Sicherheitsanalyse eines Bahnhofstellwerkes

Die Planung von Stellwerken für große Bahnhöfe ist eine sehr komplexe Aufgabe. Viele verschiedene Fahrstraßen müssen auf einem ausgedehnten Verkehrsnetz geplant werden. Darüber hinaus sind solche Komponenten, wie z. B. die Stellwerke und das entsprechende Streckennetz, sicherheitskritische Infrastrukturen. Das bedeutet, dass Fehler zu kostspieligen und gefährlichen Gefahren führen können. Die Verifizierung solcher Stellwerksysteme durch die Anwendung formaler Verifikationstechniken (z. B. Model Checking) erhöht die Qualität erheblich. Trotzdem sind Techniken wie Model-Checking in der Praxis nicht üblich.

Wir sind der Meinung, dass ein Hauptproblem darin besteht, dass es zu diesem Problem keine Standardsoftware gibt, die als einfaches Add-on in bestehende Planungstools eingebunden werden kann. Insbesondere gibt es mehrere Tools, die ihre domänenspezifische Sprache oder eine Schnittstelle zu einem bestimmten Modellierungstool bereitstellen, das die Verifikationsaufgabe übernehmen kann. Allerdings müssen Sie entweder das eine Tool finden, das für Ihre spezifische Modellierungsumgebung passt, oder Sie müssen Ihr Modell in die Eingabesprache Ihres Verifikationswerkzeugs transformieren. Um dieses Problem zu lösen, wollen wir eine strukturelle Transformation von einem akzeptierten Gleisnetzbeschreibungsstandard zu einer formalen Darstellung eines Stellwerksystems und seiner Fahrstraßentabelle definieren.

In den letzten zehn Jahren wurde von einem Konsortium ein Datenschemastandard entwickelt, das die Austauschbarkeit von Eisenbahndaten unterstützen soll. Es heißt Railway Markup Language (railML) [1] und definiert ein auf XML basierendes Datenschema. Durch die Verwendung dieses Standards wird es möglich, einen Verifikationsansatz für allgemeine Stellwerke und Fahrstraßentabellen zu definieren. Da wir davon ausgehen, dass dieser Standard in den nächsten Jahren industriell angewendet werden wird, kann ein darauf aufbauendes Verifikationstool dazu beitragen, die Hürde für den Einsatz formaler Verifikationstechniken für Stellwerke in der Praxis zu verringern.

Konkret wird bei unserem Ansatz, wie in Abb. 1 gezeigt, bei Bedarf ein analoger Gleisplan eingescannt und in ein railML-Modell überführt. Anschließend wird das railML-Modell zusammen mit dem Verschlussplan unter Zuhilfenahme vormodellierter Standardkomponenten zu einem formalen Automatenmodell zusammengefügt wird. Dieses wird anhand der gegebenen Spezifikation (Beschreibung von Fehlerzuständen) verifiziert.

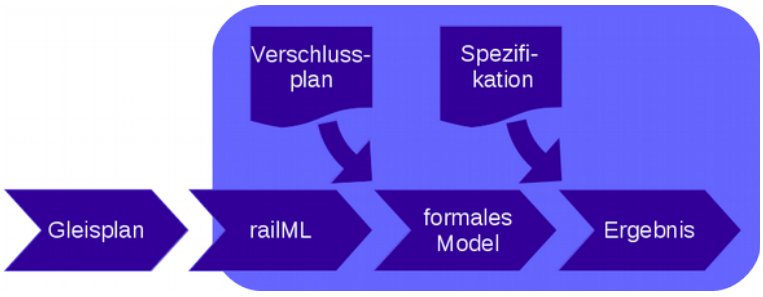

Abbildung 1: Verifikationsprozess

Railway Markup Language - railML

Wenn wir die Daten eines gegebenen Stellwerkes in mehreren Schritten mit verschiedenen Werkzeugen verarbeiten wollen, müssen wir unsere Modelltransformationen auf ein Standardschema stützen. Wir konzentrieren uns daher auf die Infrastrukturdefinition von railML.

