Closing the Validation Gap or Verifying Railway Interlockings in Agda

Anton Setzer
Swansea University, Swansea UK
Shonan Meeting
Logical Analysis of Descriptions and their Representations
Shonan Village Center, Japan

26 January 2015

Examples of Validation Problems

Closing the Validation Gap

Case Study: Formalisation of Railway Interlocking System

Proof of Safety

Proviso

- Background in mathematical logic, proof theory and type theory.
- Be prepared of misuse or naive use of terminology from software engineering.

Examples of Validation Problems

Closing the Validation Gap

Case Study: Formalisation of Railway Interlocking System

Proof of Safety

Exam Question

- Assume you have two planes:
  - The code for the first one has been **fully verified** using automated and interactive **theorem proving**, but the plane has not been tested.
  - The code for the second one has not been verified this way, but the plane has been **thoroughly tested**.
- Which one do you choose to use?

Validation Gap

- **Verification** can be done in a machine checked way.
- Verification is only relative to a given specification.
- How do you know that the specification guarantees that the program fulfils the requirements?
- **Validation** checks that a program fulfils the requirements or a specification guarantees that the requirements are fulfilled.
  - Cannot be done formally.

Example Incomplete Specification

- We have written a program for controlling a railway interlocking system using SPARK Ada.
- Specification based on Hoare logic (pre and post conditions).
- **Verification** was carried out in a **machine checked** way.
- When running the program it was incorrect.
  - **Trains disappeared**.
  - Forgotten to add to the specification that trains should not get lost.
  - This happened in real world as well (disappearance of trains from a US control system of railways).

