Towards a Framework for Modelling and Verification of Relay Interlocking Systems

Haxthausen, Anne Elisabeth

Published in:
Modeling, Development and Verification of Adaptive Systems

Publication date:
2010

Towards a Framework for Modelling and Verification of Relay Interlocking Systems

Anne E. Haxthausen

DTU Informatics, Technical University of Denmark, DK-2800 Lyngby, Denmark
ah@imm.dtu.dk

Abstract. This paper describes a framework currently under development for modelling, simulation, and verification of relay interlocking systems as used by the Danish railways. The framework is centred around a domain-specific language (DSL) for describing such systems, and provides (1) a graphical editor for creating DSL descriptions, (2) a validator for checking that DSL descriptions are statically well-formed, (3) a graphical simulator for simulating the dynamic behaviour of relay interlocking systems, and (4) verification support for deriving and verifying safety properties of relay interlocking systems. The paper also touches upon how such a framework can be developed using the RAISE Formal Method.

1 Introduction

In this paper we describe the visions of an ongoing project made in collaboration with Banedanmark.

Background and motivation A conventional means of keeping the railway traffic safe is to use interlocking systems that control signals and points in such a way that trains are only allowed to pass a signal when this cannot lead to train collisions or derailments. Many Danish interlocking systems are still implemented using complex electrical circuits containing relays. These relay based interlocking systems are documented by diagrams of the electrical circuits, and currently the only way to analyse them is to inspect the diagrams and manually draw conclusions. This is very difficult to do as the number of diagrams for a single system is very high and the logic described in each of them is complicated with many mutual dependencies. Certainly such a manual analysis is not only difficult and time consuming, but may also be error prone. This is not satisfactory for a safety-critical system. To help this, we started a project, the goal of which is to formalise and automate the validation and verification process for relay interlocking systems.

Solution approach Our solution to the above mentioned problem is to provide a framework of computer-based tools that support the validation and verification process. The tools should be centred around a domain-specific language for expressing the documentation that is usually made for relay interlocking systems,

e.g. track layout and relay circuit diagrams. The idea is that to analyse or verify a relay interlocking system the railway engineer should express the documentation of the relay interlocking system in this domain-specific language, and the framework should provide tools that can be applied to such documentation to analyse and verify the documented relay interlocking system. Prototypes of such tools have already been developed. We have chosen to centre the tools around a domain-specific language rather than a general purpose modelling language, as it is easier for railway engineers to use a language that facilitates concepts already known and used in the railway domain.

Related work The author and Jan Peleska have developed a framework of tools for the construction and verification of tramway control systems [8]. These tools are also centred around a domain-specific language, however the language and tools are different from those described in this paper. The differences are due to the fact that the controllers in that work are electronic, while ours are implemented using relay circuits. For instance, the tool set described in this paper provides a simulator for the electrical behaviour of relay circuits (which is not relevant for electronic systems) while the other tool set provides a control software generator (which is not relevant for relay systems).

For other complementary and competing approaches for the development and verification of railway control systems the reader is e.g. referred to the contributions in [11], and for a survey of results and trends the reader is referred to the paper [2].

Paper overview First, in Section 2, we describe the railway application domain. Then, in Section 3, we give an overview of the tools framework, and in the subsequent sections we describe the domain-specific language and each of the tool components in more detail. Finally in section 8 we touch upon how such a framework can be formally developed using the RAISE Formal Method [10].

2 The railway application domain

In this section we introduce concepts of the railway domain that are relevant for this paper.

2.1 Equipment at a station

The considered interlocking systems use various track-side equipment to monitor and control trains:

Track circuits: The railway tracks are divided into sections each having equipment (a circuit) for train detection. The interlocking system uses this for monitoring the occupancy status of the individual track sections.

Points: Tracks are joined at points which can guide trains into different directions depending on the position of the point. An operator can switch the points by pushing some **buttons**. The interlocking system monitors and controls the positions of points.

Signals: Signals are placed at the entrance of some track sections. They can show GO and STOP aspects. The interlocking system sets the signals to inform the train drivers whether they are allowed to enter these sections.

Figure 1 illustrates the interactions between the operator, the interlocking system, the trains, and the track side equipment.

