \bigvee \{o_{r_x}^{i+1} = t \mid \text{route}(r_x), \text{entry}(r) = \text{exit}(r_x)\}$

Satisfy specification:

- ▶ Visits happen in order (timing requirement is measured on simulation).

## From verification to synthesis

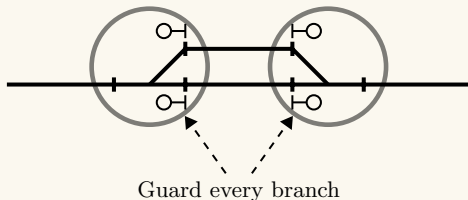
Can we use verification techniques  
to **synthesize** signaling designs?

# Initial design

- ▶ Adding a single component somewhere does **not** give any good information.
- ▶ Let's turn **synthesis into optimization** by **over-approximating** required components.

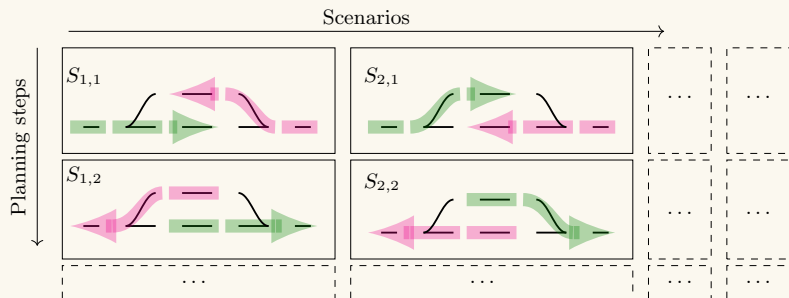
Start with an initial design:

- ▶ Include signals at **fixed distances** from **merging** paths.
- ▶ The distances correspond to choices of **overlap distance**.



# Minimize number of signals

- ▶ Instead of verifying each property **separately**, on a **known model** ...
- ▶ ... we have **unknowns** in the model, and need to satisfy **all properties** simultaneously.





## Minimize number of signals

- ▶ Then, we can add a **signal used indicator** boolean to the SAT problem, linking the usage of a signal across all planning steps and all scenarios.

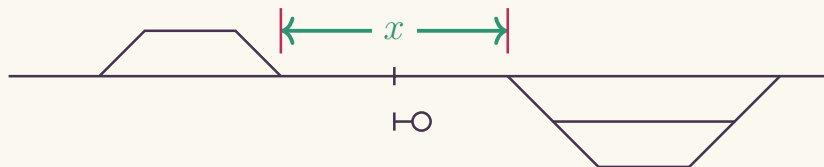
$$\forall i \in \text{State} : \forall s \in \text{Signal} : \forall t \in \text{Train} : \neg u_s \Rightarrow$$
$$\bigvee \{ (o_r^i \neq t \wedge o_r^{i+1} = t) \mid \text{exit}(r) = s \} \Rightarrow$$
$$\bigvee \{ (o_r^i \neq t \wedge o_r^{i+1} = t) \mid \text{entry}(r) = s \} .$$

- ▶ Solve MaxSAT maximising unused signals.

# Numerical optimization of component locations

Signal minimization gives a set of **signals** and a set of corresponding **dispatches** which fulfil the given specifications.

- ▶ **Adjusting positions** of components may improve timing results in simulator.
- ▶ **Discontinuous**, non-linear, multivariate real-valued optimization problem.



# The function to be optimized

The function to be optimized is a **weighted sum** of dispatch **timing** measures.

$$f_b(\vec{x}) = \sum_s w_s \left( \frac{1}{n_s} \sum_d t_{b+\vec{x}}(d) \right),$$

where

- ▶  $\vec{x}$  represents the location of each signal and detector,
- ▶  $s$  indexes capacity specifications,
- ▶  $w_s$  is the weight assigned to specification  $s$ ,
- ▶  $d$  indexes dispatch plans for each operational scenario, and
- ▶  $t_{b+\vec{x}}(d)$  is the simulation timing result.

(Trading **performance** and **cost** is performed by the user)

# Powell's method

We fix the set of components, fix the tracks that they belong to, and fix their order within the track.

## Powell's method (1964):

- ▶ Given domain  $D \subset \mathbb{R}^n$ , initial point  $\vec{x}_0 \in D$ , and cost function  $f : D \rightarrow \mathbb{R}$ .
- ▶ Iterate through search vectors  $\vec{v}_i \in V$  and do a line search for  $\alpha \in \mathbb{R}$  minimizing  $\vec{x}_{i+1} = f(\vec{x}_i + \alpha\vec{v}_i)$ .
- ▶ Remove the  $\vec{v}_i$  which yielded the highest  $|\alpha|$ , and replace it with  $\vec{x}_{i+1} - \vec{x}_i$  normalized. Repeat until  $\|\vec{x}_{i+1} - \vec{x}_i\| < \epsilon$ .

