Deductive Verification of Railway Operations

Eduard Kamburjan and Reiner Hähnle

15. November 2017

Technische Universität Darmstadt, Software Engineering Group

Big Picture

Railway Engineering as Software Engineering

Apply tools for distributed software to railway operation procedures.

Infrastructure	=	Data Structure
Operational Rules	=	Programs
Safety Property	=	Logical Formula

Focus: Communication and PZB

Deductive Verification

Definition

"... methods that use an expressive (at least first-order) logic to state that a given target system is correct with respect to some property. Logical reasoning (deduction) is then used to prove validity of such a statement ..."[Beckert and Hähnle 2014]

Program Logics and Traces

First Order Program Logic and specification of traces.

Modeling Railway Operations vs. Modeling Railway Systems

Approach

- Model railway systems as a distributed software system
- Modeling not on implementation level
- Instead: Information flow as described in rule books
- Verify safety properties for all well-formed infrastructures

Modeling Language

Abstract Behavioral Specification Language

- Based on actors and cooperative scheduling
- Executable modeling language
- Multiple static analyses available

```
class Station implements StationInterface{
   StationInterface next = ...
   Unit schedule(Event ev){
       Fut<Int> f = next!request(train(ev));
       this.id = id(ev);
       f.get;
       Int i = await next!request(train(ev));
   }
}
```

Events

A trace is a sequence of events which encodes visible actions

- Invocation (invocEv) - Invocation Reaction (invocREv)
- Suspension (awaitEv) - Reactivation (reacEv)
- Completion (futEv) - Completion Reaction (futREv)

Concurrency system encoded as axioms: e.g.,

- each invocation reaction is predated by invocation

ABS Program Logic

- Connects programs to specifications of state

$$\mathtt{i} \geq 0 \rightarrow [\mathtt{i\ =\ i\ +\ 1}]\mathtt{i} > 0$$

ABS Program Logic

- Connects programs to specifications of state

$$\texttt{i} \geq 0 \rightarrow [\texttt{i = i + 1}]\texttt{i} > 0$$

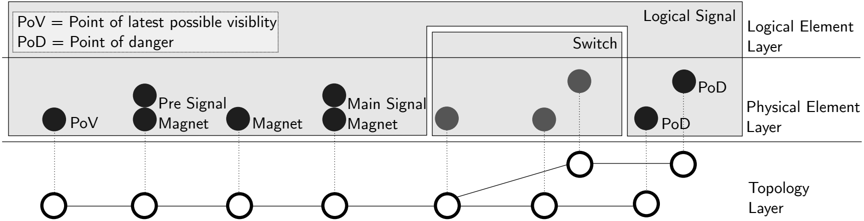
- Describes possible histories of events

$$[\texttt{o!m();}\pi]\phi \leadsto$$
$$\{\texttt{history} := \texttt{history} \circ \langle \mathsf{InvocEv}(X, \texttt{o}, f, \texttt{m}, \epsilon)\rangle\}[\pi]\phi$$

Modeling

Overview

Layers seperate topological aspects from information transmission



Point of Information Flow (PIF)

Infrastructure

Infrastructure is a graph model, where each node has at least one point of information flow.

- Information flow from infrastructure to train
  - signals, magnets,...
- Information flow from train to infrastructure
  - axle counter, balises,...
- Indirect information flow
  - End of switch, crossing,...
- Multiple PIFs per node possible

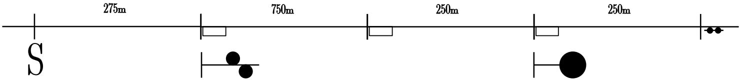
Grouping PIFs in the Graph

Infrastructure

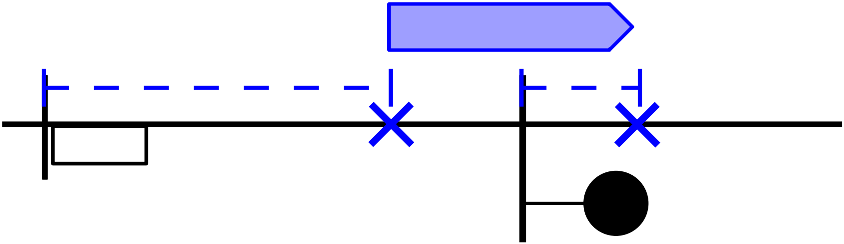
Multiple points of information flow form a logical object, each logical object is assigned to a station

Logical Signal

main signal + pre signal + point of visibility + three magnets + danger point (+ additional signals + ...)