Die Definition von railML im Allgemeinen enthält Schemata für die Fahrplanerstellung, Infrastrukturdefinitionen, Wagen sowie Loks und Stellwerke sowie Signalpläne. Interessant wäre auch

1 https://wiki.railml.org/index.php?title=Main_Page

```
<railml>
 <infrastruxcture id='example'>
  <tracks>
   <track id='T12'>
    <trackTopology>
     <trackBegin id='T12a' pos='0'>
      <connection id='T12ac' ref='T02bc'/>
     </trackBegin>
     <trackEnd id='T12b' pos='42'>
      <bufferStop id='T12bc'/>
     </trackEnd>
    </trackTopology>
    <ocsElements>
     <signals>
      <signal id='SigC' pos='10' dir='down'
                                type='main'/>
     </signals>
    </ocsElements>
   </track>
   <track id='T02'>
    <trackTopology>
     <connections>
      <switch id='P02' pos='21'>
       <connection id ='P02c' ref='P01c'
                  orientation='incomming'/>
      </switch>
     </connections>
     <trackBegin id='T02a' pos='0'>
      <bufferstop id='T01ac'/>
     </trackBegin>
     <trackEnd id='T02b' pos='42'>
       . . .
     </trackEnd>
    </trackTopology>
   </track>
```

```
   <track id='T01'>
    <trackTopology>
     <connections>
      <switch id='P01' pos='21'>
       <connection id ='P01c' ref='P02c'
                    orientation='outgoing'/>
      </switch>
     </connections>
     <trackBegin id='T01a' pos='0'>
      <connection id='T01ac' ref='T21bc'/>
     </trackBegin>
     <trackEnd id='T01b'
                     pos='42'>...</trackEnd>
    </trackTopology>
   </track>
  </tracks>
 </infrastructure>
</railml>
```

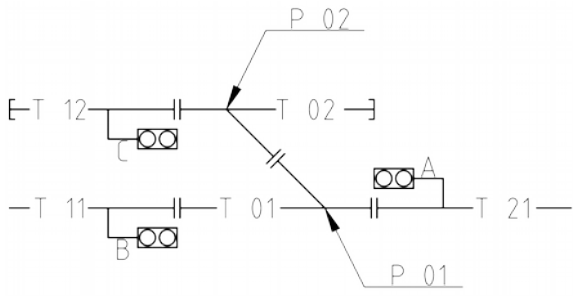

Abbildung 2: Gleisplan eines Beispielbahnhofes und ein Auszug aus seiner railML-Repräsentation

die Verwendung des *Interlocking*-Schemas gewesen, das die Definition von Verschluss- und Fahrstraßentabellen unterstützen soll. Leider befindet sich dieses System derzeit in der Entwicklung und ist daher nicht verfügbar. Stattdessen verwenden wir eine einfache csv-Darstellung der Verschlusspläne.

In Abb. 2 zeigen wir Ihnen einen kleinen Auszug aus der Infrastruktur railmML-Darstellung eines Beispiels. Im Folgenden möchten wir eine kurze Erläuterung dazugeben, ohne eine vollständige Beschreibung von railML zu präsentieren.

Die Grundelemente der Infrastruktur sind Gleise mit einer *ID*. Für jedes Gleiselement definieren wir sein Beginn (*trackBegin)*, sein Ende (*trackEnd*) und die Verbindungen zu anderen Gleiselementen bzw. einen Prellbock (*bufferStop*) oder einen Übergang zu anderen Systemen (*openEnd*). Jedes *trackBegin* und *trackEnd* hat verpflichtend eine Position (*pos*), an der die Richtung des Gleises definiert wird. Von der Position mit dem niedrigeren Wert bis zum höheren Wert ist die Richtung aufwärts, ansonsten abwärts. Die Verbindungen werden durch Verweise auf andere *trackBegin*- oder *trackEnd*-Elemente über deren eindeutige *ID* realisiert.

Weiterhin finden wir Weichenelemente innerhalb eines Gleises, die mit entsprechenden Abzweigen oder Verbindungen auf anderen Gleisen, auch über die *ID* der Elemente, verbunden werden können.

Signale werden als Unterelement von Gleisen definiert. Für sie wird Position (*pos*) und Richtung (*direction*) beschrieben. Zu Signalen kann außerdem ihre Funktion gespeichert werden. Wir beschränken uns in der Modellierung dabei auf Hauptsignale.

Mit dieser Darstellung erhalten wir einen nutzbaren Standard für den Austausch von Stellwerksdaten zwischen verschiedenen Modellierungs- und Verifikationswerkzeugen.