## Brent's method (1973):

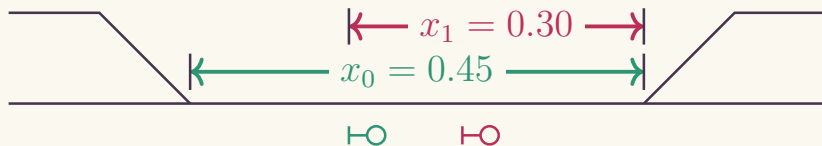
- ▶ A reliable method for root-finding or minimization for non-differentiable functions.
- ▶ For **well-behaved** functions: inverse quadratic interpolation, or linear interpolation.
- ▶ For **not-so-well-behaved** functions: bisection / golden section.

## Mapping locations to the unit cube

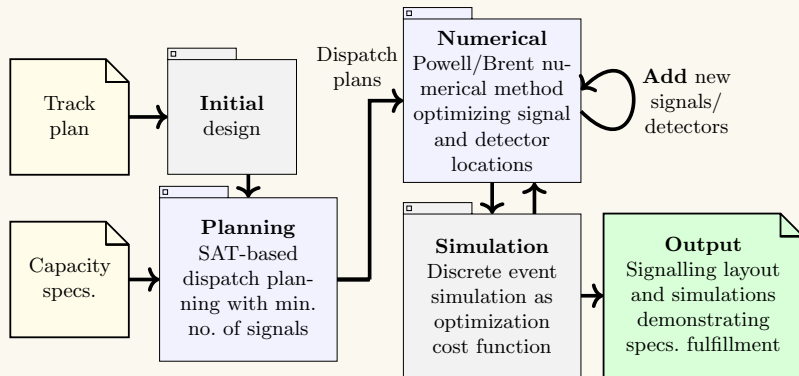
- ▶ Preserve which **tracks** components are located at, and their **order** to ensure planned dispatches are still meaningful. Minimum distance  $d$  between components.
- ▶ Map the component location space to the **unit cube**  $[0, 1]^n$  ( $n$ -tuples in  $[0, 1]$ ) so that the whole of the unit cube is a valid point in the component location space.

**Encode:** `scan(0.0, λ s, x → linstep(replace(s, x) + d, l - d, x)).`

**Decode:** `scan(0.0, λ s, x → replace(s, lerp(s + d, l - d, x))).`



# Synthesis algorithm overview



## Local optimization steps

- ▶ Synthesis **from scratch** not always suitable.
  - ▶ Instead, search for a single step of the synthesis algorithm that gives the most effect on the current design.
1. **Redundant component**: removing a single object while still satisfying specifications.
  2. **Local move of component**: moving a single object or a set of nearby objects may improve the overall capacity measure.
  3. **Adding component**: adding a single component (and performing local moves) which improves overall capacity measure.

Each of these can be **suggested** to the user.

## Related work

- ▶ Formal methods is all about **safe implementations** of control systems.
- ▶ Operations research is all about **time tabling** on large-scale networks.
- ▶ Mao, B. et al.: *Signalling layout for fixed-block railway lines with real-coded genetic algorithms*, Hong Kong Institute of Engineers, Transactions (2006).
- ▶ Weits, E. et al.: *Generating optimal signal positions*, Computers in Railways XII (2010).
  - Does not deal with schedulability.
  - Analytical performance models.
- ▶ Dillmann, S. and Hähnle, R.: *Automated planning of ETCS tracks*, RSSRAIL 2019.
  - Heuristic algorithm.



## Conclusions and future work

- ▶ Not a **complete** method:
  1. initial design does may not have maximum schedulability
  2. simultaneous planning may not be the best starting points.
  3. the cost function may have multiple local optima.
- ▶ **Scalability** concerns:
  1. specification language unsuited for large terminals.
  2. algorithm for adding new signals is naive.
- ▶ Assumes **fixed block** design principles. **ERTMS Level 3** with moving block may require different planning algorithm.
- ▶ Imperative simulation at the core allows **extending** timing calculations to be more sophisticated.
- ▶ Fast results for small infrastructures.

# RailCons project: automated verification

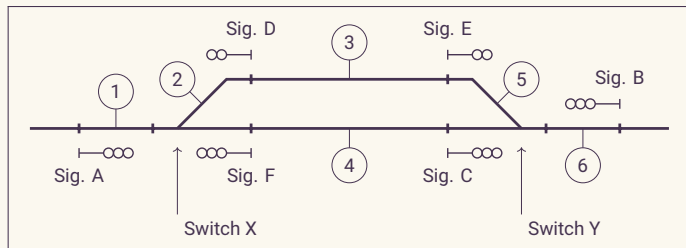
## Project objectives:

- ▶ Verify that railway signalling and interlocking **designs** **comply with regulations**.
- ▶ Provide tools which allow **railway engineers** to perform such verification as part of their **daily** routine (“lightweight verification”).

“Formal methods will never have a significant impact until they can be used by people that **don't understand** them.”

— (attributed to) Tom Melham

# Models: railway signalling and interlocking designs



(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

Static verification

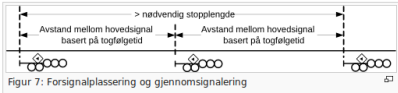
**Static** verification

Controlled natural language

# Properties: technical regulations

- ▶ In our case study: Norwegian regulations from national railways (Bane NOR)
- ▶ **Static** kind of properties, often related to object properties, topology and geometry (example on next slide)

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 [↗](#).