Trains



Code Example

```
class MainSignal(Node n, Edge track, Signal s)
 implements MainSignal {
  SignalState state = STOP;
  Info triggerFront(Train train, Edge e){
    if ( this.track == e ){
      this.s.setObserver(null);
      return Info( this.state );
    }
    return NoInfo;
  }
  Info triggerBack(Train train, Edge e){
    return NoInfo;
  }
  Unit setState(SignalState newState){...}
}
```

Communication

Three communication protocols among stations

- Change of permit - prevents head-on runs
    - One token per line, only station with token can let trains drive
    - Here “Erlaubnisholtaste”: **A** requires token from **B**
    - **A** only requests when all trains from **B** arrived
    - **B** always releases, except when it is about to use token
    - Other protocol verified in [FTSCS 2016].

Communication

Three communication protocols among stations

- Change of permit - prevents head-on runs
  - One token per line, only station with token can let trains drive
  - Here “Erlaubnisholtaste”: **A** requires token from **B**
  - **A** only requests when all trains from **B** arrived
  - **B** always releases, except when it is about to use token
  - Other protocol verified in [FTSCS 2016].
- “Zugmeldebetrieb” - prevents deadlocks
  - Each train is offered and accepted
  - On departure, train is announced

Communication

Three communication protocols among stations

- Change of permit - prevents head-on runs
  - One token per line, only station with token can let trains drive
  - Here “Erlaubnisholtaste”: **A** requires token from **B**
  - **A** only requests when all trains from **B** arrived
  - **B** always releases, except when it is about to use token
  - Other protocol verified in [FTSCS 2016].
- “Zugmeldebetrieb” - prevents deadlocks
  - Each train is offered and accepted
  - On departure, train is announced
- Block signaling - guarantees free block
  - After setting a signal to “Go”
  - Next signal must block back before “Go” can be set again
  - Blocking back is caused by train passage

Code Example

Part of the method that controls departure of trains

```
while (!permission[line]) { //controls departure
    await expectIn[line] == Nil
    lockedLine[line] = True;
    Bool res = await target!reqPermit(this, line);
    if(res) permission[line] = True;
    lockedLine[line] = False;
}
permissionLocked[line] = True;
```

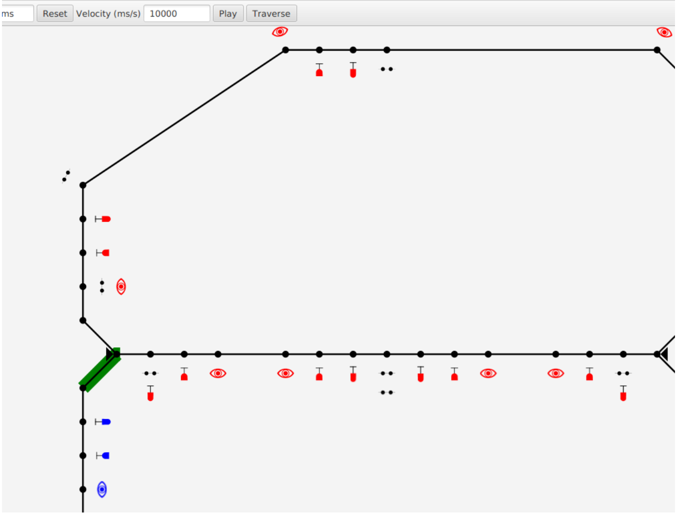
Part of the method that accepts train

```
  await !lockedLine[line] && acquireHalt(line, trackList) != null;
```

Part of `reqPermit`

```
if(permission[line] && !permissionLocked[line]){
   permission[line] = False;
   return True;
}
```