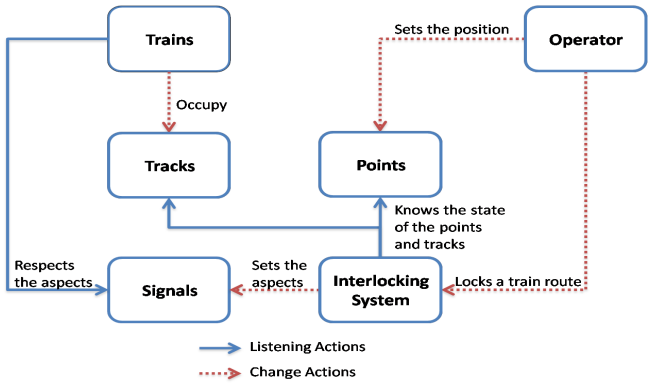

Fig. 1. Relationships between different elements of a station.

2.2 Train routes and train route tables

The stations we are considering in this paper use a *route based* approach to interlocking. The basic ideas of this approach are:

- Trains should drive on *routes* through the network.
- Each route is covered by an entrance signal that informs whether it is allowed for a train to enter the route or not. The trains must respect the signals.
- Two trains must never be allowed to drive on conflicting (i.e. overlapping) routes at the same time. *(To prevent collisions.)*
- Before a train is allowed to enter a route, the points must be locked in positions making the route connected (i.e. it is physically possible to go from one end of the route to the other end without derailing), and the route must be empty (i.e. there are no trains on the route). *(To prevent derailing and collisions, respectively.)*
- The points of a route must not be switched while a train is driving on the route. *(To prevent derailing.)*

For each station to be controlled by an interlocking system, a *train route table* is used to specify routes and interlocking rules for that station. Such tables define for each train route

– which settings of signals are required for the route to be *open* (i.e. for allowing trains to enter the route),
– which positions points must have for the route to be *connected*,
– which track sections must be unoccupied for the route to be *empty*, and
– the conditions for unlocking/freeing a route.

The tables also define which train routes are *conflicting*.

2.3 Relay circuits

The interlocking systems we are considering are implemented by electrical relay circuits. In [6] a formal domain-model for relay circuits is presented. Here we just give an informal description.

A relay circuit is made up of components such as power supplies, relays, contacts, and buttons, connected by wires. A *relay* is an electrical switch operated by an electromagnet to connect or disconnect a number of contacts in a circuit. When current goes through the relay, the magnet is *drawn* and some of the associated contacts are connected (these contacts are said to be *upper contacts*) while others (the *lower contacts*) are disconnected. When no current goes through the relay, the magnet is *dropped* and the associated upper and lower contacts will be disconnected and connected, respectively. When contacts are connected/disconnected this may imply that sub-circuits containing these contacts become live/dead. This again may imply that relays of these sub-circuits are drawn or dropped, and so on.

The system can get input from the environment:

– buttons can be pushed (and later released) by an operator
– for each track section there is a (track) relay that is dropped/drawn when a train enters/leaves that track section
– for each point there is a (point) relay that is dropped/drawn when that point is moved into a new position

2.4 Relay circuit diagrams

The Danish railways use diagrams to document the electrical circuits of a relay system.

For each internal relay one of the diagrams shows the sub-circuit that controls that relay. An example of such a diagram is shown in Figure 2. This diagram shows the sub-circuit controlling a relay named $RR1$. The circuit consists of a number of components connected by wires. The wires are depicted as black lines. At the top is the positive pole and at the bottom is the negative pole of the power supply. Relay $RR1$ is shown using this signature:

The downwards arrow informs that in the initial state this relay is dropped. (If it had been drawn the arrow would have been upwards.) A number of contacts

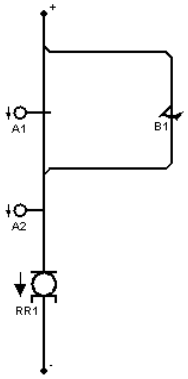

Fig. 2. Diagram for circuit controlling relay $RR1$.

belonging to other relays occur in this circuit. E.g. a contact belonging to a relay named $A1$ is shown using this signature:

The downwards arrow informs that in the initial state relay $A1$ is dropped. The horizontal bar breaks the wire – this indicates that the contact is disconnected in the initial state. If it had not been breaking the wire it would have indicated that the contact had been connected in the initial state. Also a button $B1$ is shown on the diagram using this signature:

A pushed button is shown by this signature:

3 Framework overview

As mentioned in the introduction our goal is to provide a framework of tools for analysing and verifying relay interlocking systems. In this section we give an overview of this framework.