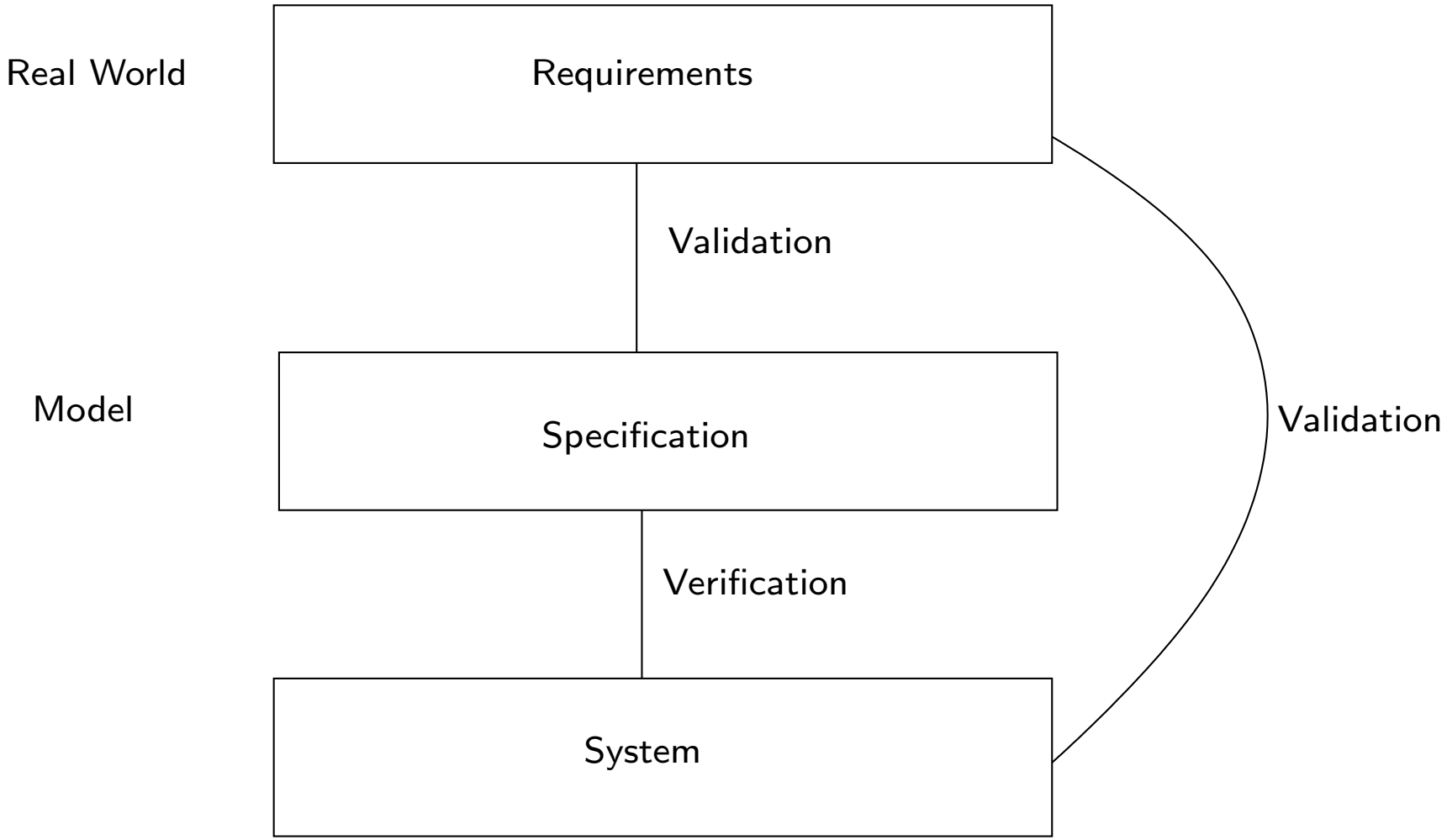
Complexity of Specification

- Tobias Nipkow has verified the security of a hotel key system.
- **Specification was substantially longer than the program**.
- Maybe it is easier to see that the program is secure than that the specification guarantees security?

Closing the Validation Gap

- **Verification** can be done **provably correct** or using systematic thorough testing.
  - We can guarantee (up to a certain degree).
- **Validation** can only be done using semi-formal, systematic methods.
  - We cannot guarantee it.
- We cannot avoid a **gap** between specification and requirements.
- However we can **make the gap as small as possible**.

