Industrialising a proof-based verification approach of computerised interlocking systems

S. Behnia[1], A. Mammar[2], J.-M. Mota[3], N. Breton[4],
P. Caspi[5] & P. Raymond[5]
[1]*RATP, France*
[2]*Télécom & Management SudParis, France*
[3]*Thales Rail Signaling Solutions, France*
[4]*Prover Technology, France*
[5]*Verimag, France*

Abstract

This paper describes the formal verification of an interlocking system. We have formally proved the non-derailing and non-collision safety properties for an existing interlocking system operating on Paris Metro's line 3Bis. These high-level properties have first been refined to an intermediate level permitting their expression in terms of the control system's inputs and outputs. The resulting properties have then been formalised in the Prover iLock Verifier engine's internal language. The Prover iLock Verifier engine is a COTS commercialised by Prover Technology. For this project some specific features have been added to the engine to provide certified proofs that can be used, instead of testing, in the SIL-4 qualification process of interlocking systems.
Keywords: formal verification, interlocking application, proof certification, environment modelling.

1 Introduction

Interlocking systems are safety critical systems and thus require thorough validation before being set into operation. Usually, this validation is performed by means of testing and even though the experience of several decades shows that intensive testing is a very safe approach, it is also very expensive: the experience of RATP,


WIT Transactions on The Built Environment, Vol 103, © 2008 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/CR080151

operator of the Paris metro, reports that testing a medium range interlocking system requires many man months of work.

With the advent of computer technologies, another validation technology becomes available which consists of formally proving that a safety critical system is safe, *i.e.* it is able to prevent any dangerous situation to happen. This formal validation approach can be applied in two different ways:

Deductive proofs of a generic software: This approach is based on challenging mathematics to prove that the software will behave safely *by construction* and thus can be applied once and for all independently of the particular track layout it will be applied to. This approach has been advocated by many authors [1, 2] and has been successfully applied to another safety related problem, the problem of driverless metros [3].

Model-checking each particular system: This approach is based on less demanding mathematics because it aims to validate a more concrete system, such as an actual interlocking system operating on a specific track layout. Thus the set of possible states that this system can reach is finite and a computer could explore all of these states so as to check whether some are dangerous. This is the celebrated *Model-Checking* technology [4, 5].

Clearly, both approaches have their pros and cons. Deductive methods are applied only once but require challenging mathematics and thus a very skilled personnel. Also, the initial investment can be very high. Model-Checking is less demanding in terms of personnel but it has to be applied to each particular installation. Still this issue can become less critical if, after some initial investment, the validation of each new installation can be automated. It was the objective of the PROOFER project to study this later possibility by showing that this can be turned into a truly industrial process. This work has been jointly conducted by Prover Technology, solution provider for the formal verification of rail control systems, Verimag, an academic laboratory specialised in formal methods, jointly held by CNRS, University Joseph Fourier and Institut national polytechnique de Grenoble, Thales Rail Signaling Solutions (TRSS), provider of computerised interlocking systems and RATP and has been supported by PREDIT, the French research program in land transport. This paper aims at reporting on the achievements of this project.

1.1 Related works

Undoubtedly, formal methods have been particularly active in interlocking systems [6–10]. This may be due to the fact that these systems are both safety-critical and have a very logical flavour which makes them particularly well suited to formal reasoning. Yet their strong complexity makes them quite sensible to combinational state explosion. Very soon it appeared that SAT-based methods combined with induction [7, 9] were particularly well adapted to handle this explosion. Many studies have been devoted to comparing these methods with others: sym-

bolic BDD-based methods [11] and abstraction methods [6]. Other approaches have been based on modelling [10] and on artificial intelligence methods [9].

Literature on the application of formal methods to rail control systems is mostly devoted to experiments, which are quite far from real-world applications, the later ones being mostly covered under industrial non disclosure. This makes comparisons with the objectives of this project difficult. Yet, in Prover Technology's long experience on formal verification of rail control systems, the problem addressed during this project is amongst the most complex ones encountered, both in terms of state space and the high-level safety properties that are analysed. This thus makes the unique feature of the experiment reported here.