To make the framework user friendly for railway engineers we exploit the idea to define and centre the tools around a *domain-specific language*, *DSL*, for expressing the documentation that is usually made for the relay interlocking system of a station. The tools we are developing comprise:

- a (graphical) *editor* for creating DSL descriptions
- a *validator* for checking that a DSL description follows static (structural) rules of the domain
- a (graphical) *simulator* that for a given DSL description can simulate the dynamic behaviour of the described interlocking system

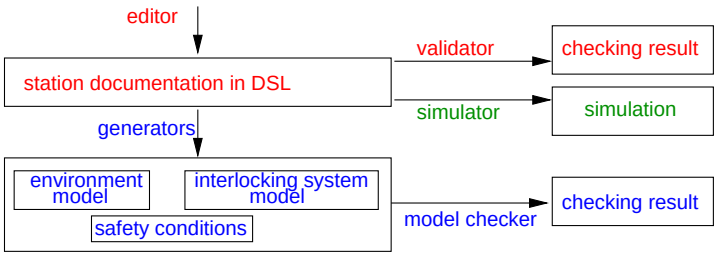

Fig. 3. Framework of tools.

– *generators* that from a DSL description produce input to a model checker:
  - a behavioural model of the described interlocking system
  - a behavioural model of the described environment (track isolations, points, and signals)
  - safety conditions

  A model checker can then be applied to this to verify that the interlocking system always satisfies the safety conditions.

Figure 3 illustrates this framework of tools. The language and tools will be further described in the next sections.

4 Domain-specific language

A *specification* D in *DSL* consists of the following station documentation:

– a *track layout diagram* describing the physical environment
– a *train route table* describing the interlocking rules
– *relay circuit diagrams* describing the physical implementation

In Figures 4 and 5 are shown the track layout diagram and the train route table for Stenstrup station (unfortunately in Danish). A detailed explanation of how to read such a table in given in [3]. The circuit diagrams for Stenstrup are too large to show in this paper.

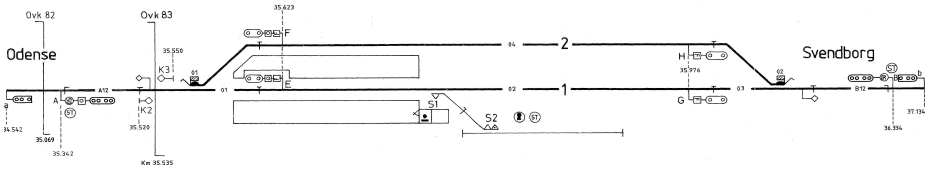

Fig. 4. Track layout for Stenstrup station.

A graphical editor for creating relay circuit diagrams and track layout diagrams has been implemented, see [5], while an editor for train route tables still has to be implemented.

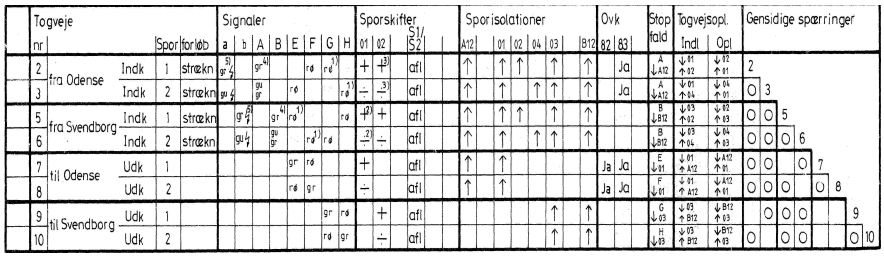

Fig. 5. Train route table for Stenstrup station.

5 Validation

The validator tool can be used to check that the station documents are statically well-formed, e.g. that

1. the track layout diagram represents a legal railway network of track elements
2. the circuit diagrams represent a legal network of circuits
3. the train route table
   (a) refers only to track elements in the track layout diagram
   (b) describes only routes that are connected paths in the railway network in the track layout diagram
   (c) marks overlapping routes as being conflicting
   (d) ...

6 Simulation

When relay diagrams for an interlocking system have been created by the editor and validated by the validator, the simulator can be used to visualise on these diagrams how the states of the relay circuits change over time. In this visualisation one can see the state of wires (current carrying or not), the state of relays (drawn or dropped), the state of contacts (connected or disconnected), and the state of buttons (pushed or released).