Figur 7: Forsignalsplassering og gjennomsignalering

f) Et forsignal skal plasseres på foregående hovedsignals mast dersom avstanden mellom det tilhørende hovedsignalet og det foregående hovedsignalet er  $\leq 2200$  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 [↗](#):

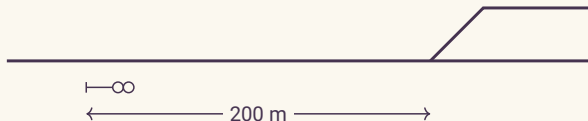
**Tabell 4: Sikt krav til forsignal**

| Sikt | Strekningens høyeste tillatte kjørehastighet [km/h] |    |    |     |     |     |     |     |     |     |     |     |     | ≥130 |     |     |     |     |     |
|------|---|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|
|      | 40  | 45 | 50 | 55  | 60  | 65  | 70  | 75  | 80  | 85  | 90  | 95  | 100 |      | 105 | 110 | 115 | 120 | 125 |
|      | 70  | 80 | 87 | 107 | 117 | 126 | 126 | 146 | 156 | 165 | 175 | 185 | 194 | 204  | 214 | 224 | 233 | 243 | 250 |

# Properties: 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.

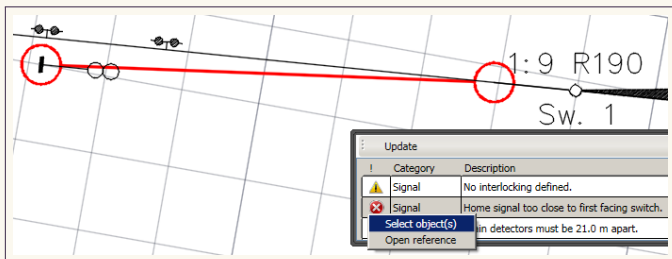


- ▶ Can be classified as follows:
  - Object properties
  - Topological layout properties
  - Geometrical layout properties
  - Interlocking properties

# Datalog verification tool

- ▶ Prototype using **XSB Prolog** tabled predicates, front-end is the **RailCOMPLETE** tool based on Autodesk AutoCAD
- ▶ Rule base in Prolog syntax with **structured comments** giving information about rules

```
%| 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 .
```



## Challenge: participatory verification

Challenge: Users (railway engineers) are not experts in verification techniques, so how can they

- ▶ **build** models of the systems to be verified?
- ▶ **write** properties in the verifier's input language?
- ▶ **interpret** the output of the verifier when violated properties are found?

Input to verification:

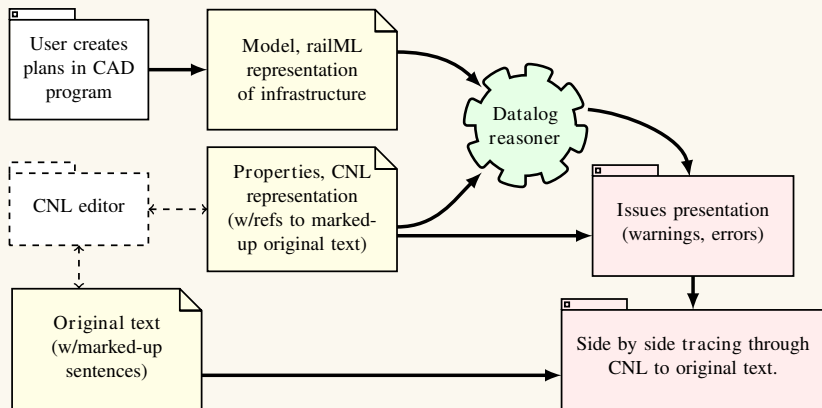
- ▶ **Models**: CAD extended with structured railway data (familiar to engineers, user-friendly)
- ▶ **Properties**: **Datalog** (unfamiliar to engineers, not user-friendly enough)

... consider another **verification property input language**?



# Overview of approach

- ▶ Define a **Controlled Natural Language** as a high-level **domain-specific** language to write properties.
- ▶ Represent properties as rephrasing of natural language specifications (adds tracability of requirements)



# Issues view

## ► Backwards tracing – explanation of non-compliance

CAD program showing issues in layout plan



CNL debug view paraphrased text and translations



Original text highlighting source of paraphrased text

|   | Category | Description   |
|---|----------|---|
| ⚠ | Signal   | No interlocking defined.                                      |
| ✖ | Signal   | The distance from a train detector to another must be greater |

### **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, End

### **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**

- No detection sections shall be shorter than 21 meters.**
- No dead zone shall be longer than 3 meters.

# Advantages

RailCNL as a front-end for property input for verification:

- ▶ RailCNL is **domain-specific**: tailored to Datalog logic and regulations terminology. Gives **readability** and **maintainability**.
- ▶ Resembles **natural language** – improves **readability** and engineer **participation**.
- ▶ Separate textual explanation (such as comments used in programming) are typically not needed.
- ▶ RailCNL statements are **linked** 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.

# Further challenges and future work

## Participatory verification:

- ▶ RailCNL is a common language **shared** between programmers and railway engineers for verification work.
- ▶ CNLs are not a magical solution to end-user programming.
- ▶ DSLs **evolve** along-side the application.

