Practical formal methods in railways - the SafeCap approach

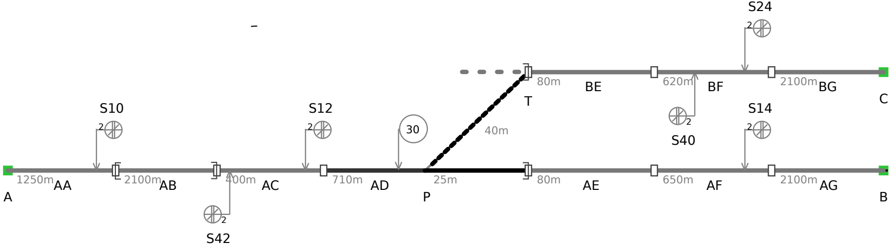
Alexei Iliasov, Ilya Lopatkin, Alexander Romanovsky

Newcastle University

26 June 2014

Railway signalling verification

Railway modelling

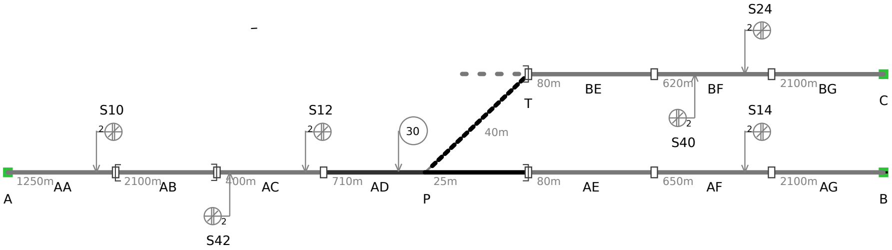


point P	:	clear(AD)
route $\mathrm{A\_S10}$	:	clear$(\mathrm{AA}, \mathrm{AB})$
route $\mathrm{S10\_S12}$	:	clear$(\mathrm{AB}, \mathrm{AC}, \mathrm{AD}) \wedge$ locked(P)
route $\mathrm{S12\_S14}$	:	clear$(\mathrm{AD}, \mathrm{AE}, \mathrm{AF}, \mathrm{AG}) \wedge$ normal$(\mathrm{P}) \wedge$ normal(T)
route $\mathrm{S12\_S24}$	:	clear$(\mathrm{AD}, \mathrm{BE}, \mathrm{BF}, \mathrm{BG}) \wedge$ reverse$(\mathrm{P}, \mathrm{T}) \wedge$ occupied$(\mathrm{AC}{:}15)$
...	:	...

Verification concerns

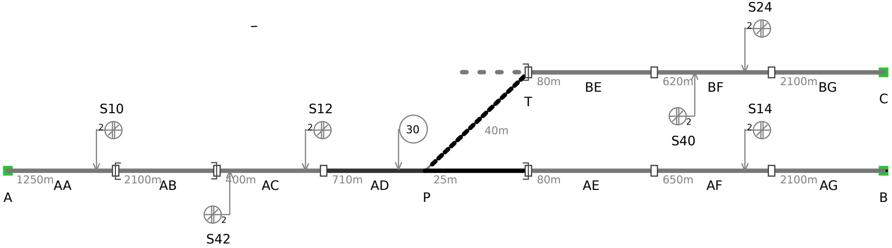
1. A schema must be free from collisions
2. Flank protection
3. Physical layout properties
4. Quality of service

Collisions freeness



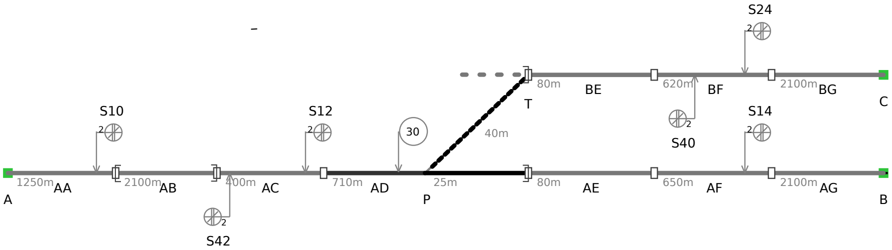
- no two trains may even potentially occupy same track
- discrete section of train occupation detection
- laws of train movement and assumptions about train driver
- principal way of assurance: route locking and holding