Execution



Code Example

```
productline Examples;
features ETCS1Demo, ETCS2Demo, ETCS3Demo, ...;

delta ETCS1Ex after ETCSRBC when ETCS1Demo;
delta ETCSRBC after ETCSCore
    when ETCS1Demo || ETCS2Demo || ETCS3Demo;
delta ETCSCore
    when ETCS1Demo || ETCS2Demo || ETCS3Demo;

product ETCS1 (ETCS1Demo);
product ETCS2 (ETCS2Demo);
product ETCS3 (ETCS3Demo);

root Scenarios { group oneof { ETCS1Demo,
  ETCS2Demo, ETCS3Demo } }
```

Notion of Safety

Scope

Terminology

- Edge between two PIFs : Track
- Tracks between two signals: Section
- Sections between two stations: Line
- In presentation: between two stations there is only one line

Scope

Terminology

- Edge between two PIFs : Track
- Tracks between two signals: Section
- Sections between two stations: Line
- In presentation: between two stations there is only one line

Assumptions

- German rules differ for driving inside and outside of stations
- We only consider driving outside
- No level crossings
- We ignore faults and assume that infrastructure is well-formed

Well-Formedness

Coherent Encoding

- Relations between elements is coherent, e.g.,
- If a signal is marked as covering section S, then it exists

Correct Encoding

- Designed according to Ril. 819, e.g.,
- Every mainsignal has a presignal

Coherence and Correctness

- Coherence connects the proof to reality
- Correctness connects the safety theorem to reality

Safety on the Line

Theorem (Departure Safety)

If an exit or block signal to section S of line L is set to “Go”:

1. L has no trains driving in the opposite direction
2. S is free from trains going in the same direction

Proof

Methodology

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

Formalize

Reformulate/Connect

Split

Lemma: Permission change (1) – Informal/global state

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

"If station A acquires the token for line L from station B, then there is no train on L towards A."

Lemma: Permission change (2) – Informal/global history

Formalism	**Scale**
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

"If station A has an completion reaction event for B.`reqPermit` *and reads* `True`*, then at this moment A expects no trains on L."*

Lemma: Permission change (3) – Formal/global history

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

Lemma

The following formula holds for all generated histories with a well-formed infrastructure. Let A be a station and L a line with *section* being the first section of L from A and $A.\mathtt{other}(\mathit{section})$ the last.

$$\forall i, f.\ h[i] = \mathrm{futREv}(A, \mathtt{rqPerm}, f, [\mathtt{True}, \mathit{section}]) \rightarrow \\ \sigma[i](A) \models \mathtt{expectIn}(A.\mathtt{other}(\mathit{section})) = \mathtt{Nil}$$

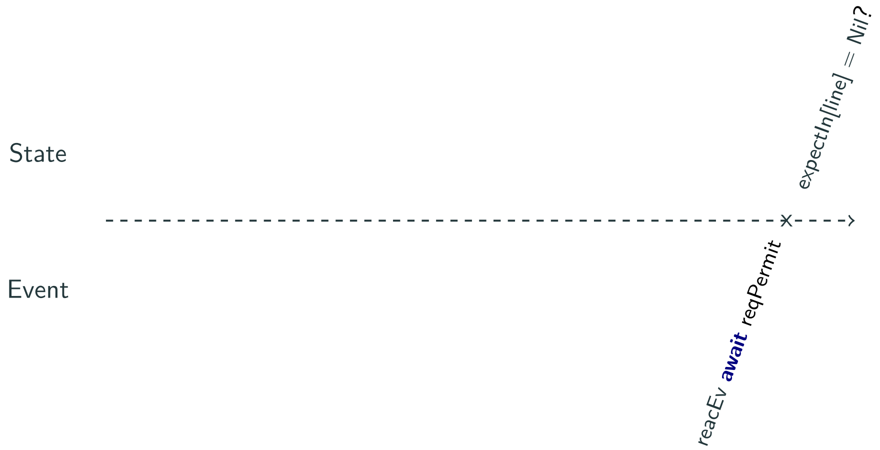
Lemma: Permission change (4)