Die gesamte railML-Definition eines Gleisfeldes enthält natürlich mehr als nur diese Elemente, aber sie sollen an dieser Stelle ausreichen, um unser Verfahren vorzustellen.

Model-Checking

Das Model-Checking benötigt als Eingabe eine Beschreibung des Systems und eine Beschreibung von Systemzuständen, nach denen gesucht wird. Dabei beinhaltet die Systembeschreibung – das Modell – alle möglichen Systemzustände und deren Übergänge untereinander, während die Liste an Zuständen – die Spezifikation – eine Beschreibung von Fehlerzuständen enthält. Im Rahmen des Model-Checkings wird dann im Zustandsgraphen des Modells nach den beschriebenen Fehlerzuständen gesucht und geprüft, inwiefern sie vom Initialzustand aus erreichbar sind.

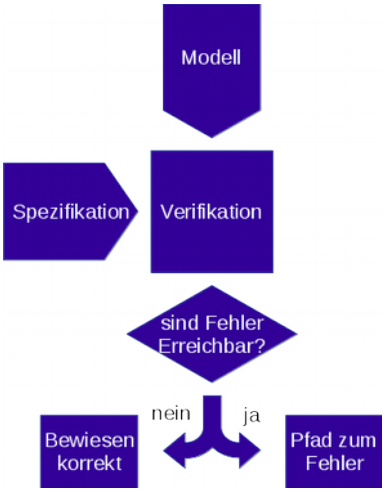

Abbildung 3: Ablauf des Model-Checkings

Wer sich mit dieser Thematik tief greifender beschäftigen möchte, dem seien an dieser Stelle die Bücher „Model Checking“[2] von Edmund M. Clarke, Orna Grumberg und Doron Peled sowie „Principlesof model checking“[3] von Christel Baier, Joost-Pieter Katoen und Kim G. Larsen empfohlen.

Ein Template-System zur Modellierung von Stellwerken

Die Grundidee unseres Transformationssystems ist es, eine Reihe von gängigen Templates in einer formalen Sprache, wie z. B. SAML[4] in unserem Prototyp, für einzelne Gleiselemente zur Verfügung zu stellen und die Instanziierungen und Verknüpfungen zwischen den Elementen aus den railML-Darstellungen abzuleiten.

Alle Infrastrukturelemente (Gleise, Signale, Weichen) werden über Fahrstraßen gesteuert. Jede Fahrstraße enthält Informationen über die enthaltenen Gleisabschnitte und die entsprechenden Signalzustände und Weichenstellungen. Daher sind alle Abläufe mit den aktuell aktiven Fahrstraßen korreliert, z. B. welche Gleise belegt sind, welches Signal gestellt werden muss oder welche Weiche auf eine bestimmte Position gestellt werden muss.

2 Model Checking; Edmund M. Clarke, Orna Grumberg und Doron Peled; MIT Press; 1999; ISBN 9780262032704

3 Principles of model checking; Christel Baier, Joost-Pieter Katoen und Kim G. Larsen; MIT Press; 2008; ISBN 9780262026499

4 https://cse.cs.ovgu.de/vecs/index.php/techniques/language

Routenplanung

Das Schalten der Fahrstraßen ist indeterministisch realisiert, so kann z. B. jede nicht aktive Fahrstraße jederzeit angefordert werden. Wenn mehr als eine Fahrstraße angefordert und deren Fahrtwege frei sind, lösen wir diesen , indem wir die Fahrstraße mit der niedrigeren *ID* wählen (z. B. Route 1 vor Route 2). Insgesamt können mehrere Fahrstraßen gleichzeitig aktiv sein, aber nur eine weitere Fahrstraße kann in jedem Zeitschritt aktiv werden.

Logische Repräsentation von Zügen

Die modellierten Züge bewegen sich deterministisch. Das bedeutet, dass ein Zug in jedem Zeitschritt ein neues Gleis belegt und eines verlässt (ausgenommen er steht vor einem *Halt* zeigenden Signal). Dabei vernachlässigen wir physikalische Parameter wie Zuggeschwindigkeit und Länge des Gleisabschnitts. Bei Bedarf kann das formale Modell leicht um physikalisches Verhalten erweitert werden. Darüber hinaus definieren wir zwei Zug-IDs für die Modellierung von Kollisionen, Flankenschutz und anderen Sicherheitseigenschaften nötig sind.

