Using formal methods for quality assurance of interlocking systems

Keywords: Formal methods, verification, specification, quality assurance, interlockings

Using formal methods for quality assurance of interlocking systems

L.-H. Eriksson
Industrilogik L4i AB, Box 440, SE-101 28 Stockholm,
Sweden. E-mail: lhe@L4i.se

K. Johansson
Swedish National Rail Administration, SE-781 85 Borlänge,
Sweden. E-mail: kjell.johansson@hk.banverket.se

Abstract

”Formal methods” denotes mathematical techniques making it possible to perform a formal mathematical proof of the compliance – or non-compliance – of a system to its specification. We discuss the views of the Swedish National Rail Administration on the use of formal methods to increase safety and reliability of safety-critical signalling systems.

As a study, a formal requirements specification for interlockings has been developed. and an existing relay-based interlocking of a common design has been modelled and analysed. The analysis uncovered a safety-critical design error which had not been discovered using the traditional methods of quality assurance.

1 Background

1.1 Drawbacks with traditional design review methods

Due to the severe safety requirements in railway signalling systems, the logic of an interlocking system must be thoroughly scrutinised before installation and commissioning. This work has traditionally been carried out by design reviews of schematics and other design documents. This manual process has a number of drawbacks:

- it is very time consuming.
- it must be performed by experienced specialists, who are scarce.
- it is very difficult to perform a complete review of a complex system, and it is also very difficult to prove the completeness or incompleteness of a traditional design review.
- it is difficult to accomplish a complete and unambiguous set of rules for signalling systems design, and a form for the rules that also is user-friendly.
- the manual review work is monotonous, tedious and boring.

1.2 Formal methods

In contrast to traditional methods, the term *formal methods* refers to the use of precise logical and/or mathematical methods to reason about properties of systems. Using a *formal specification*, an unambiguous description of the requirements of a system can be given. The description is unambiguous in the sense that there need be no doubt about its meaning. With formal methods one can also analyse the specification to check that it correctly reflects the intentions of the user.

Provided that a suitable description of the design of a system (e.g. an interlocking) is also available, *formal verification* can give a precise answer to the question of whether or not the system design actually satisfies its specification. Testing or other empirical analysis is – in principle – not necessary.

Of course, it is never possible using formal methods to guarantee that a device works as intended. Formal methods state nothing about whether or not a specification correctly captures the demands of users. Nor can they demonstrate that the device has been built in accordance with the design, or whether the designer has made correct assumptions about the properties of the components used to build the device.

However, many of the errors than can arise when designing a system can be eliminated using formal methods.

1.3 Earlier work with formal methods

Since 1978 the Swedish National Rail Administration has installed and commissioned more than 130 computer based interlocking systems. The interlocking logic in those systems has a geographical structure, and is built up of standardised modules. The correctness of these modules is ensured by a combination of traditional reviews and use of formal methods.

The modules are very stable, and updates are not frequent. Because of this, it has been possible to spend a relatively large amount of work on the reviews. With this experience, we are convinced that much more work is required to ensure that a computer based interlocking system is correct, compared with the corresponding work for a relay base interlocking system.

1.4 A formal specification

In 1995 the Swedish National Rail Administration initiated the work to describe the Swedish functional safety requirements in a formal specification. The aim of this project was to investigate the possibilities to enhance the speed and the quality of the design review work. The aim was also to, if possible, make use of the formal specification as the requirement part of a formal proof of an interlocking logic's correctness. An important prerequisite for the project was that the specification had to be generic, and not limited to a specific technical solution or a specific type of interlocking system.

During the work, a number of flaws in the interlocking design rules have been identified. It has become necessary to improve the set of rules in use, and to make decisions on new rules where appropriate. Thus, the work with a formal specification has enhanced the quality of the Swedish set of interlocking design rules.

2 A formal specification of interlockings

2.1 General

A formal specification for safety properties of interlockings has been developed. We will briefly describe that specification and how it has been used to formally verify the correctness of an actual interlocking. A complete presentation of this specification and verification work is given in Eriksson [1,2].

In contrast with previous work on specification and/or verification of interlockings (e.g. Groote [3], Hansen [4]), this work has been intended to describe an at the same time general and complete set of requirements. To facilitate the acceptance and practical use of the specification, it has been written using concepts traditionally used in Swedish signalling practise.

The specification only includes *functional* safety requirements. Safety requirements relating to the construction of the interlocking, such that certain failure modes must not lead to dangerous situations, are not included. Safety requirements relating to failures of track side equipment are included, however, as handling such situations is part of the normal function of an interlocking.

The specification has been written using a variant of first-order predicate logic including simple extensions to express changes over time. A tool has been built to facilitate development of the specification. Specifically, the tool can perform simulation of the specification as well as prepare input to the theorem prover tool used for formal verification.