Requirements - Specification - System



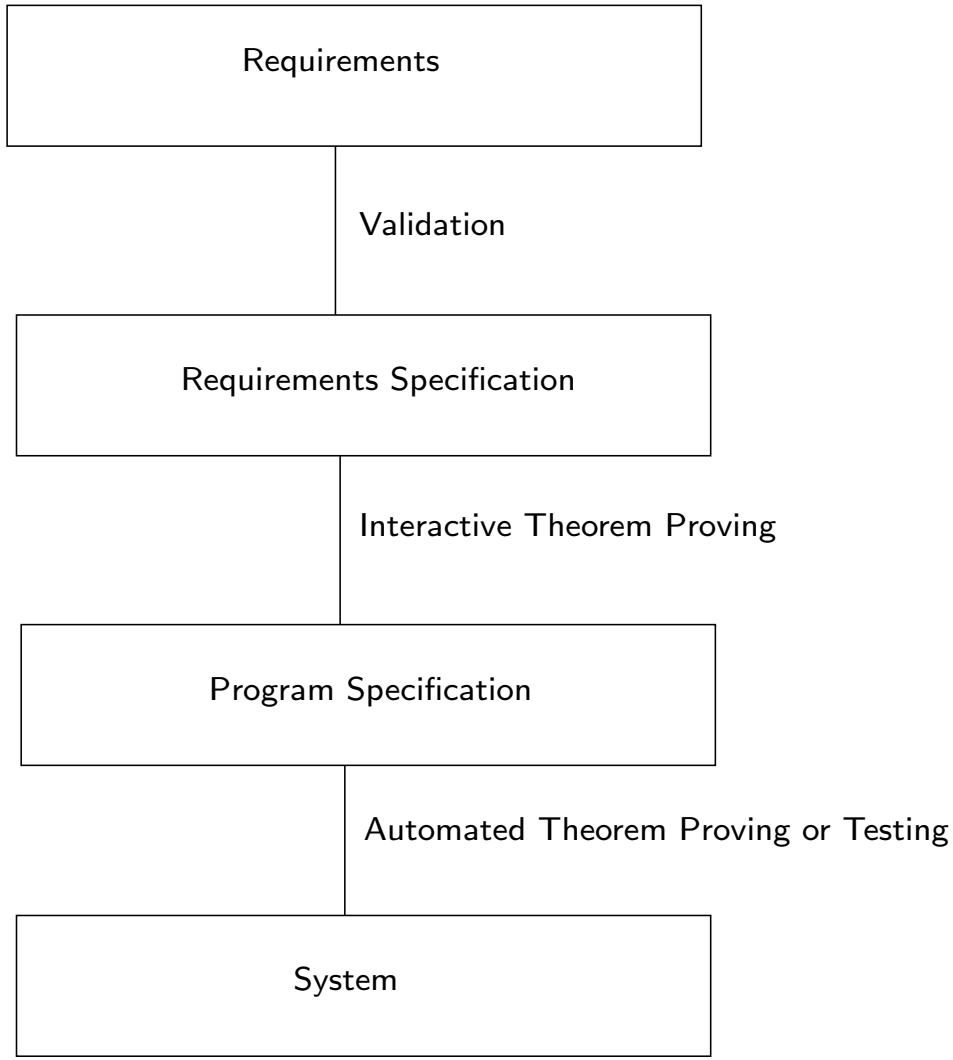
Suggestion to have two Specifications

- **Requirements specification** which is as close as possible to the requirements.
  - Corresponds as close as possible to a model of the real world situation.
  - Example: In railway interlocking systems model of railways.
- **Program specification** which is used to verify the program.
  - Should make it easy to verify that a program fulfils the specification.
  - Example: In railway interlocking systems signalling principles
    E.g.: If signal A is green, signal B is red.

Interactive vs Automated Theorem Proving

- That the **program fulfils the program specification** is typically provable by **automated theorem proving**.
  - In case of railway interlocking systems show that a railway interlocking system fulfils signalling principles.
- That the **program specification implies the requirements specification** is typically provable by **interactive theorem proving**.

Requirements and Program Specification

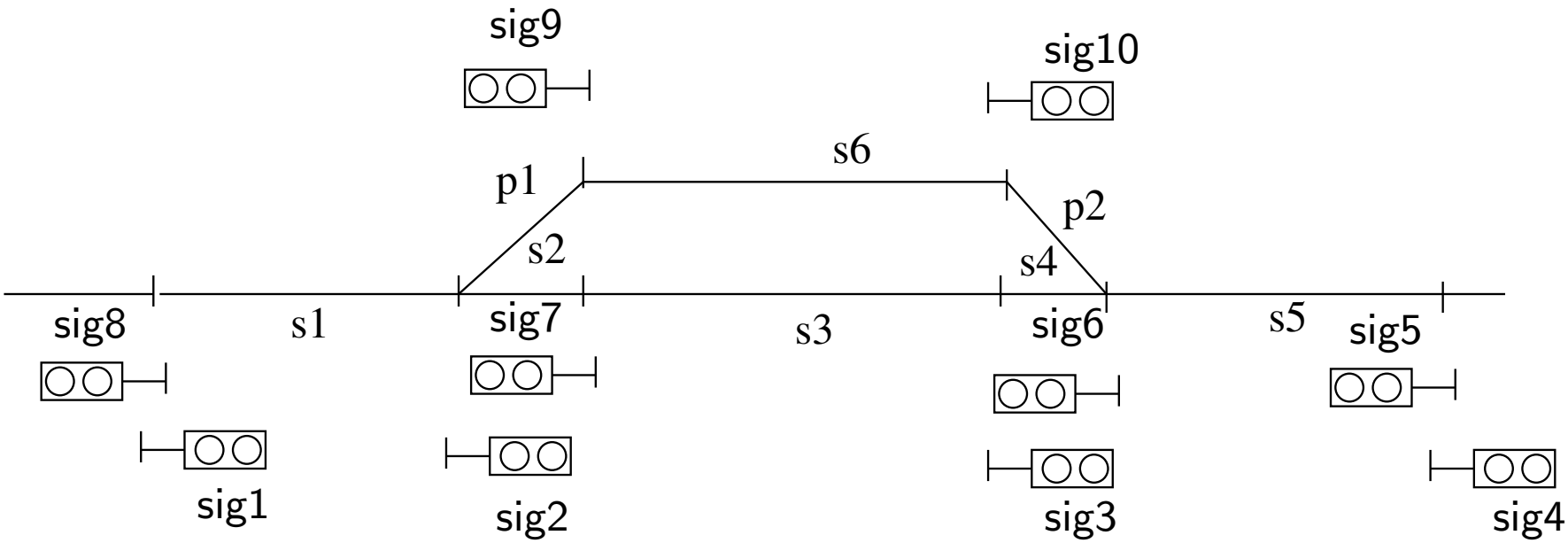


Track Segments

- The basic unit into which one divides a rail yard is that of a **track segment**.
- A track segment is stretch of a track without any further smaller parts, which are significant for an analysis of a interlocking system.
  - there are no sets of points in between (but a set of points might form one segment)
  - there are no crossings in between,
  - they are not divided by signals into parts.