Im Folgenden stellen wir Ihnen die Automatenmodelle der Templates und deren grundlegendes Verhalten vor.

Gleise

Im Allgemeinen kann ein Gleis *Frei*, *Reserviert* oder *Besetzt* sein. Da wir das System mit zwei Zug-IDs verifizieren, kann das Gleis mit dem Zug Nr. 1 (*occupiedBy1*) oder dem Zug Nr. 2 (*occupiedBy2*) belegt werden. Dabei ist es möglich beliebig viele Züge im modellierten Stellwerk fahren zu lassen, auch wenn diese unter Umständen die gleiche ID haben. Das Model-Checking, in Kombination mit nichtdeterministischem Verhalten, deckt alle möglichen Kombinationen von Zugpositionen und Strecken ab, sodass zwei Zug-IDs ausreichen, um alle möglichen Zug-zu-Zug-Situationen zu analysieren.

RailML-Relation - Im Folgenden benötigen wir Informationen über benachbarte Gleise. Diese Informationen werden aus dem railML-Element *Track* (vgl. Abb. 2) abgeleitet. Um zu erkennen, welche Gleise vor und hinter einem Gleiselement liegen, verwenden wir die gängige Richtungskonvention (aufsteigende Streckenkilometrierung). In railML somit vom unteren *pos*-Wert zum Oberen. Von hier sind Gleise, auf die *trackBegin* referenziert, vorhergehende Gleise, und die Referenzen *trackEnd* zeigt auf nachfolgende Gleise. Die Zuordnung von Gleisen zu Fahrstraßen und deren gewünschte Richtung (*up* oder *down*) wird aus dem Verschlussplan extrahiert.

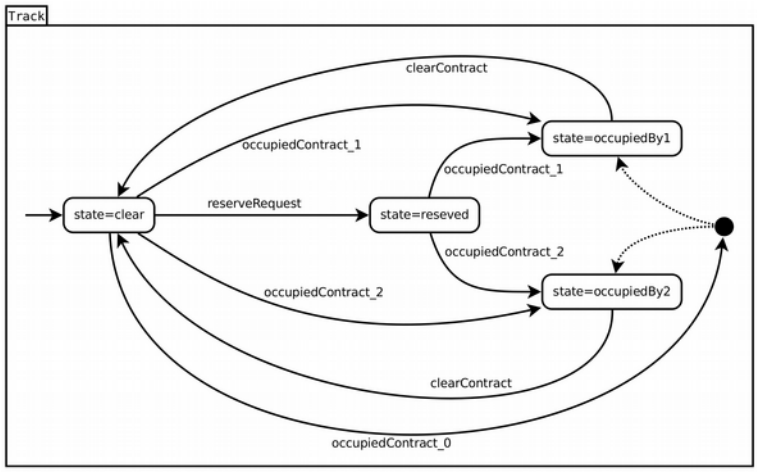

Abbildung 4: vereinfachtes Automatenmodell der Gleiselemente

Verhalten – Das Verhalten der Gleie ist vereinfacht in Abb. 4 gezeigt. Ein Gleis kann über eine Fahrstraße angefordert werden. Dies geschieht indirekt, indem der aktuelle Zustand aller Fahrstraßen, die das Gleis enthalten, überprüft werden. Der Ausgangszustand des Modells ist mit leeren Gleisen, sodass es eine Möglichkeit geben muss, Züge auf den Gleisen zu platzieren. Daher haben alle offenen Gleisenden, also Gleise über die Züge in den Stellwerksbereich

ein- oder ausfahren können, die Möglichkeit, Züge zu generieren. Der Übergang von *Besetzt* nach *Frei* wird wird ausgelöst, wenn das nächste Gleis besetzt und das Vorhergehende Frei ist. Dadurch wird sichergestellt, dass ein Zug nicht virtuell geteilt oder entfernt wird.

Signale

Die Signale können sich in einem von zwei Zuständen befinden - *Stopp* (rot / Hp0) oder *Weiterfahren* (grün / Hp1). Zustände, die es den Zügen erlauben, mit niedriger Geschwindigkeit zu fahren, werden nicht modelliert.