The user can give input to the system by playing:

1. an *operator* that pushes a button of the relay circuits
2. a *a train* that enters or leaves a track section shown in the track layout diagram

After having given an input the user can step through the sequence of states that the relay circuits will go through after such an input.

In Figure 6 is given an example of a simulation showing how the state of a circuit changes when a button is pushed. Wires that are current carrying are shown by a green colour (seen as a grey colour in black&white print). State 0 is the initial state. In the initial state, no wires are current carrying. When the button is pushed, current is going from plus to minus through relay 37, see state

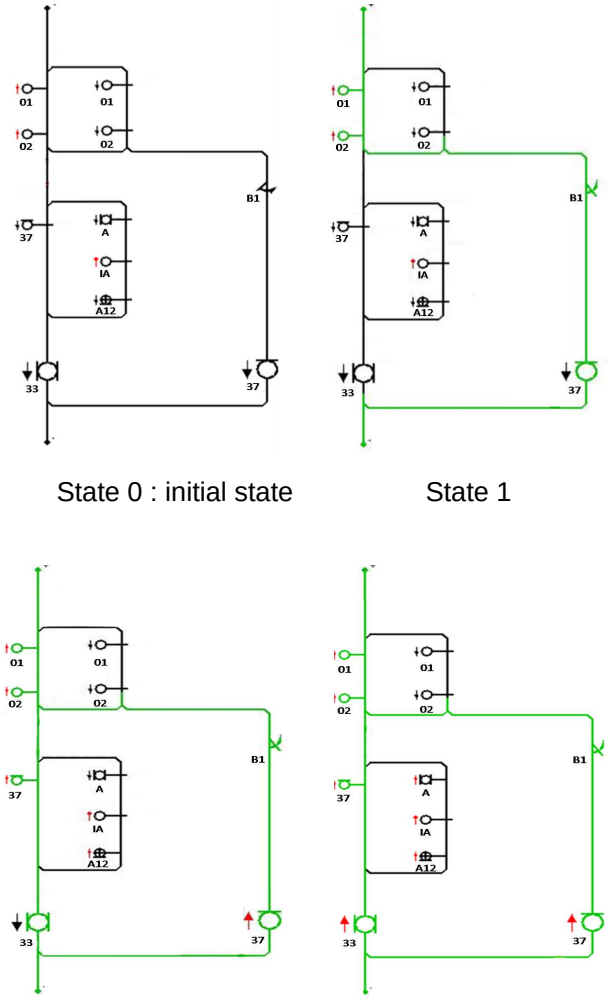

Fig. 6. A state sequence for a circuit.

1. As a consequence of this, relay 37 is drawn and its associated upper contact becomes connected, opening a second path of current from plus to minus through relay 33, see state 2. As current is going through relay 33, this will be drawn, see state 3. In state 3 no more internal events can happen.

A detailed description of the simulator tool and its development is given in [5].

7 Verification

This section describes how our framework provides verification support for a relay interlocking system.

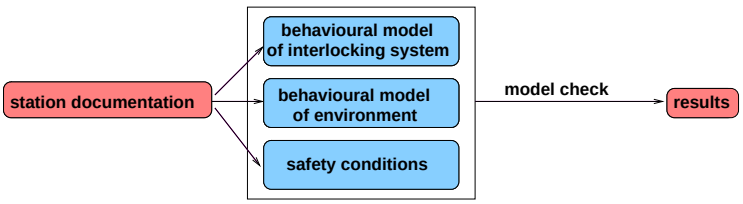

Fig. 7. Verification in two steps.

When the station documentation for the interlocking system has been created in the domain-specific language and validated by the validator, verification can be performed in two steps as illustrated in Figure 7. First generators are applied to automatically generate input to a model checker:

- a behavioural model M_c of the relay control system,
- a behavioural model M_e of the environment (signals, points, and track isolations), and
- safety conditions ϕ.

and then the model checker is applied to check that the concurrent composition of the models M_c and M_e always satisfies the safety conditions ϕ.

We have chosen *model checking* as the verification approach as this allows for full automation. As model checker tool we have chosen to use the SAL model checker [1]. The behavioural models are state transition system models and the safety conditions are assertions in the temporal logic LTL expressed in the SAL Language [4].

Details of this work is described in [7, 3].