Derailments



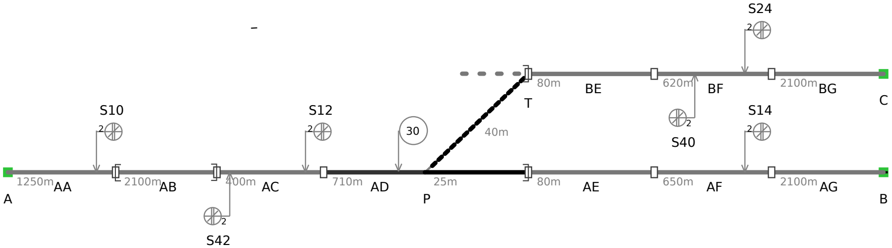
- over-speeding (esp. curved track)
- unlocked, moving or misconfigured point
- laws of train movement and assumptions about train driver
- principal way of assurance: coordinate speed control and point locking with route locking

Flank protection



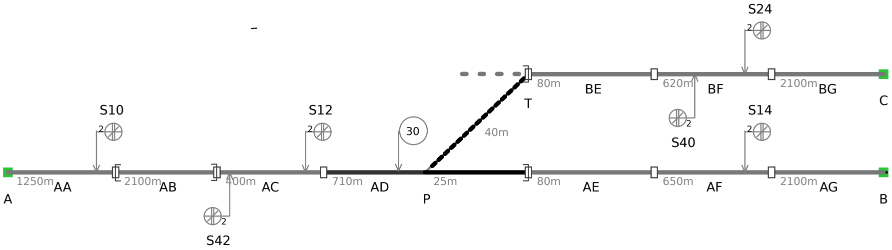
- essential in presence of gradients
- drastic solution: trap points
- principal way of assurance: extending locking area to neighbouring points, additional overlaps

Geographical layout properties



- gradients (esp. near platforms)
- signal sighting distance
- fouling points
- positioning of signals, speed limits, platforms, authority delimitations, ...
- principal way of assurance: verification of topology constraints

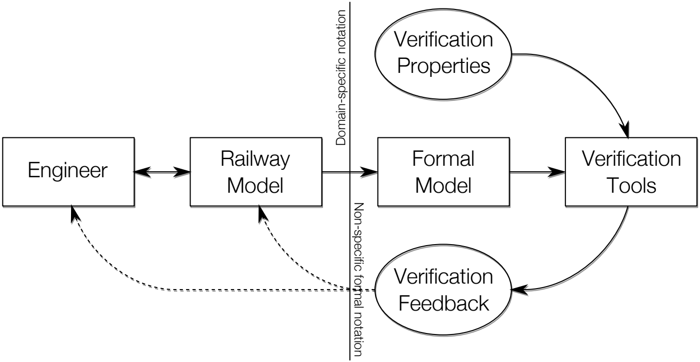
Quality of service



- traffic density, timetable, capacity utilisation
- energy efficiency
- stability
- principal way of assurance: computer simulation

