Interlocking Design Automation using Prover Trident

Arne Borälv[1]

[1] Prover Technology, Krukmakargatan 21, SE-118 51, Stockholm, Sweden
arne.boralv@prover.com

Abstract. This article presents the industrial-strength Prover Trident approach to develop and check safety-critical interlocking software for railway signaling systems. Prover Trident is developed by Prover Technology to meet industry needs for reduced cost and time-to-market, by capitalizing on the inherent repetitive nature of interlocking systems, in the sense that specific systems can be created and verified efficiently as specific instances of generic principles. This enables a high degree of automation in an industrial-strength toolkit for creation of design and code, with seamless integration of push-button tools for simulation and formal verification. Safety assessment relies on formal verification, performed on the design, the revenue service software code as well as the binary code, using an independent toolset for formal verification developed to meet the applicable certification requirements. Basic ideas of this approach have been around for some time [1,2], while methodology and tools have matured over many industrial application projects. The presentation highlights the main ingredients in this successful application of formal methods, as well as challenges in establishing this approach for production use in a conservative industry domain.

Keywords: Formal verification, Sign-off, Interlocking, Prover Trident.

1 Background

In railway signaling, interlocking systems control the signals, switches and other wayside objects to ensure railway operations are always safe. Interlocking systems used to be based on electro-mechanical relays, with most new systems being computerized. The interlocking principles vary considerably, in different countries and for different railway infrastructure managers, with high life-cycle cost (one of the costliest railway signaling components). The plethora of different signaling principles contributes to that development and checking is time-consuming and costly. The adoption of new technology such as automation tools is slow, due to the conservative nature of the industry, and the stringent safety integrity levels that pose a challenge of trust in automation tools. There is also resistance in terms of minds to change and win over.

This article presents Prover Trident, an industrial-strength approach based on formal methods to develop and check safety-critical interlocking software for railway signaling systems that is used in production.

2 The Prover Trident Process

The Prover Trident process automates development and checking of interlocking software, leading to reduced effort and cost, and predictable schedules. This section outlines the main steps in configuring and using Prover Trident. Compared to "traditional" processes, more effort is spent on requirement specification, and a greater level of requirements precision is required. This extra effort is worthwhile, due to savings achieved from automated development of many systems, and the reduced long-term maintenance costs. Prover Trident is based on the following three main components:

- **PiSPEC IP:** A formal specification library of generic interlocking system requirements defined in PiSPEC, an object-oriented language supporting many-sorted first order logic and Boolean equations.
- **Prover iLock:** An Integrated Development Environment (IDE) that generates the design, test cases and safety requirements based on the PiSPEC IP and a system configuration, with push-button tools to automate simulation, formal verification and code generation.
- **Prover Certifier**: An independent sign-off verification tool that formally verifies that the revenue service code satisfies all safety requirements, using a process and tool chain designed to meet safety certification standards thanks to the use of diversified translation of input models, and proof logging and checking.

2.1 PiSPEC IP

The generic principles for a family of interlocking systems are determined by analyzing applicable standards, requirement specifications, interfaces, rules and regulations. To ensure good quality of results, with predictable development and maintenance, the principles are defined based on an *object model* (see **Fig. 1**). The object model defines the underlying ideas and objects, including both physical objects (e.g. signals, switches) and virtual objects (e.g. routes, protection areas), along with their properties and relations. The object model provides a common interface to ensure coherence and consistency in defining all interlocking software requirements, with clear separation of design, test and safety requirements. Configuration of individual interlocking systems populates the object model.

The object model and the generic design, test and safety requirements are defined in PiSPEC, using many-sorted first order logic. This enables to express requirements that are generic, using quantifiers over different sorts, corresponding to different types of objects in the object model. For example, to express a generic safety requirement that *a signal must display the stop aspect if it does not have control line safety* can be expressed as follows, where cl_safety() and stop() are predicates defined for signals:

$$\text{ALL si:SIGNAL (not cl\_safety(si) } -> \text{ stop(si))} \quad (1)$$

A non-trivial task is to verify completeness of the safety requirements. This task is usually managed based on manual review by domain experts, and/or diversification.