## Language:

- ▶ Structures in regulations that span several phrases/rules (**scopes, exceptions**) – represent on textual or GUI level?
- ▶ Macros – can users extend the language within the scope of their texts?

## Tool support:

- ▶ Can railway engineers from other disciplines create their properties themselves, from scratch, with editor support?
- ▶ Is example-based and editor-supported language learning good enough?

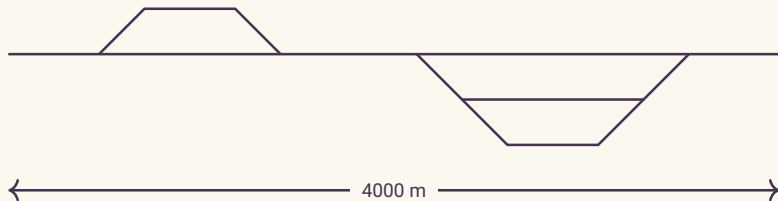
# **Capacity** verification

local capacity specifications

synthesis and optimization

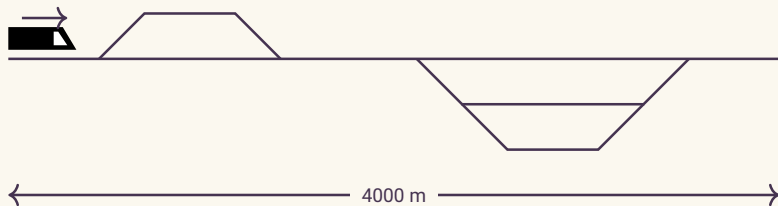
# Railway control systems

Constructing a new railway line starts with a **track plan**:



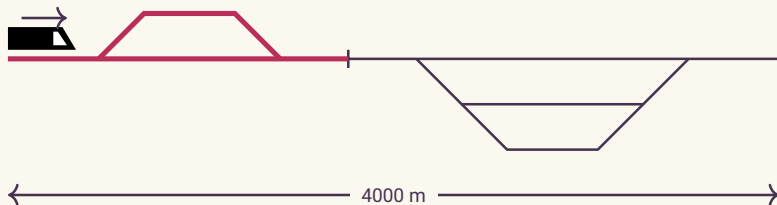
# Railway control systems

Constructing a new railway line starts with a **track plan**:



# Railway control systems

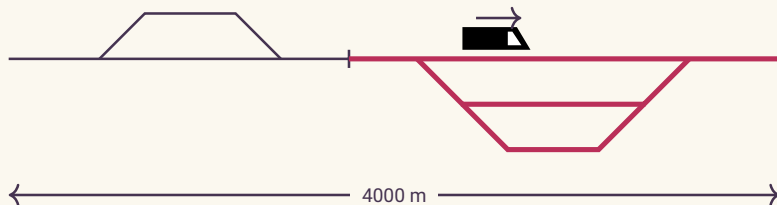
By adding **detectors**, we can allocate smaller pieces of tracks to the train:





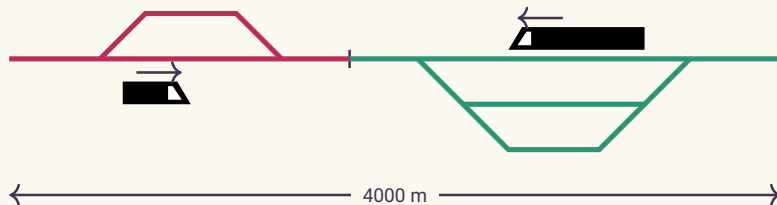
# Railway control systems

By adding **detectors**, we can allocate smaller pieces of tracks to the train:



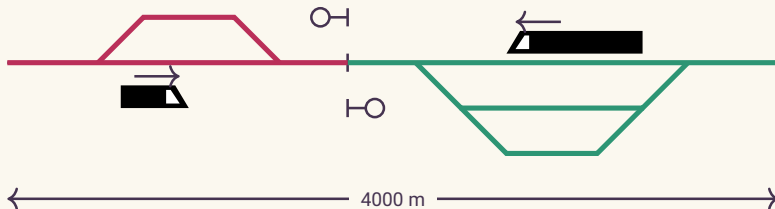
# Railway control systems

Now, **other trains** can occupy different sections.



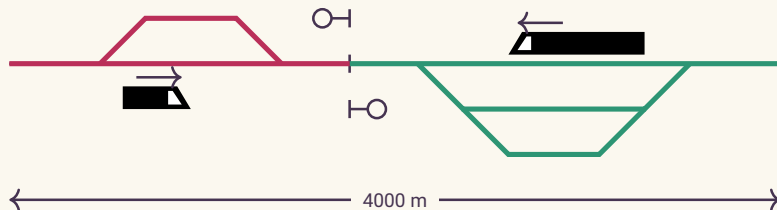
# Railway control systems

We add **signals** to indicate to drivers when they can proceed.



# Railway control systems

This situation is in principle **safe**, but is it a **good design**?