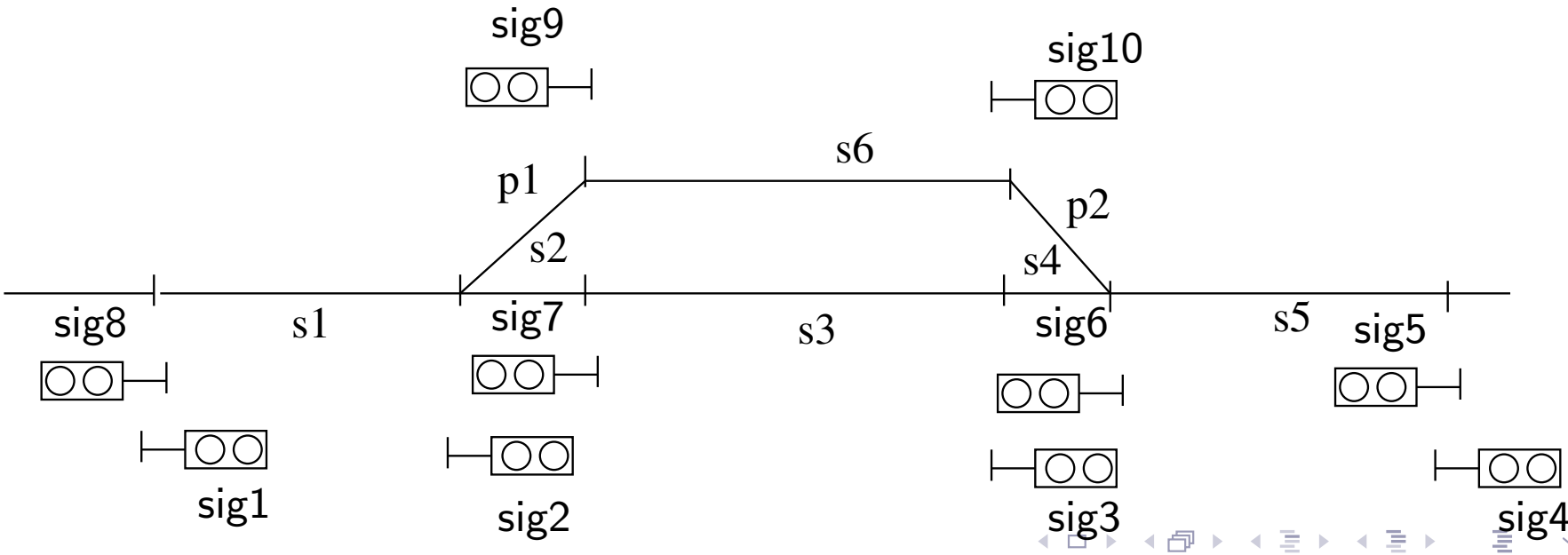
Example

- In the following example we have track segments s1 - s6.
- The two branches of the set of points p1 form segment s2.
- The two branches of the set of points p2 form segment s4.



Signals

- Signals control the access from one train segment to the next one.
- They are drawn in the direction of use, e.g. Signal sig2 is visible from s1 and controls access to s2.
- In the example sig2, sig7, sig9, control access to the set of points p1, and sig3, sig6, sig10 control access to p2.
- sig1, sig5 control access to s1, s5 respectively, and sig8, sig4 control access to the neighbouring rail yards.