We identify *section* and $A.\texttt{other}(\textit{section})$ with line

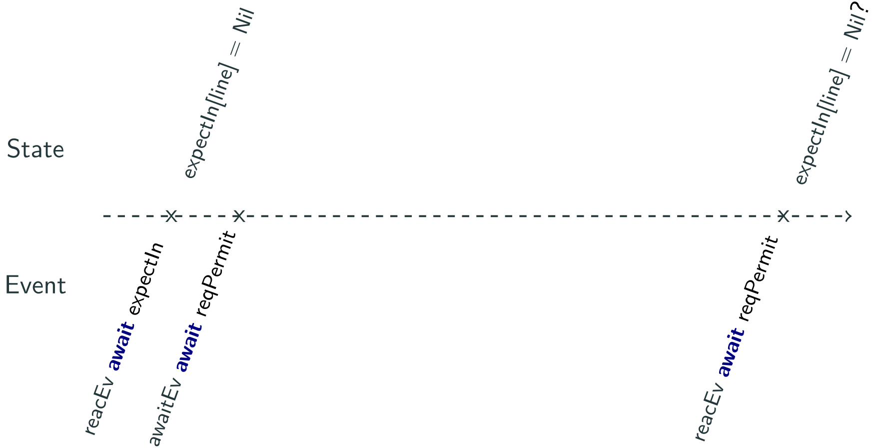
```
while (!permission[line]) { //in method run
    await expectIn[line] == Nil
    lockedLine[line] = True;
    Bool res = await target!reqPermit(this, line);
    if(res) permission[line] = True;
    lockedLine[line] = False;
}
permissionLocked[line] = True;
```

```
Unit offer(Train train, Line line){
  await !lockedLine[line] && acquireHalt(line, trackList) != null;
  expectIn = [train]::expectIn;
}
```

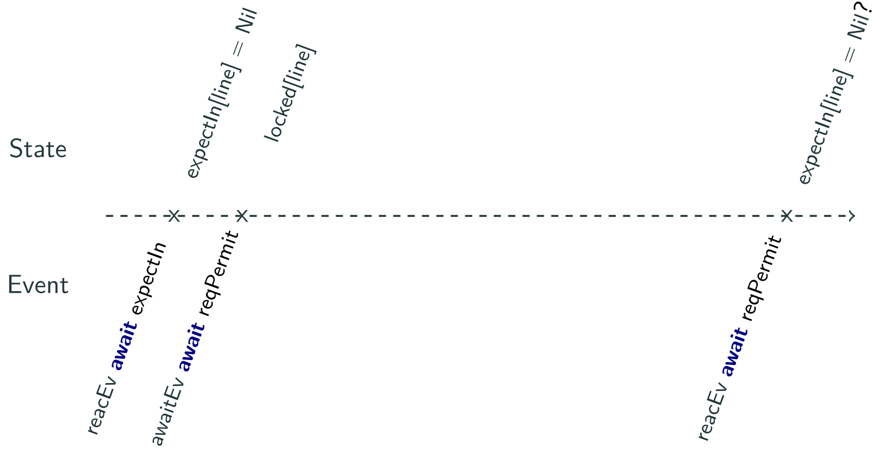
Lemma: Permission change (5) – Proof Outline



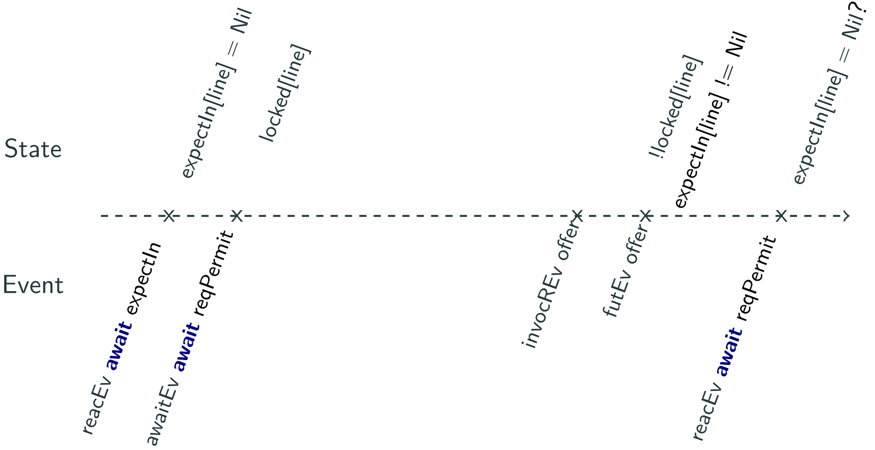
Lemma: Permission change (5) – Proof Outline