2.2 First-order predicate logic

The language of first-order predicate logic uses symbolic expressions to describe facts about the world. E.g. the fact that the track circuit of point 21 is occupied could be represented by the expression `occupied(pt21)`. `pt21` is a symbolic name of the point in question and `occupied` is a symbolic name of the property of having an occupied track circuit. A relation between two objects, such that point 21 is a part of the train route beginning at signal 21 and ending at signal 31 could be represented by the expression `part_of(tr2131,pt21)`. Here `tr2131` is a symbolic name for the train route. A particular situation in an interlocking is represented by assigning to these expressions a *truth value*, truth or falsity.

More complex expressions are built using *logical connectives* AND, OR, NOT, -> (implies) and <-> (equivalence). E.g. the expression *X* AND *Y* states that *X* and *Y* are both true. Implication states that if one expression is true the other must be true also, e.g. *X* -> *Y* states that whenever *X* is true, then *Y* must also be true. Equivalence states that two expressions have the same truth value. In order to express general properties, the *quantifiers* ALL and SOME are used. The expression ALL pt *X* represents the fact that for *every* point, it must be the case that *X* is true. Similarly, SOME is used to state that a property holds for *some* object of a certain kind. The *variable* pt does not designate a particular point, but is used inside *X* to refer to an arbitrary point. In the actual specification it must be made explicit to which kinds of objects each variable refers, but here we will note this informally.

2.3 Descriptions of concepts and requirements

The formal specification consists of a number of expressions in predicate logic (*axioms*). The set of axioms can be roughly be divided into two parts. One part describes the different physical objects making up the environment of the interlocking (signals, points, etc.) as well as the abstract concepts used when expressing the requirements (train routes, geometric relations between objects in rail yards, etc.). The other part describes the actual requirements in terms of these defined concepts.

The part describing objects and concepts turns out to be most voluminous and difficult one and no examples from this part will be given here. Geometric properties in particular are complex to describe formally. At the same time, this part is of the least interest from the point of view of a signalling engineer, as these properties are generally intuitively self-evident and unproblematic. One of the conclusions of this work is that a specialised specification language where such properties and concepts are predefined would greatly facilitate the work of writing and understanding formal requirements on railway related systems.

The specification expresses the safety requirements in terms of predicate logic expressions. E.g. there is a requirement that if a point is occupied (by an engine or car), the point must be locked. This requirement can be expressed simply as:

```
ALL pt (occupied(pt) -> point_locked(pt))            (1)
```

...where pt is a variable that ranges over points. Given predicates to express the locking of train routes and to relate names of train routes to the parts of the rail yard that constitute the route, the requirement that all points in a locked train route must also be locked can be expressed as:

```
ALL pt (SOME tr (locked(tr) AND part_of(tr,pt))   (2)
 -> point_locked(pt)))
```

2.4 Time aspects

These sample formulæ all express requirements that must be satisfied in any moment of time without regard to the situations in previous moments. This is not sufficient to express such a requirement as ”A locked point which must not be instructed to *change* its position”. In order to express requirements that consider previous situations, a way of referring to different time instances is required.

We make the reasonable assumption that the interlocking operates fast enough that its output is always available when needed and that transients on the outputs are so short that they can not affect the environment. This motivates the *synchronous hypothesis* under which the interlocking can be described as working in a sequence of instantaneous steps (or moments of time). The interval between these steps have no fixed relation to actual time intervals, except that they are assumed to be short enough for the interlocking to fulfil any response time requirement.

In the few cases where the requirements need to refer to an actual time interval, the passage of time is modelled using a timer that sends a signal when the interval has passed.

In the logic, requirements on the behaviour of the interlocking over time is expressed using the temporal operator `PRE`. An expression `PRE` *X* refers to the truth value of *X* in the previous moment.

The requirement that ”a locked point must not be instructed to change its position” can now be expressed using the axiom:

```
ALL pt (point_locked(pt) ->                      (3)
          (left(pt) <-> PRE left(pt)))
```

In other words, if the point is locked, then it must be instructed to be in the same position at this moment as in the previous moment.

It suffices to consider the left position if we assume that the point is always instructed to assume either the left or right position. A separate predicate, `controlled`, represents information about whether the point actually is in the intended position.

2.5 Validation

To ensure that the formal specification correctly captures the intended safety properties, the specification has been *validated* in several ways. It has been used to simulate the behaviour of interlockings, and the behaviour has been checked for safety. Several safety properties not directly expressed by the specification has been formally proved to follow from it. Also, signalling experts from the Swedish National Rail Administration have inspected and approved a plain text translation of the specification.