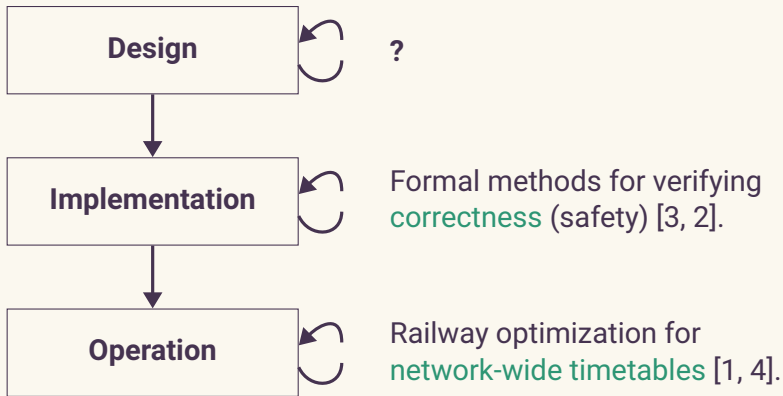
# Requirements

Will my **station design** handle the  
actual **traffic**?

Two methods used in practice:

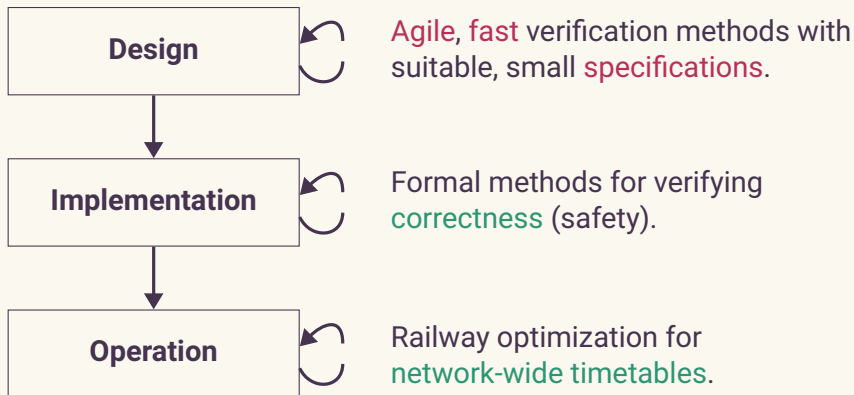
1. Whole-network **time table** analysis: a whole discipline in itself – complicated theory and software
2. Manual, **ad-hoc** analysis: varying quality, little documentation, low repeatability.

# Design-implementation-operation



- [1] M. Abril, F. Barber, L. Ingolotti, M.A. Salido, P. Tormos, and A. Lova. An assessment of railway capacity. *Transportation Research*, 44(5):774 – 806, 2008.
- [2] Arne Borälv and Gunnar Stålmärck. Formal verification in railways. In *Industrial-Strength Formal Methods in Practice*, pages 329–350. Springer, 1999.
- [3] A. Fantechi, W. Fokkink, and A. Morzenti. Some trends in formal methods applications to railway signalling. In *Formal Methods for Industrial Crit Sys.*, 2012.
- [4] Alex Landex. *Methods to est. railway cap. and passenger delays*. PhD thesis, 2008.

# Design-implementation-operation



# Specification capture

Railway engineers gave us examples of **performance properties** that governed their designs.

Typical categories:

1. Running time (get from A to B)
  - Similar to a simulation test, but smaller specification.
2. Frequency (several consecutive trains)
  - Route trains into alternate tracks.
3. Overtaking
4. Crossing
  - Let one train wait on a side track while another train passes.



# Capacity specifications

**Local** requirements suitable for construction projects.

- ▶ Operational scenario  $S = (V, M, C)$ :
- ▶ **Vehicle types**  $V = \{(l_i, v_i^{\max}, a_i, b_i)\}$ , defined by length, max velocity, max accel, max braking.
- ▶ **Movements**  $M = \{(v_i, \langle q_i \rangle)\}$ , defined by vehicle type  $v$  and ordered sequence of visits  $\langle q_i \rangle$ .
  - ▶ Each **visit**  $q_i = (\{l_i\}, t_d)$  is a set of alternative locations  $l_i$  and an optional dwelling time  $t_d$ .
- ▶ **Timing constraints**  $C = \{(q_a, q_b, t_c)\}$  which orders two visits and sets a maximum time from the first to the second  $t_{q_a} < t_{q_b} < t_{q_a} + t_c$ . The maximum time constraint can be omitted ( $t_c = \infty$ ).

# Constraints

**Verification** of these **specifications** would involve finding satisfying **train trajectories** and **control system state**:

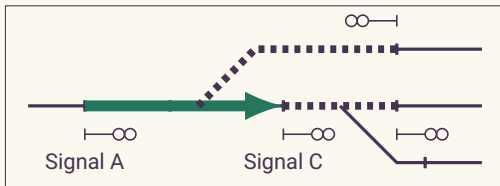
$$\exists p : \text{spec}(p)$$

Also, constrained by:

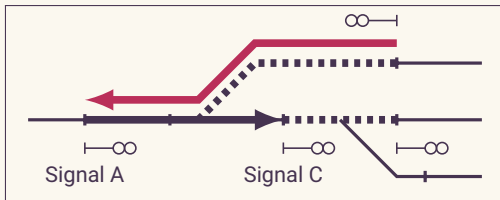
- ▶ 1 - Physical infrastructure
- ▶ 2 - Allocation of resources (collision safety)
- ▶ 3 - Limited communication
- ▶ 4 - Laws of motion

## Constraints (2) Allocation of resources

An **elementary route** is a set of resources allocated together.