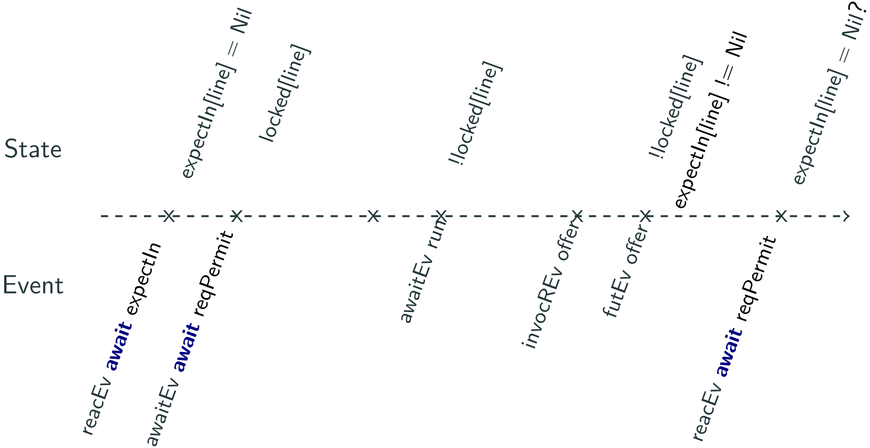
Lemma: Permission change (5) – Proof Outline



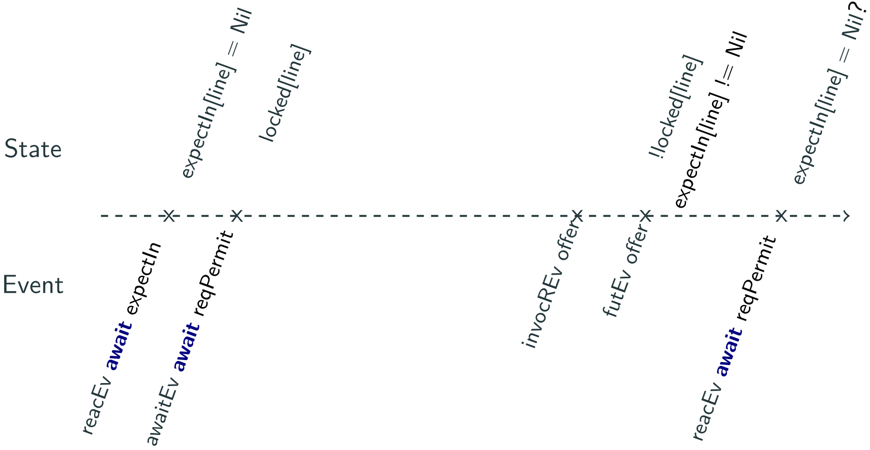
Lemma: Permission change (5) – Proof Outline



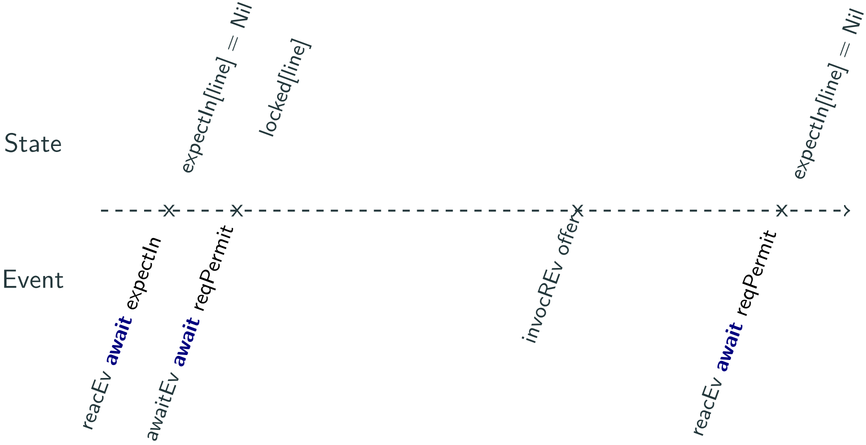
Lemma: Permission change (5) – Proof Outline