2.6 Formal verification

The specification describes *general* safety requirements. In order to use the specification to describe requirements on a particular interlocking – e.g. to formally verify that interlocking – the specification must be supplemented with a description of the layout and properties of the particular rail yard controlled by the interlocking. This description is given as a set of facts in predicate logic. The specialised specification thus obtained states exactly what behaviour – i.e. output – of the interlocking is *permitted* for each input, given the situation in the previous moment of time.

To see this, assign to each predicate the truth value `TRUE` or `FALSE` depending on the state of the corresponding input or output of the interlocking. Suppose that point 21 is instructed to be in the left position, it was instructed to be in the right position in the previous moment and its track circuit is occupied, then `left(pt21)`, `PRE left(pt21)` and `occupied(pt21)` will be assigned `TRUE`, `FALSE` and `TRUE`, respectively. This is clearly incorrect behaviour as the interlocking has instructed the point to move although its track circuit is occupied.

This will violate the axioms of the specification. Exactly how depends on what the interlocking considers the locking status of the point to be. If the interlocking considers the point to be locked (`point_locked(pt21)` is true), then axiom (3) is violated. If it does not consider the point to be locked, then axiom (1) is violated.

By describing the working of the interlocking in logic, a description is obtained which states exactly what behaviour is *possible*. By checking that every possible behaviour is also permitted, the correctness of the interlocking can be demonstrated. Checking every possible behaviour separately would be unfeasible, but using modern theorem proving algorithms (such as the ones by Stålmarck [5] or Groote [6]), the check can be carried out very rapidly without having to check individual cases.

The difficulty of describing the interlocking in logic depends on the technology performing the interlocking function. With relays, it is straightforward. Programmable logic controllers offer no major problems, while general computer programs are more difficult but by no means unfeasible.

2.7 A case study

As a case study, the safety-critical part of the relay interlocking from the station at Brunna, close to the city of Uppsala, was formally verified as described above. The station in question comprised 2 points, 8 main signals and a number of other signals/information points. The relevant part of the interlocking included some 80 relays, 90 (binary) input signals and 60 (binary) output signals.

The check required a few minutes of computer time on a medium-speed UNIX workstation and revealed that the interlocking had a safety-critical design error. This interlocking and some 20-30 other interlockings of the same basic design were subsequently modified.

3 Future use of formal methods

3.1 Conclusions from the Brunna verification

Brunna is not a complex station. It is controlled by a relay based interlocking system, which is equally non-complex. The personnel responsible for the original, manual design review of Brunna were experienced and well qualified for their work. Despite all this, a safety-critical error passed the traditional, manual design review. The design error was easily recognised by an experienced reviewer, once he had become aware of the traffic situation where the error occurred. The difficulty was, even for this small, non-complex station, for the reviewers to find and foresee all possible traffic situations that could occur, and to take them into account during the design review. We have, through the formal verification of Brunna, become convinced that a formal verification is a useful method to overcome many of the drawbacks related to the traditional design review methods.

3.2 Applications for small, computer based interlocking systems

The Swedish National Rail Administration is in the process of introducing a new generation of small, computer based interlocking systems. In these, major parts of the safety logic is unique for each station. As software based systems are difficult to survey and to grasp, we consider it a major obstacle to show how all possible traffic situations will be handled by an interlocking system during a traditional design review. With the new generation of interlocking systems, we are therefore planning to introduce formal verification as a mandatory part of the system review before installation and commissioning of every new system. During 1998 work has been started to develop methods and tools for these activities. We are also working with the formal verification of the first interlocking system of the new type, which is planned for installation during autumn 1998.

The introduction of formal verification requires a large effort of work from highly educated and qualified experts, to design the necessary specifications, methods and tools. Once this is achieved, the future verification of a specific interlocking system can be performed faster and with improved quality by persons without expertise within the field of railway signalling.

References

[1] Eriksson, L.-H. *Formalising Railway Interlocking Requirements*, Technical report 1997:3, The Swedish National Rail Administration.

[2] Eriksson, L.-H., *Formal Verification of Railway Interlockings*, Technical report 1997:4, The Swedish National Rail Administration

[3] Groote, J.F., et.al., *The Safety Guaranteeing System at Station Hoorn-Kersenboogerd*, Logic Group Preprint Series No. 121, Department of Philosophy, Utrecht University 1994.

[4] Hansen, K.M., Validation of a Railway Interlocking Model, *FME'94: Industrial Benefit of Formal Methods*, eds. Naftalin, Denvir & Bertran, Lecture Notes in Computer Science 873, Springer-Verlag.

[5] Stålmarck, G. & Säflund, M., Modelling and Verifying Systems and Software in Propositional Logic, *Proc. SAFECOMP '90*, Pergamon Press 1990.

[6] Groote, J.F., *The propositional theorem prover HeerHugo*, World Wide Web, http://www.cwi.nl/~jfg/heerhugo.html, 1998.