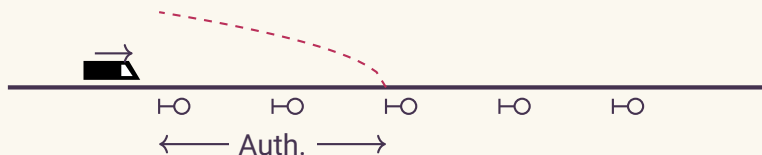
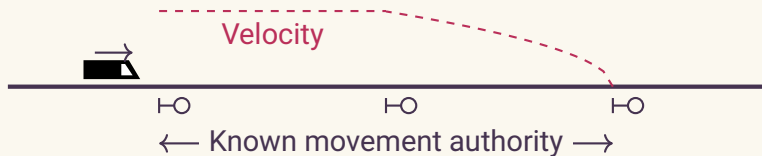


Routes are **conflicting** if they use any of the same resources.



## Constraints (3) Limited communication

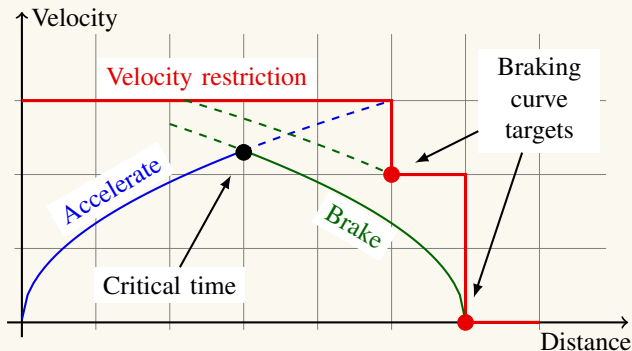
Signal information only carries across two signals ("pre-signalling").



## Constraints (4) Laws of motion

Trains move within the limits of given maximum acceleration and braking power. Train drivers need to plan ahead for braking so that the train respects its given movement authority and speed restrictions at all times.

$$v - v_0 \leq a\Delta t, \quad v^2 - v_i^2 \leq 2bs_i.$$



# Automated verification

Design-time capacity verification amounts to **planning** in a **mixed discrete/continuous** space.

Some suggestions:

- ▶ **PDDL+**, planning domain description language for mixed discrete-continuous planning domains [1].
- ▶ **SMT** with **non-linear real** arithmetic [2, 4].
- ▶ **dReal**:  $\delta$ -complete decision proc. for FOL with reals [3].

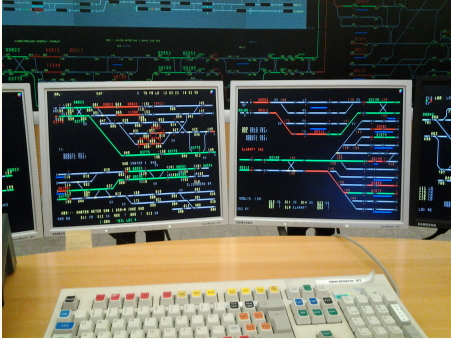
Using these tools/techniques and straight-forward modeling did **not** make our problem manageable on relevant scales.

- [1] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.
- [2] M. Franzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *J. SAT*, 1:209–236, 2007.
- [3] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. CADE-24 vol. 7898 of *LNCS*, pages 208–214. Springer, 2013.
- [4] D. Jovanovic and L. de Moura. Solving non-linear arithmetic. *ACM Comm. Computer Algebra*, 46(3/4):104–105, 2012.

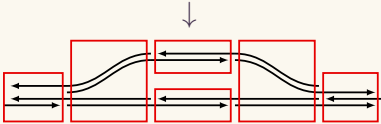
# Dispatch vs. driver

Split the planning work into two separate points of view:

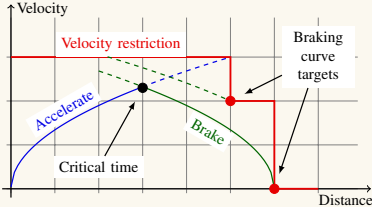
### Dispatcher



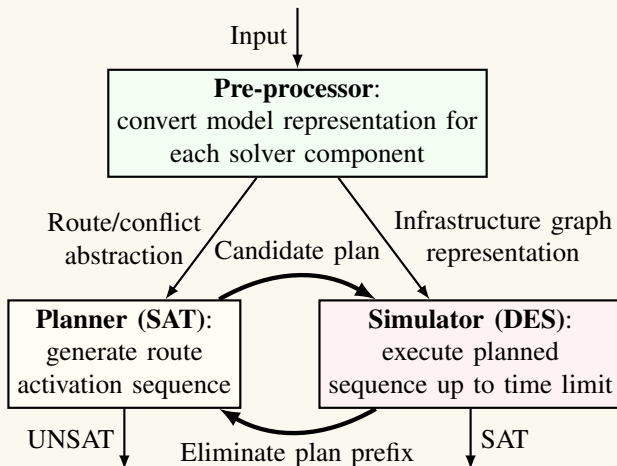
### Train driver



Elementary routes and their conflicts



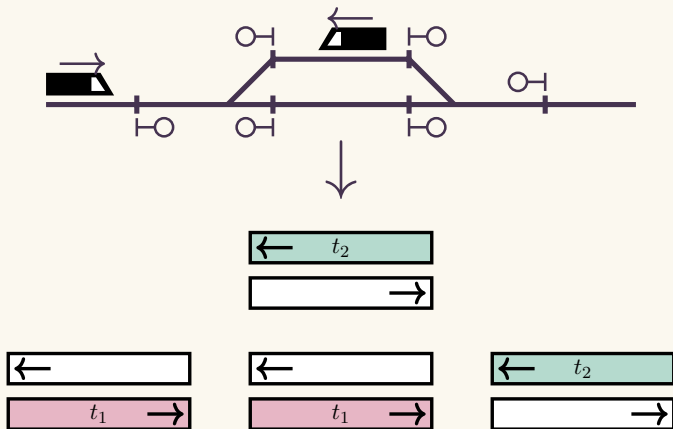
# Solver architecture





# SAT encoding of dispatch planning

General idea: represent **which train** occupies **which elementary route** in each of a sequence of **steps**.



# SAT encoding

Planning as **bounded model checking** (BMC). Build planning steps as needed using **incremental** SAT solver interface.

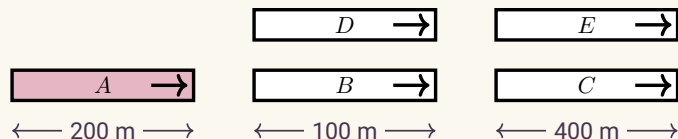
Movement correctness:

- ▶ **Conflicting** routes are not active simultaneously  
 $\text{conflict}(r_1, r_2) \Rightarrow o_{r_1}^i = \text{Free} \vee o_{r_2}^i = \text{Free}.$
- ▶ Elementary route allocation is **consistent** with train movement:  $(o_r^i \neq t \wedge o_t^{i+1} = t) \Rightarrow \bigvee \{o_{r_x}^{i+1} = t \mid \text{route}(r_x), \text{entry}(r) = \text{exit}(r_x)\}$

Satisfy specification:

- ▶ Visits happen in order (timing requirement is measured on simulation).

# Freeing



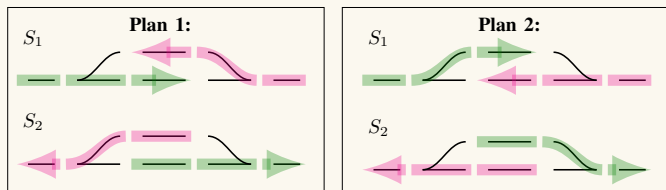
If  $A$  holds a train  $t$  of length 200.0 m, freeing  $A$  is constrained by:

$$A^i \Rightarrow (A^{i+1} \vee (B^i \wedge C^i) \vee (D^i \wedge E^i)).$$

# Eliminate equivalent solutions

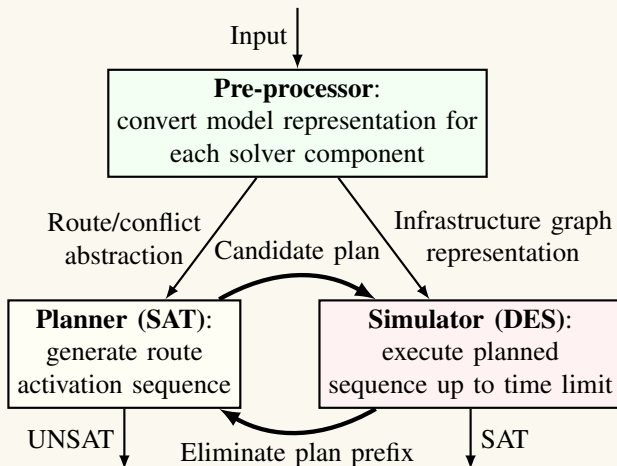
- ▶ Can free  $\Rightarrow$  must free
- ▶ Can allocate  $\Rightarrow$  must allocate
- ▶ Exception to allocation: **deferred progress**  
a train may be waiting for a conflict to be resolved, even if the conflict starts in the future.

Crossing example: **exactly two** solutions:



- ▶ Overlaps. Partial release.
- ▶ Loops in the infrastructure / loops in the dispatch.