Verification and validation techniques

Principal actors and flow

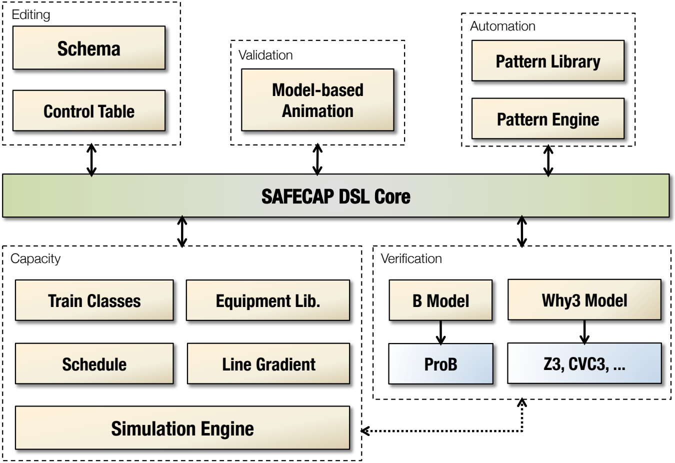


Characterisation of major verification techniques

	Review	*Simulation*	*Theorem proving*	*Model checking*	*SafeCap*
Rigour	$--$	$--$	$++$	$+$	$+$
Productivity	$--$	$\sim$	$--$	$+$	$++$
Expertise	$--$	$++$	$--$	$+$	$+$
Scalability	$+$	$-$	$++$	$--$	$+$
Expressiveness	$++$	$++$	$\sim$	$-$	$-$
Feedback	$+$	$++$	$-$	$+$	$\sim$

SafeCap Platform

SafeCap Platform architecture



Event-B

```
machine m0
  variables balance
  invariant balance ∈ CLIENT → ℕ
  events
     payin =
     any a, c where
        a ∈ ℕ
        c ∈ CLIENT
     then
        balance(c) := balance(c) + a
     end
     withdraw =
     any a, c where
        a ∈ ℕ
        c ∈ CLIENT
        balance(c) ≥ a
     then
        balance(c) := balance(c) − a
     end
end
```

refinement m0

variables *history*, *len*

invariant

$$history \in CLIENT \rightarrow (\mathbb{N} \mathrel{+\!\!\!\!\to} \mathbb{Z})$$
$$len \in CLIENT \rightarrow \mathbb{N}$$
$$\forall c \cdot c \in CLIENT \Rightarrow dom(history(c)) = 0 \mathinner{..} len(c) - 1$$
$$\forall c \cdot c \in CLIENT \Rightarrow \sum(history(c)) = balance(c)$$

events

payin =

any a, c **where**

$$a \in \mathbb{N} \wedge c \in CLIENT$$

then

$$history(c) := history(c) \mathbin{\lhd\!\!\!-} \{len(c) \mapsto a\}$$
$$len(c) := len(c) + 1$$

end

withdraw =

any a, c **where**

$$a \in \mathbb{N} \wedge c \in CLIENT \wedge \sum(history(c)) \geq a$$

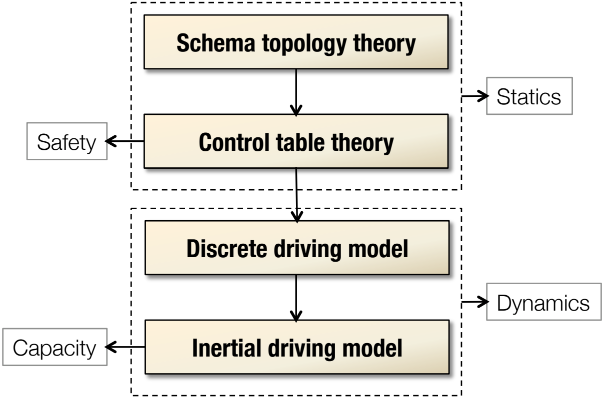
then

$$history(c) := history(c) \mathbin{\lhd\!\!\!-} \{len(c) \mapsto -a\}$$
$$len(c) := len(c) + 1$$

end

end

Verification layers



Discrete driving model

- Capturing train, signal and point behaviour
- *Safety invariants* corresponding to
  - Collisions freeness
  - Derailments
  - Flank protection
- Modelling train movement, route reservation, point locking, route cancellation and so on
- *Inertia-less trains*

machine route0
  sees ctx_line
  variables
    t_line // Train/line association
    t_r_hd // Train head position on a line
    t_r_tl // Train tail position on a line
  invariant
    $t\_line \in TRAIN \rightarrow\!\!\!\!\!+ LINE$
      // A train is mapped to the id of a route occupied by the head of a train
    $t\_r\_hd \in TRAIN \rightarrow\!\!\!\!\!+ \mathbb{N}_1$
      // correspondingly, $t\_r\_tl(t)$ is the id of the route occupied by the tail of train t
    $t\_r\_tl \in TRAIN \rightarrow\!\!\!\!\!+ \mathbb{N}_1$
    $\mathrm{dom}(t\_line) = \mathrm{dom}(t\_r\_hd)$
    $\mathrm{dom}(t\_line) = \mathrm{dom}(t\_r\_tl)$
      // A train occupies a continuous route interval of route from tail till head
    $\forall t \cdot t \in dom(t\_line) \Rightarrow t\_r\_tl(t) .. t\_r\_hd(t) \neq \varnothing$
    The routes a train occupies are the routes defined by the train line
    $\forall t \cdot t \in dom(t\_line) \Rightarrow t\_r\_tl(t) .. t\_r\_hd(t) \subseteq dom(Line(t\_line(t)))$
   // Initially, there are no trains in the system
  initialisation
    $t\_line, t\_r\_hd, t\_r\_tl := \varnothing, \varnothing, \varnothing$
  events