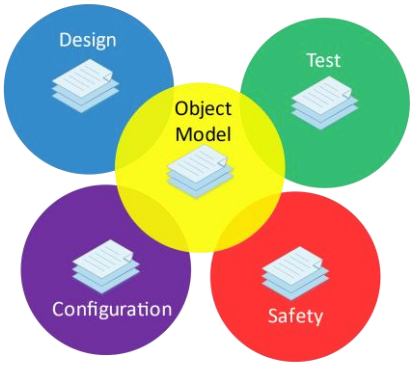

Fig. 1. Structure of generic principles (PiSPEC IP)

2.2 Prover iLock

Prover iLock is an Integrated Development Environment (IDE) for development of interlocking software. An individual system is created based on a high-level configuration of its track and signal arrangement that is created graphically (and other tabular data that can be imported). Prover iLock generates the design, test cases and safety requirements based on the PiSPEC IP and the system's configuration. Push-button tools can then be applied directly, for formal verification of design safety (Verifier), automated simulation of the design with its environment models using time-compression optimization techniques (Simulator), and generation of software code for revenue service (Coder).

Prover iLock supports animation in the graphical railyard configuration, providing visualization of state from interactive simulation or from counterexamples to safety requirements. This is useful for education purposes and during development of the PiSPEC IP. (In production use, simulation and verification are run in batch mode, normally not failing).

2.3 Prover Certifier

Using Prover Trident, safety verification of the revenue service code (target code) generated from Prover iLock is done using an independent sign-off verification tool. This step proves that the target code satisfies the same safety requirements that were verified against the design. This step also proves the equivalence of the design and the target code, or alternatively the equivalence of the target code and the resulting binary code.

A sign-off verification tool is based on Prover Certifier, which has been designed and developed as a reusable component that meets strict certification requirements. Prover Certifier uses techniques to reduce risks that errors go undetected, including diversified processing of input data using multiple implementation languages and using proof logging and proof checking. A sign-off verification tool extends Prover Certifier with diversified translators for the target code and the binary code.

The configuration data that needs manual review per interlocking system should be kept small. A dedicated format called LCF is used for this purpose. This format provides easy-to-review, compact representation of configuration data, supported by diversified translation to Prover Certifier input.

3 Results and conclusions

The Prover Trident approach is the result of many years of experience from formal verification of interlocking systems, and from development of high-integrity tools for formal verification. This has made it possible to automate repetitive and time-consuming tasks in development and checking of railway interlocking software, with manual tasks mainly for creating the system configuration, the use of push-button tools and running the sign-off verification. With new systems developed with much less effort, efficiency is increased, and time-to-market and cost are reduced. It also ensures that each system is developed and verified based on same (reusable) principles.

The Prover Trident approach is used for creating revenue service interlocking software for application in urban metro, mainline railway, light-rail and, in its core parts, even in freight railways. The process and tools used are essentially the same – the differences imposed for different target platforms and railway infrastructure managers are minor, with a variety of target platforms being supported. Generic tools can be extended by adding code generators and customizing the sign-off verification tool. This enables the support of any interlocking system type and any target, at least in principle.

The big difference lies in input data, in the form of different PiSPEC IP. Using the Prover Trident approach, the truly creative aspects lie in the specification work required for defining the generic interlocking software principles.

There are no real technical obstacles for using Prover Trident for interlocking software development. Rather, the main challenge lies in conversion from old habits and the many hearts and minds to win over in the conservative railway signaling domain. In addition, commercial aspects and job security are also real concerns.

References

1. Borälv, A. Case Study: Formal Verification of a Computerized Railway Interlocking. In Formal Aspects of Computing (1998) 10: 338. https://doi.org/10.1007/s001650050021.
2. Borälv A., Stålmarck G. (1999) Formal Verification in Railways. In: Hinchey M.G., Bowen J.P. (eds) Industrial-Strength Formal Methods in Practice. Formal Approaches to Computing and Information Technology (FACIT). Springer, London.
3. Duggan, P. (Siemens) and Borälv, A. Mathematical Proof in an Automated Environment for Railway Interlockings, Technical Paper in IRSE Presidential Programme, 2015. IRSE NEWS 217.
4. Layout Configuration Format (LCF) v1.1, Format Specification. PCERT-LCF-FMT, version 1.0, Prover Technology, 2018.