RailML-Relation - Für die Signale müssen wir wissen, an welchem Gleis das Signal platziert ist und in welche Richtung es zeigt, sowie das nächste Gleis hinter dem Signal, um zu definieren, wann der Zug in eine Fahrstraße eingefahren ist und wann das Signal zurückgesetzt werden kann. Diese Informationen lassen sich aus dem Signalelement der railML-Beschreibung (vgl. Abb. 2) und dem übergeordneten Gleiselement ableiten.

Verhalten - Das vollständige Automatenmodell des Signal-Templates ist in Abb. 5 dargestellt. Sein Ausgangszustand ist *Halt*. Er ändert sich zu *weiterfahren*, wenn eine von hier ausgehende Fahrstraße gesichert ist. Wenn ein Zug das Signal passiert hat, fällt er auf *Halt* zurück. Daher ist ein Signal mit dem dahinter liegenden Gleis verbunden, um einen durchgefahrenen Zug zu erkennen.

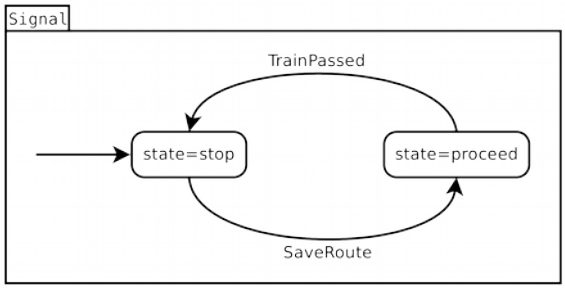

Abbildung 5: Automatenmodell der Signale

Weichen

Eine Weiche kann sich je nach seiner Stellung in zwei Zuständen befinden: positiv (geradeaus / *normal*) und negativ (abzweigend / *reverse*).

RailML-Relation - Neben der *ID* müssen wir die Informationen der Verbindungsbeziehungen aus der Weiche ableiten, z. B. welche Gleise über gerade und abzweigende Stellung miteinander verbunden sind. Dies geschieht durch die Verwendung der des Zustandes der Weiche (vgl. Abb. 6) und der Verweise auf das benachbarte Gleis bzw. die benachbarte Weiche in Kombination mit dem übergeordneten Gleiselement.

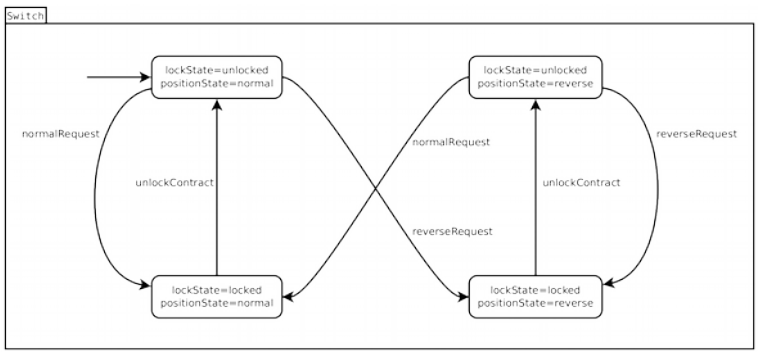

Abbildung 6: Automatenmodell der Weichen

Verhalten - Das Automatenmodell der Weichen hat vier Zustände, zwei für jede Position (gerade oder abzweigend), und zwei zur Verriegelung (verriegelt oder entriegelt). Durch Fahrstraßen werden Weichenpositionen angefordert. Dabei verriegelt die Weiche. Hat ein Zug die Weiche passiert entriegelt sie wieder und ist bereit für neue Stellbefehle.

Fahrstraßen

RailML-Relation - Wie bereits erwähnt, ist das railML Interlocking-Schema noch nicht fertig, bzw. nicht verfügbar. Deshalb werden an dieser Stelle csv-Daten verarbeitet.

Verhalten - Eine Fahrstraße steuert und überwacht alle Gleisseitenelemente, die für eine sichere Zugfahrt notwendig sind. Sie kann sich in einem von drei Zuständen befinden (*frei*, *angewiesen* oder *belegt*). Eine Fahrstraße wird als *frei* initialisiert. Wie bereits erwähnt, werden die Fahrstraßen nicht deterministisch angewiesen. Wenn eine Fahrstraße belegt werden kann, z. B. im *angewiesen* Zustand, bewirkt sie, dass die entsprechenden Gleise für diese Fahrstraße reserviert werden und die Weichen ihre Position nach Bedarf ändern. Die Fahrstraße überprüft den Zustand der Elemente (*tracksReserved* und *switchesLocked*). Danach wird die Route belegt. Wenn der Zug die gesamte Strecke passiert hat, wechselt die Fahrstraße wieder in den Grundzustand.