Lemma: Permission change (5) – Proof Outline



Lemma: Permission change (5) – Proof Outline



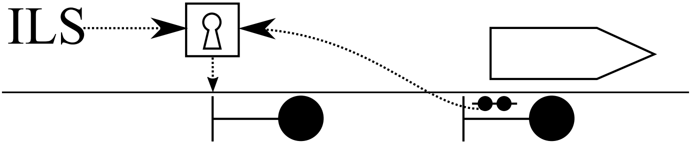
Lemma: Permission change (6) – Formal/local history

- All steps are formal, most are easily verified
- To connect history and state, we can state a local invariant

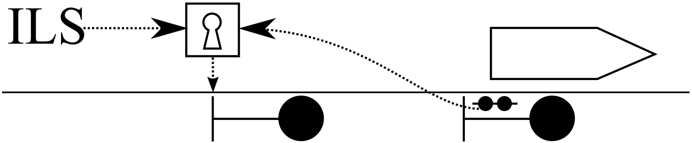
$$\forall \texttt{Train}\ T.\ \forall \texttt{Line}\ \text{line}.$$
$$\mathit{last}(h) = \text{futEv}(\textbf{self}, \texttt{offer}, f, [T, \text{line}]) \rightarrow$$
$$\textbf{self}.\texttt{lockedLine}[\text{line}] = \texttt{False}$$

- Verified with the KeY-ABS theorem prover
- Use similar argument on `permissionLocked` to establish that a train only leaves when the station has the permission

Lemma: Train Involvement (1)



Lemma: Train Involvement (1)



Well-formedness assumption: At $t = 0$, all trains are in stations

Lemma: Train Involvement (2)

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

1) "If a non-entry signal S is set to "Go", then the covered section is free of trains going away from it."

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

2) "If a signal S is set to "Go" twice, then a train triggered the point of danger of the next signal at some time in between."

Lemma: Train Involvement (3)

Formalism	Scale
informal	global state
informal	global history
ABS Program Logic	global history
ABS Program Logic	local history

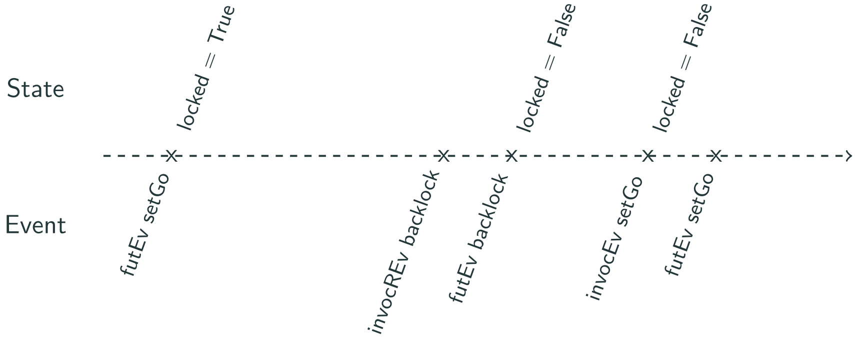
Lemma

The following formula holds for all histories generated by the model in with a well-formed infrastructure. Let A be a station and S a signal.

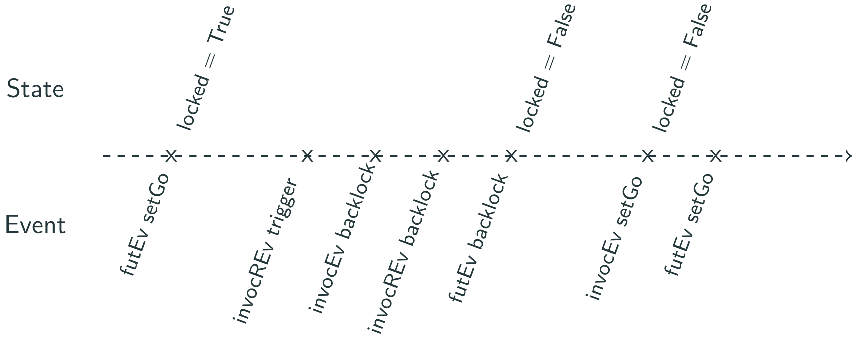
$$\forall i.\ \Big(h[i] = \mathsf{invocREv}(A, S, \mathtt{setGo}, f, []) \rightarrow$$
$$\forall j.\ \big(j < i \wedge h[j] = \mathsf{invocREv}(A, S, \mathtt{setGo}, f', []) \rightarrow$$
$$\exists\ \mathtt{PoD}\ P. \exists k.\ j < k < i \wedge$$
$$h[k] = \mathsf{invocREv}(P, \mathtt{next}(S.covers), \mathtt{trigger}, f'', [])\big)\Big)$$