1.2 Paper's content

The second section presents an overall view of the proposed validation method, while the following sections will be devoted to giving insights on particular points: the third section focuses on the formalisation of safety properties, the fourth section discusses issues of environment modelling, and the fifth section highlights the proof certification technique used in the project. Finally, in conclusion we present the main results that have been obtained and the future work currently in progress.

2 The overall view of the validation process

The interlocking system that has been the basis for this work is the PMI Interlocking Solution provided by TRSS and chosen by RATP to equip different lines of its network.

Before explaining the validation process in section 2.2, the following section gives a brief description of the PMI system, in order to define the parts of the system that are covered by the formal verification.

2.1 PMI Interlocking Solution

The PMI Interlocking Solution is made of several subsystems: a local control and supervision subsystem which is mainly in charge of the signalmen dialogs and the communication with the centralised traffic control system; a SAM subsystem which is used for maintenance and helps the operator to diagnose encountered problems; and an MEI subsystem which constitutes the safety critical core of the system.

The MEI is interfaced with the field equipment and the local control and supervision subsystem. It is in charge of the execution of the signalman's controls and must ensure the appropriate control of the field equipment in order to guarantee the absence of dangerous situations. The interlocking application is implemented in the MEI software, constituted of two parts:

- A kernel software which ensures the internal functions such as field I/O, synchronisation, communication and graph interpretation. This piece of software is independent of any particular station.

- A specific software, built with the parameterisation tools developed by TRSS for PMI systems, implementing the interlocking logic of a particular station.

The specific software is developed using a graph formalism, similar to Petri Nets, specific to the PMI system. The interlocking logic is first implemented as generic concepts in this formalism. In order to build the interlocking application of a specific station, the generic graphs are *instantiated* using the station's track layout so that generic objects, representing for example track circuits or points as well as routes, are given specifics names representing specific objects of the track layout. The resulting instantiated graphs are called *applied graphs*.

2.2 Validation process

The validation process for the MEI software takes into account the specific design of this software that separates it in two distinct parts: the kernel software and the applied graphs. As the kernel software is independent of any particular interlocking application, its validation can be done once and for all. The applied graphs on the other hand are to be validated each time a new station is equipped.

The validation of the kernel software is achieved by testing processes following the requirements of the CENELEC EN 50128 standard [12] for SIL-4 software. The formal validation process aims to validate the applied graphs.

The formal validation solution is built upon the Prover iLock Verifier engine which is a Component Off The Shelf commercialised by Prover Technology. In order for the proof engine to take into account the applied graphs, they must first be translated into the internal formalism of the Prover iLock Verifier engine. The execution semantics of the applied graphs is informally expressed by the algorithm of the graph interpreter which is part of the kernel software. This algorithm has been analysed and formalised into an execution model included in the translation. The translation is done automatically by a tool specifically developed for this project by Prover Technology. The formalisation of the execution semantics of the applied graphs has also been fully documented and justified by Prover Technology.

The validation process is shown in Figure 1. The translated applied graphs are provided to the Prover iLock Verifier engine along with the safety properties that the software must satisfy. The safety properties are obtained by analysing the dangerous situations that must be avoided. These dangerous high-level situations are

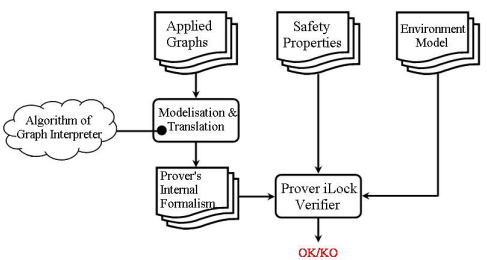

Figure 1: Validation process of the applied graphs.

derailing of a train and collision of two trains. How these high-level situations are formalised into safety properties for formal verification is described in the following section.

Proving safety properties may fail if inputs and outputs representing the field equipment are left completely free. This would mean that the trains can circulate without respecting the lineside signalling or that the points can move without being controlled. A model of the environment must thus be developed in order to restrain its behaviour. The environment modelling will be discussed in section 4.