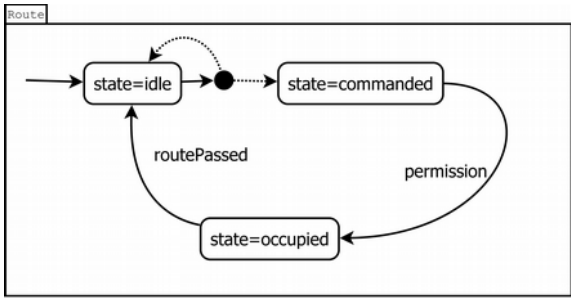

Abbildung 7: Automatenmodell der Fahrstraßen

Sicherheitsspezifikationen

Im Folgenden möchten wir Ihnen die Sicherheits-Spezifikationen vorstellen, die wir für unsere Methode zur Verfügung stellen. Selbstverständlich sind Erweiterungen um weiterführende Sicherheitsanalysen denkbar.

Kollisionserkennung

Zwei Züge kollidieren, wenn sie sich zur gleichen Zeit am gleichen Ort befinden. Um Kollisionen zu erkennen, müssen die Gleise erkennen, ob es im nächsten Schritt zu einer Kollision kommen kann. Dies geschieht durch Beobachtung der mit *Belegt* markierten Gleiselemente. Wenn ein besetztes Gleis im nächsten Schritt durch einen Zug anderer ID besetzt werden soll, haben wir eine mögliche Kollision gefunden.

Entgleisungsdetektion

Wir beschränken uns auf die Entgleisung auf Weichen. Die beiden Fälle, die auftreten können, sind i) Entgleisung, weil ein Zug über eine Weiche fährt, obwohl diese unverriegelt ist, und ii) Entgleisung, weil ein Zug eine Weiche passiert, die sich in einer falschen Position befindet.

Die Spezifikation legt fest, dass die Position einer Weiche mit dem belegten Gleisobjekt korrelieren muss, sowie dass die Weiche verriegelt sein muss. Ansonsten wird ein Fehler erkannt.

Flankenschutzprüfung

Vor allem auf Abstellgleisen können einige abgestellte Waggons unbedacht ins Rollen kommen. Um Unfälle zwischen diesen Wagen und Zügen zu vermeiden, können bestimmte Weichen, die sich nicht auf der Strecke des Zuges befinden, in eine Position gebracht werden, sodass sie die Wagen auf das andere Gleis als das vom Zug benutzte bringen. Ähnliche Probleme werden durch durchrutschende Züge verursacht. Also Züge, die an einem Halt zeigenden Signal nicht halten.

Zur Überprüfung dieser Sicherheitsrichtlinie wird jedes Mal, wenn ein Zug über eine Weiche fährt, eine Rückwärtssuche durchgeführt. Diese Suche beginnt an dem nicht benutzten Zweig der Weiche

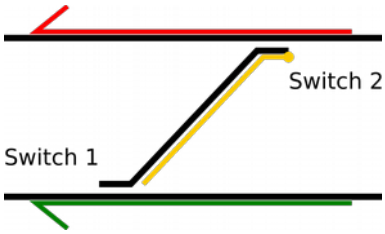

Abbildung 8: Flankenschutz sichergestellt - Die Suche (gelb) endet an der abweisend gestellten Weiche. Ein rollender Waggon (rot) wird von der Flanke des Zuges (grün) abgehalten.

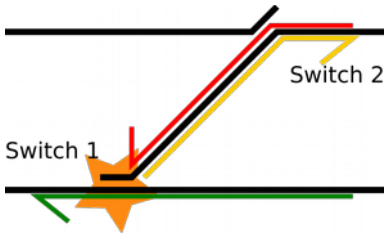

Abbildung 9: Flankenschutz nicht sichergestellt - Die Suche (gelb) führt auf ein Bahnhofsgleis. Ein rollender Waggon (rot) kann in die Flanke des Zuges (grün) fahren.