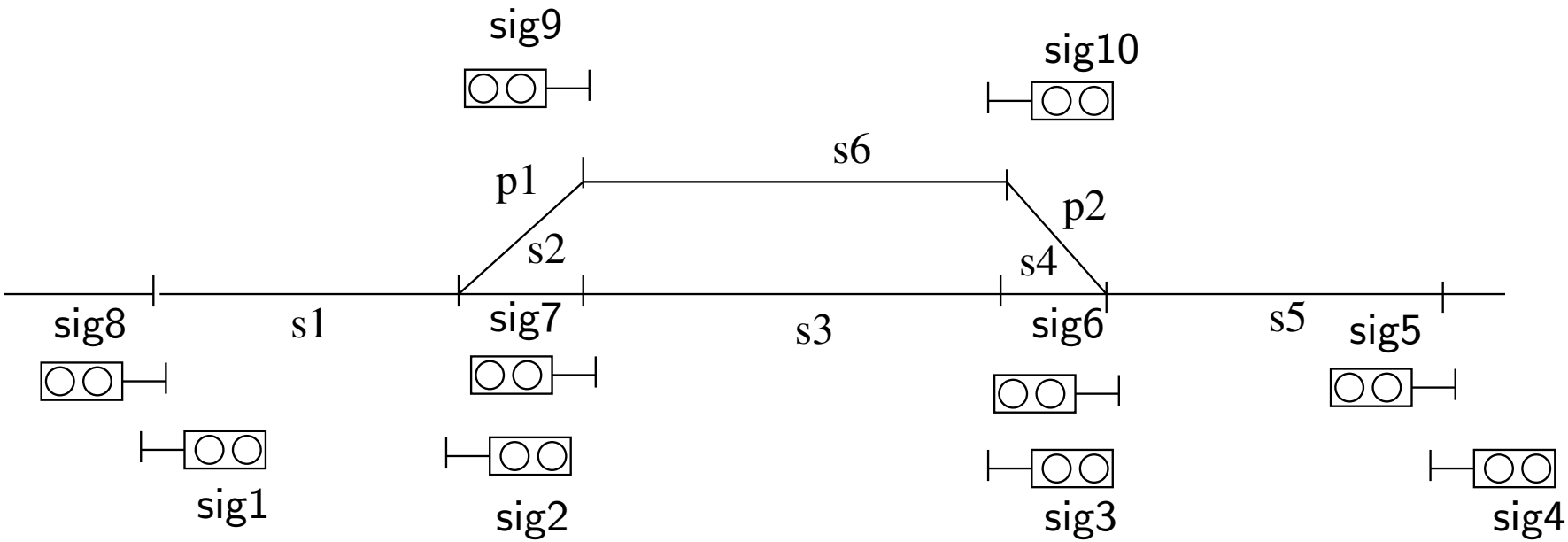


Train Routes

- The control system for such a rail yard has several **train routes**.
  - A **train route** is a sequence of track segments, the train can follow without ever having to stop in between (except in emergency cases).
  - The beginning of a train route and its end should be delimited by signals.
    - The first one prevents entering the train route, the second one, delimits access from this train route to the following train routes.
  - The segment before the guarding signal belongs to the route.

Train Routes

- So we have a train route (s1,s2,s6)
  - with segments s1,s2, s6
  - guarded by signal sig2
- Routes r1, r2 are connected if after having traversed route r1 one can proceed to route r2
  - route (s1,s2,s6) and route (s6,s4,s5) are connected.



Formalisation in Agda

- We follow
  Karim Kanso and Anton Setzer: *A light-weight integration of automated and interactive theorem proving*. Mathematical Structures in Computer Science, FirstView, 2014, pp. 1 - 25.

Formalisation

- We have sets and relations

$$
\begin{array}{lcl}
\text{Segment} & : & \text{Set} \\
\text{Train} & : & \text{Set} \\
\text{Route} & : & \text{Set} \\
\text{Connected} & : & \text{Route} \to \text{Route} \to \text{Set} \\
\text{SegInRoute} & : & \text{Segment} \to \text{Route} \to \text{Set}
\end{array}
$$

Model

- Time is given as

$$\mathrm{Time} = \mathbb{N} : \mathrm{Set}$$

- Depending on $t : \mathrm{Time}$ we assume

$$\begin{array}{lcl} \mathrm{trainRoute}_t & : & \mathrm{Train} \to \mathrm{Route} \\ \mathrm{signalAspect}_t & : & \mathrm{Route} \to \{\mathrm{proceed}, \mathrm{danger}\} \end{array}$$

Abstract Assumptions about Routes and Trains

- Single-Entry-Point: If two routes $route_1$ and $route_2$ are connected to route $route_3$, there is a segment (the one before the signal of $route_3$ which is in $route_1$ and $route_2$):

$$\begin{aligned}\forall route_1, route_2, route_3.&\text{Connected } route_1\ route_3\\ &\rightarrow\ \text{Connected } route_2\ route_3\\ &\rightarrow\ \exists segment.(\text{SegInRoute } segment\ route_1\\ &\qquad\wedge\text{SegInRoute } segment\ route_2)\end{aligned}$$