The proof engine will formally verify that the safety properties are satisfied by the applied graphs. If a safety property can be violated by the applied graphs, the proof engine will provide a counter-example showing a possible execution of the applied graphs that leads to the violation of the property. The analysis of the counter-example permits either to debug the applied graphs or to adjust the environment model. In order to trust that formal verification results are correct, the project utilises proof logging and independent checking of proofs, as discussed in section 5.

3 Safety properties formalisation

The safety properties one would ideally like to formally verify are high-level properties; the interlocking system should prevent collisions between trains and train derailment. Unfortunately, checking such high level properties would require, besides a model of the interlocking program, also a full model of the environment, defining in details train movements, how drivers obey wayside signalling, how points move when controlled etc. Though such modelling is not out of reach in theory, specifying it would be difficult and error prone. In particular, one would have to be very careful not to over-constrain the model; it could lead to excluding real-life situations from the environment model. Also, such a detailed model could increase the complexity of the model and worsen the combinational explosion problem.

However, these high level properties can be refined into more manageable properties based on the various situations that can be considered, such as front collision, rear collision, side collision, derailment on badly set points, etc. Properties are finally instantiated on a particular track layout, *e.g.*, on which route does the collision take place, on which point does the derailment take place, etc.

The proposed solution consists of two steps:

1. Define medium-level safety properties in terms of variables of the MEI, by means of an informal analysis performed by railway engineers on fault-trees describing the high-level safety properties;
2. Instantiate these properties on the actual track layout and formally verify them against the translation of the interlocking system. When a check fails, a counter-example is returned and a human analysis is performed in order to determine whether this is actually due to a bug in the system or whether this counter-example is indeed an unrealistic one due to an inaccurate environment modelling. In the later case, environment constraints are introduced

up to a point where the counter-example disappears and the property can be proved. The way these environment constraints are defined is addressed in section 4. This method of modelling progressively the environment is expected to alleviate the complexity problem with respect to a full environment modelling.

From this fault-tree, instantiated properties in Prover iLock Verifier engine's internal language are manually extracted. In the course of the project two improvements have been prototyped:

- A formalisation of fault-trees [13] which allows an automatic extraction of these medium-level properties.
- A new type of generic graphs allowing the expression of safety properties. Using such graphs would make it possible to automatically instantiate a generic safety property on a particular track layout.

These two improvements have not yet been interconnected and this remains to be done in order to fully automate the process of safety property expression (see section 6).

4 Environment modelling

As we have seen in the previous section, the safety properties that we want to prove aim at ensuring safe train traffic without any risk of accident (derailing, collision). As stated in section 2.2, it is inconceivable to prove such properties if the environment can change freely. So, we have to describe the behaviour of this environment with respect to the different controls (outputs) sent to the environment by the MEI: we have to specify that a point cannot move without being controlled by the MEI, and that a train will not pass a signal with a restrictive indication.

As stated previously, environment modelling, *i.e.* the description of the behaviour of the field equipment and the signalmen, has been guided by counter-examples found by the proof engine. Indeed, our goal is to find the less constrained environment that enables us to prove the safety properties. We will only discuss here the environment modelling for points.

The MEI reads the position of a point by means of two sensors corresponding to the right and left position of the point. The sensors are wired to the MEI's I/O device by means of *fail-safe* relays. By *fail-safe* relay, we mean that if the MEI detects, through its sensors, a point in a given position, it is guaranteed that this point is really in this position. This behaviour is described in our model by the following predicates:

$$MEI\_SW_R = SW_R \wedge Sensor_R$$
$$MEI\_SW_L = SW_L \wedge Sensor_L$$

These predicates mean that the MEI sees a point SW in the right position MEI_SW_R (resp. left position MEI_SW_L) if the point is actually on the right SW_R (resp. on the left SW_L) and if the sensor associated with this position $Sensor_R$ (resp. $Sensor_L$) is not defective.

The first constraint that we have written describes the fact that a point cannot be in both positions at the same time.