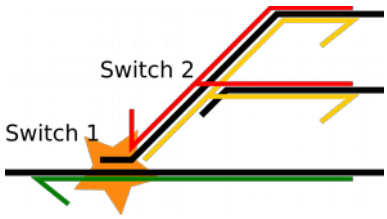

Abbildung 10: Flankenschutz nicht möglich - Die Suche (gelb) wird an beiden Enden der Weiche fortgeführt, da ein Waggon (rot) von beiden Strängen in die Flanke des Zuges (grün) fahren kann.

und verläuft entlang der Gleise. Erreicht sie eine Weiche, die in der Lage ist, die Flanke des Zuges zu schützen (die Wagen abzulenken) stoppt die Suche (Abb. 8). Wenn sich die Weiche in der anderen Position befindet, wird die Suche fortgesetzt (Abb. 9). Wenn die Suche eine zusammenführende Weiche erreicht, wird sie auf beiden Zweigen fortgesetzt (Abb. 10). Die Spezifikation schreibt vor, dass die Rückwärtssuche niemals einen Bahnhof oder ein Hauptgleis erreicht.

Skalierbarkeit des Verfahrens

Um die Skalierbarkeit, also den Umgang mit Problemen unterschiedlicher Größe, bei unserem Lösungsansatz auszutesten, haben wir drei Bahnhöfe unterschiedlicher Ausdehnung untersucht.

Minimal-Beispiel – Als kleinsten Bahnhof haben wir ein fiktives Minimal-Beispiel verwendet, an dem sich gut die Grundprinzipien von Sicherungssystemen in Stellwerken erklären lassen. Es besteht lediglich aus einer durchgehenden Strecke und einem Abstellgleis, dabei können vier Fahrstraßen geschaltet werden.

Braine l'Alleud – Als Bahnhof mittlerer Größe wählten wir den Bahnhof Braine l'Alleud in Belgien. Er durfte schon bei anderen wissenschaftlichen Analysen von Stellwerken als Erpro-bungsträger dienen. In dem viergleisigen Bahnhof an einer zweigleisigen Hauptstrecke können 32 verschiedene Fahrstraßen eingestellt werden.

Horb – Wir nahmen die Horber Schienentage als Anlass, uns in der Größe der Gleispläne weiter zu entwickeln. Der Bahnhof Horb mit sechs Gleisen sowie drei vorgelagerten Gleisen bietet die Möglichkeit, Züge auf nicht weniger als 48 Fahrstraßen zu kontrollieren.

Im Vergleich (Abb. 11) zeigt sich, dass sehr einfache Netzpläne noch in Sekunden schnelle verifiziert werden können. Bei größeren Bahnhöfen kann die Verifikations-

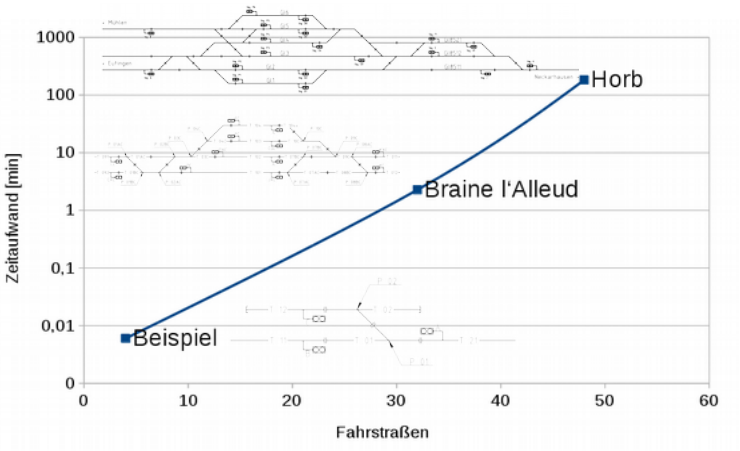

Abbildung 11: Zeitaufwand des Model-Checkings in Abhängigkeit von der Anzahl der Fahrstraßen

zeit jedoch im Stundenbereich liegen. Insgesamt zeigt sich ein für das Model-Checking übliches Verhältnis.

[Seitenränder normal]

[Ende Text:]

Ludwig Bedau / Tim Gonschorek / Frank Ortmeier
Otto-von-Guericke Universität Magdeburg – Fakultät für Informatik - Lehrstuhl für Software Engeeniering

Master Student / Doktorand / Lehrstuhlleiter