Topology model

- Verifying logical conditions expressed over track layout: track connections, point placement, ...
- Cross-checking logical topology (i.e., routes and lines as paths through a schema)
- Validation of platform placement

```
/* (1)  */  {} <<: NODE  &
/* (2.a)  */  {} <<: TRACK  &
/* (2.b)  */  TRACK <: NODE * NODE  &
/* (2.c)  */  elm(TRACK) = NODE  & /* all nodes are connected by tracks */
...
/* (10) */ AMBIT : LA --> (POW(NODE) * POW(TRACK)) &
/* (11)  */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p ) => p <: q * q & {} <<:
/* (12) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p) => p~ <: p) &
/* (13) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p) =>
                ! (n) . (n : q => closure(p)[{n}] = q) ) &
/* (14)  */ union({p | # (a, q) . ( a : ran(AMBIT) & q <: NODE &
                p <: TRACK & a = ( q |-> p))}) = TRACK &
/* (15)  */ ! (a, b, r, s, t, q) . (a : ran(AMBIT) & b : ran(AMBIT) & a /= b &
                a = (r |-> s) & b = (t |-> q) => s /\ q = {}) &
...
```

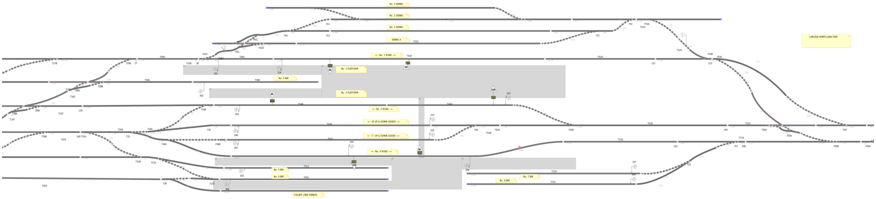
Signalling verification

- Conditions of *operational safety*
- Formally derived from the discrete driving model
- No dynamics - just static constraints on data (control table)
- Tuned for constraint solving

```
...
/* @label (CT.1): A permissive signal may be lit only when all route ambits are clea
! (l, r). (l |-> r : CT0_DOM => ! (n). (n : 1 .. RASPECT(l, r)-1 =>
                  routeambits(r) <: CT_CLEAR(l, r, n) )) &
/* @label (CT.2): A route with an overlap may have permissive signal only
 when its overlap is reserved and confirmed as clear  */
! (l, r). (l |-> r : ROVERLAP & r : dom(LINE(l)) => ! (n). (n : 1 .. RASPECT(l, r)-
                  TA[fst(ROUTE(LINE(l)(r)))] <: CT_CLEAR(l, r, n))) &
/* @label (CT.3.a): No point is set both normal and reverse */
! (l, r). (l |-> r : CT0_DOM => CT_NORMAL(l, r) /\ CT_REVERSE(l, r) = {} ) &
...
```

Experimental results & Demo

Case study: Carlisle Citadel



Experimental results

Benchmark	*Points/Lines/ Routes*	*Conditions, topology*	*Conditions, control table*	*Run time, topology*	*Run time, control table*
Station 1	8/12/14	117	230	4s	2s
Junction 1	23/4/21	280	602	24s	8s
Station 2	6/23/21	104	678	18s	6s
Carlisle, west	24/112/30	350	888	1m 17s	12s
Carlisle	63/161/79	892	1270	6m 4s	19s

Table : Verification run times for several sample layouts.

Questions?