We have then specified that a point cannot move without being controlled, *i.e.* if a point is in a specific position in a given MEI execution cycle, and the MEI is not controlling this point to the opposite position at that time, the point will stay in this position in the next MEI cycle.

Other constraints, necessary to prove the safety of the application, specifies that if a point is in a position in a given MEI execution cycle, it cannot be in the opposite position in the next cycle. In other words, a point cannot move from a position to the opposite during one MEI execution cycle. Considering MEI's execution cycle time and the time necessary for a point to physically move from one position to the opposite, these constraints can be approved.

Failures of devices are handled by means of cancellation operations, performed by the signalman, which override safety functions. For example, if, because of the failure of the sensor of a point, the MEI doesn't permit a route to be set, the signalman can cancel the corresponding locking of the point and force the MEI to set the route. In this case, the safety becomes the responsibility of the signalman. The cancellation begins with a control sent to the MEI by the signalman, and must be confirmed by the action of a second operator on a button placed on the track near the point.

The cancellation operation has also been specified in our study by means of constraints. Suppose that *track_side_button* represents the MEI's input which is set if the cancellation button is pressed. The constraint specifies that if the point is not in the required position with respect to the route to be set, the cancellation button must not be pressed, *i.e. track_side_button* can not be set. In other words, a human failure resulting in a misunderstanding of the required position of the point may cause a dangerous situation.

5 Proof certification

To enable the use of the proofs performed by Prover iLock Verifier as the safety verification activity in the development of a SIL-4 safety critical software, it must be guaranteed that the proofs are *correct*, *i.e.* the proof engine will not report a property as valid if it can be falsified.

Prover Technology's proof engines are made of a large number of complex cooperating algorithms in constant evolution, and involves confidential intellectual property. Furthermore the analysis of a system by the proof engine is a task that involves heuristics and numerous try-and-fails before a proof is found. Taking all of these parameters into account, attempts to verify the proof engine implementation would have been unreasonable.

The Proof Logging and Proof Checking mechanisms permit to overcome this difficulty by saving the result of the proof search in a file (*i.e.* Proof Logging) and having a simple software checking that this proof is correct (*i.e.* Proof Checking).

Prover Technology has extended its proof engine with the functionality of saving in a file the proof of a property when it has been found to hold. This proof

log file includes two sections; the first section contains an encoding of the system and the properties that were analysed, and the second section contains the proof of each property expressed in terms of a deduction tree using a few generic inference rules. Such a proof brings the mathematical evidence that the property holds. The Proof Log Checker takes as input the model of the system and its properties together with the corresponding proof log, and checks that the proof log encodes the correct system, and that the proofs of the properties are correct. It is important to realise that, if finding the proof of a property is in general difficult, checking its correctness is rather easy. The Proof Log Checker is thus a simple software that can be thoroughly reviewed and verified. The review and verification process of the Proof Log Checker has been defined based on the CENELEC EN 50128 standard [12] requirements for a safety critical software.

6 Conclusions and perspectives

We have presented in this paper the results of the PROOFER project. During this project, a set of tools and methods dedicated to the formal verification of an interlocking system have been developed.

In order to verify that the software of the interlocking system does not allow dangerous situation to happen, such as derailing and collision of trains, this formal verification process involves 3 elements:

1. A formal model of the system obtained by translating a set of graphs implementing the interlocking logic;
2. A set of safety properties obtained by refining high level properties into medium level properties using a fault-tree formalism;
3. A model of the system's environment obtained using proof results.

The methods and tools designed in the PROOFER project were applied to an industrial interlocking system. These experiments allowed to demonstrate the feasibility of this method and the capacity of the tools to handle industrial systems.

For the Paris metro line 3Bis, 11 generic properties have been defined: 5 to prevent the derailing of a train and 6 to prevent the collision of two trains. These generic properties have generated 124 instantiated safety properties that the system must verify to meet its safety requirements. The proofs of these properties has been conducted by TRSS thanks to the help of Télécom & Management SudParis and RATP. The whole proof activity was supervised by RATP and supported by Prover Technology and Verimag.