- Trains follow connected routes and obey signals:

$$\begin{aligned}\forall t, train.&(\text{trainRoute}_t\ train \equiv \text{trainRoute}_{t+1}\ train)\\ &\vee(\text{Connected } (\text{trainRoute}_t\ train)\ (\text{trainRoute}_{t+1}\ train)\\ &\wedge\text{signalAspect}_t(\text{trainRoute}_{t+1}\ train) \equiv \text{proceed})\end{aligned}$$

Abstract Signal Principle 1: Opposing Signals are not both Green

- If a segment is in two different routes, the signal of one of the routes must have aspect danger:

$$\begin{aligned}
&\forall t, \mathit{route}_1, \mathit{route}_2, \mathit{segment}.\\
&\quad \mathit{route}_1 \not\equiv \mathit{route}_2\\
&\quad \rightarrow \mathrm{SegInRoute}\ \mathit{segment}\ \mathit{route}_1\\
&\quad \rightarrow \mathrm{SegInRoute}\ \mathit{segment}\ \mathit{route}_2\\
&\quad \rightarrow (\mathrm{signalAspect}_t\ \mathit{route}_1 \equiv \mathit{danger}\\
&\qquad \vee \mathrm{signalAspect}_t\ \mathit{route}_2 \equiv \mathit{danger})
\end{aligned}$$

Abstract Signal Principle 2: Routes of Trains are Guarded

- If a train is using a route, all routes with access to the segments of this route are guarded by red signal:

$$\begin{aligned}&\forall t, \mathit{train}, \mathit{segment}, \mathit{route}.\\ &\quad \mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_t\ \mathit{train})\\ &\quad \rightarrow \mathrm{SegInRoute}\ \mathit{segment}\ \mathit{route}\\ &\quad \rightarrow \mathrm{signalAspect}_t\ \mathit{route} \equiv \mathrm{danger}\end{aligned}$$

Initial Condition

$$\forall \mathit{train}_1, \mathit{train}_2, \mathit{segment}$$
$$\mathit{train}_1 \not\equiv \mathit{train}_2$$
$$\rightarrow \neg(\mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_0\ \mathit{train}_1)$$
$$\wedge \mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_0\ \mathit{train}_2))$$

Collision Free

Theorem

Assume the above abstract conditions. Then trains don't collide, i.e.

$$\begin{aligned}&\forall t, \mathit{train}_1, \mathit{train}_2, \mathit{segment}\\ &\quad \mathit{train}_1 \not\equiv \mathit{train}_2\\ &\quad \rightarrow \neg(\mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_t\ \mathit{train}_1)\\ &\qquad \wedge \mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_t\ \mathit{train}_2))\end{aligned}$$

Proof of Theorem

- Induction on $t : \mathrm{Time}$.
- $t = 0$ follows by the initial condition.
- For $t \to t + 1$ assume $\mathit{train}_1, \mathit{train}_2, \mathit{segment}$ s.t.

$$\begin{aligned} &\mathit{train}_1 \neq \mathit{train}_2 \\ &\mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_{t+1}\ \mathit{train}_1) \\ &\mathrm{SegInRoute}\ \mathit{segment}\ (\mathrm{trainRoute}_{t+1}\ \mathit{train}_2) \end{aligned}$$

  and show a contradiction.
  - If none of the trains have moved (so their routes are as before) this follows by IH.

Proof of Theorem

- If only $train_1$ has moved we have:

$\mathrm{signalAspect}_t(\mathrm{trainRoute}_{t+1}\ train_1) \equiv proceed$
(since $train_1$ obeys signals)
$\mathrm{SegInRoute}\ segment\ (\mathrm{trainRoute}_{t+1}\ train_2)$
$\mathrm{SegInRoute}\ segment\ (\mathrm{trainRoute}_t\ train_2)$
(by $\mathrm{trainRoute}_{t+1}\ train_2 = \mathrm{trainRoute}_t\ train_2$)
$\mathrm{SegInRoute}\ segment\ (\mathrm{trainRoute}_{t+1}\ train_1)$
$\mathrm{signalAspect}_t\ (\mathrm{trainRoute}_{t+1}\ train_1) \equiv \mathrm{danger}$
(by Abstract Signal Principle 2)
Contradiction

- The case where only $train_2$ has moved follows similarly.