Lemma: Train Involvement (4) – Proof Outline

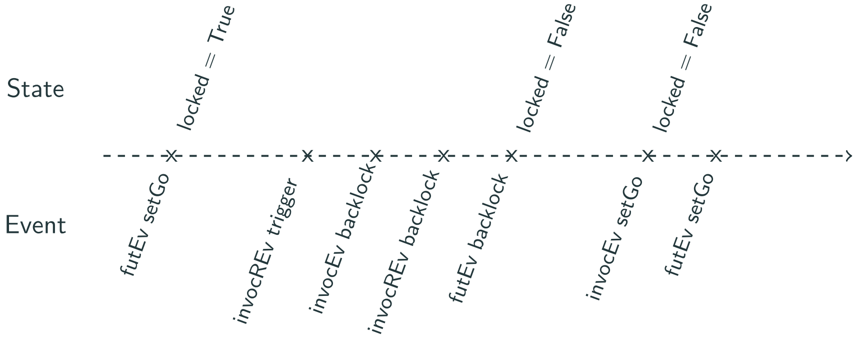
Lemma: Train Involvement (4) – Proof Outline



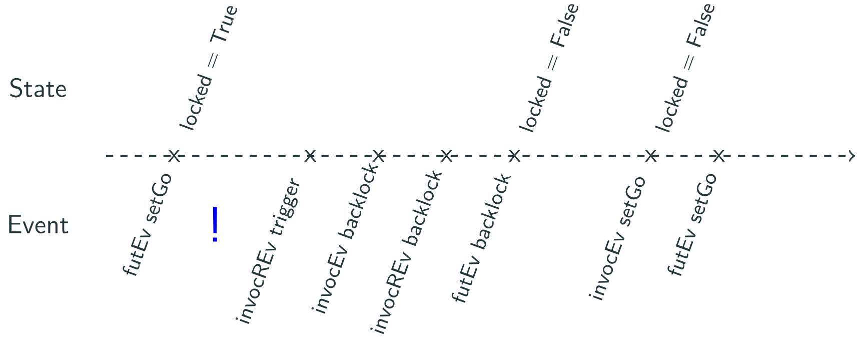
Lemma: Train Involvement (4) – Proof Outline



Lemma: Train Involvement (4) – Proof Outline



Lemma: Train Involvement (4) – Proof Outline



Theorem

Theorem (Departure Safety)

If an exit or block signal to section S of line L is set to "Go":

1. L has no trains driving in the opposite direction
2. S is free from trains going in the same direction

- Additionally to Lemmas, use of well-formedness is needed
- Result holds for any well-formed infrastructure

Conclusion

Aspects of Active Objects and Deductive Verification

+ Verification not bound to concrete infrastructure
+ Verification does not bound size of infrastructure
+ Combines interaction, simulation and formal analysis
- Not fully automatic

Conclusion

Summary

- Deductive Verification allows verification of procedures
- Split safety and well-formedness, needs no concrete infrastructure
- Tools for distributed software generalize to railway operations

Future Work

- Modeling of all relevant rulebooks and infrastructure elements
    - Level crossings, special signals
    - ETCS L2+3
- Safety proofs in presence of faults and inside of stations
- Application of further tools (e.g., deadlock analysis)

Conclusion

Summary

- Deductive Verification allows verification of procedures
- Split safety and well-formedness, needs no concrete infrastructure
- Tools for distributed software generalize to railway operations

Future Work

- Modeling of all relevant rulebooks and infrastructure elements
  - Level crossings, special signals
  - ETCS L2+3
- Safety proofs in presence of faults and inside of stations
- Application of further tools (e.g., deadlock analysis)

Thank you