These results have encouraged RATP, TRSS and Prover Technology to pursue their partnership in order to realise the formal verification of future PMI systems. For this purpose, the tools will be improved and industrialised in order to:

- Automate the process as much as possible;
- Reduce the time of the translation of graphs (currently, for an average application, the translation can take between a few hours to several days).

As stated previously, safety properties are manually defined based on an informal analysis performed by railway engineers on fault-trees describing high-level safety properties. In [13], Verimag proposes a formal fault-tree formalism to auto-

matically extract those safety properties. This is an improvement that could be integrated into the existing methodology.

Currently, safety properties are expressed using temporal logic formulas (this is a low-level input language of Prover iLock Verifier engine). Another change being considered for future work is to express safety properties using the Petri Net formalism of the applied graphs, to enable:

1. Using the same formalism for both design and safety properties;
2. Signalling and design engineers that are used to the formalism of the applied graphs to develop the safety properties;
3. Using existing tools to perform the instantiation of generic safety properties extracted from a fault-tree for a particular track layout.

Thus, it would ease the process of expressing the safety properties, giving a highly automatic solution for the verification of interlocking systems.

Acknowledgement

This work has been jointly conducted by Prover Technology, Verimag, Thales Rail Signaling Solutions and RATP and has been supported by PREDIT under the name PROOFER.

References

[1] Jones, C.B., *Systematic Software Development using VDM*. Prentice Hall International, 1990.
[2] Abrial, J.R., *The B-Book*. Cambridge University Press, 1995.
[3] Behm, P., Desforges, P. & Meynadier, J., Météor : An industrial success in formal development. *B'98: Recent Advances in the Development and Use of the B Method*, ed. D. Bert, Springer, volume 1393 of *Lecture Notes in Computer Science*, 1998.
[4] Queille, J. & Sifakis, J., Specification and verification of concurrent systems in Cesar. *International Symposium on Programming*, Springer Verlag, volume 137 of *Lecture Notes in Computer Science*, pp. 337–351, 1982.
[5] Clarke, E., Emerson, E. & Sistla, A., Automatic verification of finite-state concurrent systems using temporal logic specifications. *TOPLAS*, **8(2)**, pp. 244–263, 1986.
[6] Bernardeschi, C., Fantechi, A., Gnesi, S. & Mongardi, G., Proving safety properties for embedded control systems. *Dependable Computing ? EDCC-2: Second European Dependable Computing Conference*, eds. A. Hlawiczka, J.G. Silva & L. Simoncini, volume 1150 of *Lecture Notes in Computer Science*, 1996.
[7] Sheeran, M. & Stålmarck, G., A tutorial on Stålmarck's proof procedure for propositional logic. *Formal Methods in System Design*, **16(1)**, pp. 23–58, 2000.
[8] Roanes-Lozano, E., Laita, L.M. & Roanes-Macas, E., An application of an ai

methodology to railway interlocking systems using computer algebra. *Tasks and Methods in Applied Artificial Intelligence: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, eds. A.P. Pobil, J. Mira & M. Ali, volume 1416, 1998.

[9] Ferreira, N.G. & Silva, P.S.M., Automatic verification of safety rules for a subway control software. *Formal Methods for Industrial Critical Systems, FMICS05*, volume 130 of *Electronic Notes in Theoretical Computer Science*, 2005.

[10] Banci, M., Geographical versus functional modelling by statecharts of interlocking systems. *Formal Methods for Industrial Critical Systems, FMICS04*, volume 133 of *Electronic Notes in Theoretical Computer Science*, 2005.

[11] Huber, M. & King, S., Towards an integrated model checker for railway signalling data. *FME 2002: Formal Methods - Getting IT Right : International Symposium of Formal Methods Europe*, volume 2391 of *Lecture Notes in Computer Science*, 2002.

[12] CENELEC, Software for railway control and protection systems. Technical Report EN50128:2001, Comité Européen de Normalisation Electrotechnique, 2001.

[13] Torrini, P., Caspi, P. & Raymond, P., From Fault-Trees to Safety Conditions. Technical Report TR-2007-6, Verimag, 2007. Available at http://www-verimag.imag.fr/TR/TR-2007-6.pdf.