# Solver architecture

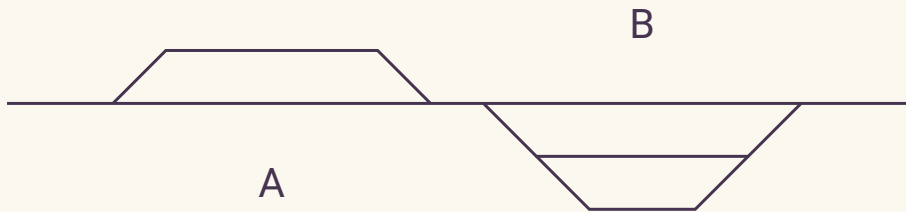


# New design process

Interactive design session:

✓ Running time

✗ Crossing on A

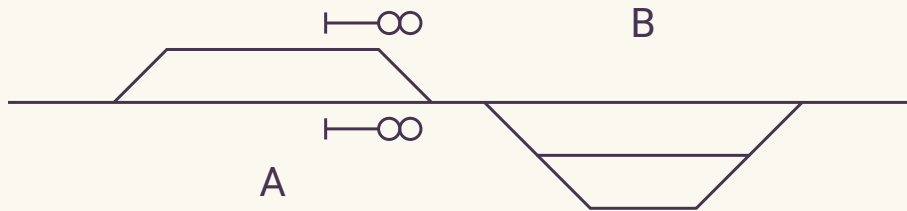


# New design process

Interactive design session:

✓ Running time

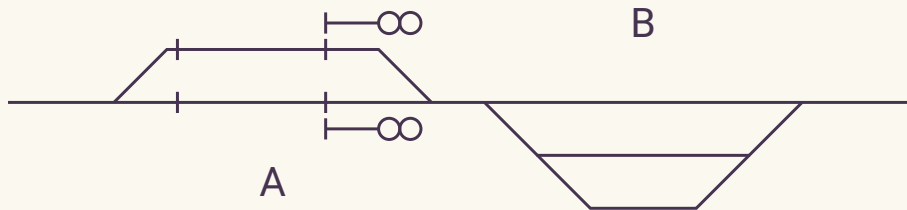
✗ Crossing on A



# New design process

Interactive design session:

- ✓ Running time
- ✓ Crossing on A



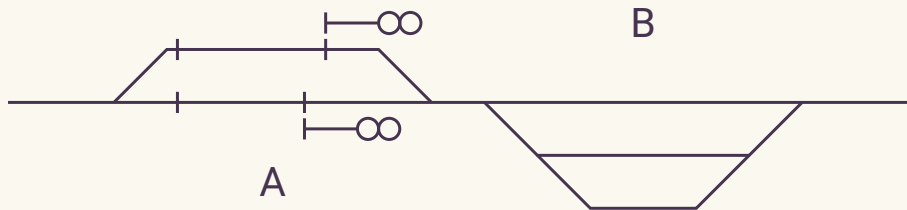


# New design process

Interactive design session:

✓ Running time

✗ Crossing on A



## Conclusions

- ▶ Formalized capacity specifications for **construction** projects.
- ▶ Verification by discrete planning and simulation: **abstract** away from continuous time, distance, velocity.
- ▶ In practical cases: naive refinement works well enough.

## Future work

- ▶ Improved abstraction refinement? Needs **difficult** cases.
- ▶ Integrate with **graphical engineering editor**.
- ▶ Interface with **commercial simulators**.



# Satisfiability queries in system design

System properties can be:

- ▶ **Qualitative**: the system has or does not have the property.  
Ask whether properties are satisfied: **satisfiability query**.
  - Railway regulations.
- ▶ **Quantitative**: the system has more or less of the property.  
Measure by **objective function**.
  - Railway capacity.

Qualitative properties are **modular**.

# SAT-based algorithms

If your problem is not (efficiently) expressible as SAT:

- ▶ generate an **abstracted** problem as SAT
- ▶ ... meaning that you leave out variable or constraints
- ▶ ... **preserving UNSAT** results, but not SAT results.
- ▶ For SAT results, check whether the abstracted system model are still valid in the full model.
- ▶ If not, add variables or constraints to the SAT system that eliminates this mismatch.

This technique has given rise to a wide range of **SAT-based algorithms**.

# SAT-based algorithms

SAT-based algorithms have various levels of sophistication:

- ▶ **Generate and test** (Add the negation of the **current solution**.)
- ▶ **Lazy constraints** (E.g. non-cyclicity constraints, add path cycles.)
- ▶ **Lazy SMT / Fully lazy SMT** (Use theory knowledge to create new **constraints** in the SAT abstraction.)
- ▶ **Counter-example guided abstraction refinement** (Use full system model to create **new variables and/or constraints**.)

# SAT-based local railway capacity verification algorithm

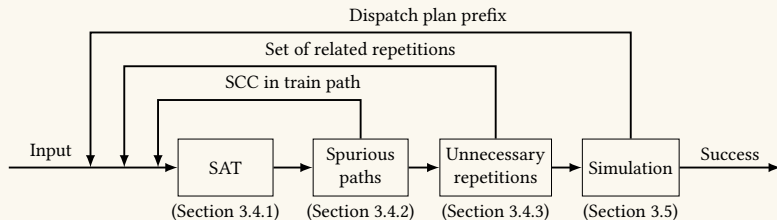


Figure 3.12: Main algorithm for local capacity verification (extended from Figure 3.4) with two more tests for handling loops and repetitions.

# Schematic drawings

- ▶ **Schematic** drawings with linearly ordered nodes  
 $x_0 < x_1 < x_2 < x_3 < \dots$
- ▶ Optimize **size** and **simpleness**.