Proof of Theorem

- If both $train_1$, $train_2$ have moved to the same route we have

$\text{trainRoute}_{t+1}\ train_1 \equiv \text{trainRoute}_{t+1}\ train_2$
$\text{Connected}\ (\text{trainRoute}_t\ train_1)\ (\text{trainRoute}_{t+1}\ train_1)$
$\text{Connected}\ (\text{trainRoute}_t\ train_2)\ (\text{trainRoute}_{t+1}\ train_2)$
Since train routes of trains are connected
$\exists segment$
$\text{SegInRoute}\ segment\ (\text{trainRoute}_t\ train_1)$
$\wedge \text{SegInRoute}\ segment\ (\text{trainRoute}_t\ train_2)$
(by single entry to routes)
Contradiction to IH

Proof of Theorem

- If both $train_1$, $train_2$ have moved to different routes we have

$$\begin{aligned}
&\text{signalAspect}_t\ (\text{trainRoute}_{t+1}\ train_1) \equiv \text{proceed}\\
&\text{signalAspect}_t\ (\text{trainRoute}_{t+1}\ train_2) \equiv \text{proceed}\\
&\quad \text{(Since trains obey signals)}\\
&\text{trainRoute}_{t+1}\ train_1 \not\equiv \text{trainRoute}_{t+1}\ train_2\\
&\text{SegInRoute}\ segment\ (\text{trainRoute}_{t+1}\ train_1)\\
&\text{SegInRoute}\ segment\ (\text{trainRoute}_{t+1}\ train_2)\\
&(\text{signalAspect}_t\ (\text{trainRoute}_{t+1}\ train_1) \equiv \text{danger}\\
&\ \vee \text{signalAspect}_t\ (\text{trainRoute}_{t+1}\ train_2) \equiv \text{danger})\\
&\text{Contradiction}
\end{aligned}$$

Points in Routes of Trains are Locked

- Similarly we were able to show that under additional conditions on points we have
  If a set of points is in facing direction of a route of a train, then the set of points is locked.

Sketch of Reduction to Real Interlockings

- The conditions on $\mathrm{trainRoute}_t$ and $\mathrm{signalAspect}_t$ are still abstract.
- In order to reduce it to concrete interlockings we take the following steps
  - Formalise state (consisting of interlocking state, location circuits, trains).
  - Formalise desired inputs to state.
  - Define initial state.
  - Define functions computing next state depending on state and desired input.
  - Define concrete signalling principles and conditions on locations/trains for initial state and next state.
  - Show that the functions above fulfil these concrete conditions.
  - Compute $\mathrm{trainRoute}_t$, $\mathrm{signalAspect}_t$.
  - Show that concrete conditions above imply the abstract conditions on $\mathrm{trainRoute}_t$, $\mathrm{signalAspect}_t$.
  - Therefore the interlocking system is safe.

Evaluation

- Even in this simplified situation it is rather complicated to see that the signalling principles imply safety.
- In usual validation this is done by hand.
- In the above approach we have formalised it mathematically in Agda and shown that the signalling principles imply safety.
- Therefore the validation gap has been narrowed.

Conclusion

- **Validation gap** between Specification and Requirements.
- By having a **requirements specification** which is as close as possible to the requirements this gap can be narrowed.
- Replaces arguments which are carried out informally in the head of the validator by robust mathematical arguments.
- Two step verification:
  - **Step 1**: Program fulfils program specification.
  - **Step 2**: Program specification implies requirements specification.
- Full verification of real world interlocking system in Agda has been carried out (PhD thesis Karim Kanso).
  Interlocking system could be executed in Agda as an interactive program.

References

- Karim Kanso and Anton Setzer: *A light-weight integration of automated and interactive theorem proving*. Mathematical Structures in Computer Science, FirstView, 2014, pp. 1 - 25.
- Karim Kanso: Agda as a Platform for the Development of Verified Railway Interlocking Systems. PhD thesis, Department of Computer Science, Swansea University, Swansea, UK. http://www.swan.ac.uk/∼csetzer/articlesFromOthers/index.html
- Karim Kanso: Formal Verification of Ladder Logic. MRes thesis, Department of Computer Science, Swansea University, Swansea, UK. http://www.swan.ac.uk/∼csetzer/articlesFromOthers/index.html