8 Development of a domain-specific language and tools

We are using the RAISE Specification Language, RSL, [9] to specify the domain-specific language and most of the tools. Examples of this can be found in [7, 3]. As implementation language we are using Java.

The specifications are typically developed starting with a property-oriented specification and ending with an executable specification. Some of the advantages of this are:

- It is easier first to make an abstract specification in which e.g. only the properties of functions are given, and then later make an executable specification in which algorithms for the functions are given.
- It is easier to define data types and algorithms in an RSL executable specification and then translate these into Java, than coding directly in Java.

9 Conclusions

In this paper we have given an overview of a tool set that is intended to help railway engineers to analyse and verify relay interlocking systems. To describe a system to be analysed or verified, the railway engineer just has to use an editor for a domain-specific language to create documents that railway engineers are already used to create: track layout diagrams, train route tables, and relay circuits. The tool set includes a validator, a simulator, and model and safety condition generators that all take such documents as input. The validator can be used to check that the created documents follow the rules of the domain. The simulator can be used to validate that the electrical circuits behave as expected. The model and safety condition generators can be used to generate input to a model checker that then can be used to automatically verify that the described relay interlocking system is safe. To use such automated tools is a great improvement compared to manual inspections of diagrams: it is faster and easier to do, and it reduces the risk that errors or omissions are made.

Prototypes of most of the tools have been implemented, while a few of them are currently under development. In future work we plan also to experiment with other editors, visualisations, and model checking approaches to see what is most valuable and most efficient.

The framework has successfully been applied to verify that Stenstrup station in Denmark is safe. In future work it should be tested whether the framework can be applied to large stations without problems such as state space explosion during model checking.

Acknowledgements I would like to thank Kirsten Mark Hansen, Banedanmark, for providing the initial idea for this project and for many valuable discussions and suggestions. My thanks also go to my students Louise Elmose Eriksen and Boe Pedersen, and my former students Marie Le Bliguet and Andreas A. Kjær, who have all contributed to this project in their bachelor and master theses, respectively, and who have helped producing some of the figures in this paper.

References

1. Symbolic Analysis Laboratory, SAL, home page: `http://sal.csl.sri.com`, 2001.

2. Dines Bjørner. New Results and Current Trends in Formal Techniques for the Development of Software for Transportation Systems. In *Proceedings of the Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'2003), Budapest/Hungary*. L'Harmattan Hongrie, May 15-16 2003.
3. Marie Le Bliguet and Andreas A. Kjær. Modelling Interlocking Systems for Railway Stations. Technical Report IMM-M.Sc.-2008-68, Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2008. Master thesis supervised by Anne Haxthausen.
4. Leonardo de Moura, Sam Owre, and Natarajan Shankar. The SAL Language Manual. Technical Report SRI-CSL-01-02, SRI International, 2003. Available from `http://sal.csl.sri.com`.
5. Louise E. Eriksen and Boe Pedersen. Simulation of Relay Interlocking Systems. Technical Report IMM-M.Sc.-2007-04, Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2007. Bachelor thesis supervised by Anne Haxthausen and Hubert Baumeister.
6. Anne E. Haxthausen. Developing a Domain Model for Relay Circuits. *International Journal of Software and Informatics*, 3(2–3):241–272, 2009.
7. Anne E. Haxthausen, Marie Le Bliguet, and Andreas A. Kjær. Modelling and Verification of Relay Interlocking Systems. In Christine Choppy and Oleg Sokolsky, editors, *15th Monterey Workshop: Foundations of Computer Software, Future Trends and Techniques for Development*, number 6028 in Lecture Notes in Computer Science. Springer, 2010. Invited paper.
8. Anne E. Haxthausen, Jan Peleska, and Sebastian Kinder. A Formal Approach for the Construction and Verification of Railway Control Systems. *Formal Aspects of Computing*, online first 2009. Special issue in Honour of Dines Bjørner and Zhou Chaochen on Occasion of their 70th Birthdays.
9. The RAISE Language Group. *The RAISE Specification Language*. The BCS Practitioners Series. Prentice Hall Int., 1992.
10. The RAISE Method Group. *The RAISE Development Method*. The BCS Practitioners Series. Prentice Hall Int., 1995.
11. E. Schnieder and G. Tarnai, editors. *Proceedings of Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2007)), Braunschweig, Germany*. GZVB e.V., 2007. ISBN 13:978-3-937655-09-3.