Some Trends in Formal Methods Applications to Railway Signaling

Alessandro Fantechi[1], Wan Fokkink[2,3], and Angelo Morzenti[4]

[1] Università degli Studi di Firenze, Dipartimento di Sistemi e Informatica, Florence, Italy
[2] Vrije Universiteit, Sectie Theoretische Informatica, Amsterdam, The Netherlands
[3] CWI, Specificatie en Verificatie van Embedded Systemen, Amsterdam, The Netherlands
[4] Politecnico di Milano, Dipartimento di Elettronica e Informazione, Milan, Italy
fantechi@dsi.unifi.it,wanf@cs.vu.nl,morzenti@elet.polimi.it

Keywords: railway signaling, interlockings, CENELEC EN50128, B method, EURIS, software procurement

Abstract. Railway signaling is often considered as one of the most fruitful areas of intervention by formal methods. Many success stories have been told about the application of formal specification and verification techniques in this area. In this chapter, we investigate the reasons of these successes, and we offer some insight into the actual industrial usage of formal methods in this field, which does not yet meet the promises of the aforementioned success stories, but is steadily increasing, especially on the side of railway operators and infrastructure companies. The external conditions which are driving industrial choices and the trends of operators and infrastructure companies are also discussed, as well as the classification of railway signaling devices into categories more or less amenable to formal methods application.

1 Introduction

National railway systems are managed by transportation service providers that do not develop computer-based systems themselves. Rather, they act as system integrators of systems purchased from external suppliers. Service providers therefore have the problem of managing acquisition and integration of purchased subsystems. Hence, such organizations need clear, unambiguous, possibly formal requirement specifications. Both the purchaser and their suppliers must agree on rigorous acceptance procedures, based on verification and validation, functional and safety assessment, and safety approval. Furthermore, uniform, possibly standardized documentation is essential to permit monitoring of system development and to facilitate operation and maintenance. The procurement task is made more difficult by the requirement, under applicable national and international laws, that safety-critical railway systems must satisfy international standards, dictating procedures for design, deployment and maintenance.

Railway signaling is often considered as one of the most fruitful areas of intervention by formal methods. Many success stories have been told about the application of formal specification and verification techniques in this area. The width of the proposed usage of formal methods in this field is witnessed by the number of references to relevant articles in a recent (still far from complete) review by Dines Bjørner [7]: 182 references, and many others have followed in the last two years. Moreover, work performed at railway companies is often not published for confidentiality reasons.

There are two main reasons for the success of formal methods applications to railway signaling. On the one hand, railway signaling has always generated the interest of formal methods researchers, due to its safety-criticality, and the absence of complex computations and hard real-time constraints, making it a promising application field. On the other hand, railways have always had a very strong safety culture, based on simple fail-safe principles. In electromechanical equipments, used in most signaling systems before the introduction of computers, gravity was used to bring a system to the fail-safe state (e.g. all signals to red) in any occurrence of a critical event. The fact that computers have no gravity, that is, the impossibility of predicting in general the effects of the occurrence of faults, has long delayed the acceptance of computer-controlled signaling equipment by railway companies. The employment of very stable technology and the quest for the highest possible guarantees have been key aspects for the adoption of computer-controlled equipment in railway applications. Formal proof, or verification, of safety is therefore seen as a necessity.

In this chapter, we offer some insight into the actual industrial usage of formal methods in this field, which does not yet meet the promises of the aforementioned success stories, but is steadily increasing. The external conditions which are driving industrial choices are also discussed, as well as the classification of railway signaling devices into categories more or less amenable to formal methods application. Here we give only some personal and partial views and experiences in the field, and do not intend to exhaustively cover the field of railway signaling applications of formal methods. In particular, we only address the European railway signaling market, where actually the most important applications of formal methods to railways can be found, and which has undergone several dramatic changes in the last decade.

The structure of this chapter is as follows. In Section 2, the EN50128 guidelines by the European Committee for Electrotechnical Standardization regarding the development of software for railway signaling are discussed. Section 3 reports on a comparative case study of the applicability of different formal methods to railway signaling. Section 4 is devoted to applications in the railway domain of one such formal method, namely B. In Section 5, we focus on formal methods applications to railway signaling equipment, which is divided into train control systems and interlocking systems. Finally, Section 6 contains some conclusions.

2 CENELEC Guidelines

The EN50128 guidelines [29], issued by the European Committee for Electrotechnical Standardization (CENELEC), address the development of "Software for Railway Control and Protection Systems", and constitute the main reference for railway signaling equipment manufacturers in Europe, with their use spreading to the other continents and to other sectors of the railway (and other safety-related) industry.

The EN50128 document is part of a series of documents regarding the safety of railway control and protection systems, in which the key concept of Software Safety Integrity Level (SSIL) is defined. One of the first steps indicated by these guidelines in the development of a system is to define a Safety Integrity Level (SIL) for each of its components, on the basis of the level of risk associated, by means of a risk assessment process. Assigning different SILs to different components helps to concentrate the efforts (and therefore the production costs) on the critical components. The SILs are: 4 (very high), 3 (high), 2 (medium), 1 (low), and 0 (not safety-related).

The EN50128 guidelines dictate neither a precise development methodology for software, nor any particular programming technique, but they classify a wide range of commonly adopted techniques in terms of a rating (from "Forbidden" to "Highly Recommended" and "Mandatory") with respect to the established SIL of the component. Formal methods (in particular CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B are cited as examples) are rated as highly recommended for the specification of systems/components with the higher levels of SIL. Formal proof is also highly recommended as a verification activity. Anyway, both are not classified as mandatory, since alternative, more traditional techniques are also accepted. We should notice however that this is the first time (the first edition of EN50128 dates back to 1994) that a strong indication about the usage of formal methods appears in standard guidelines.

Indeed, despite CENELEC directives and success stories, formal methods have not permeated the whole railway signaling industries, where much software is still written with traditional means. This is due to the investments needed to build up a formal method culture, and to the high costs of commercial support tools. Moreover, equipment can conform to CENELEC without applying formal methods. Verification by thorough testing can be claimed compliant to EN50128. But the guidelines require, for the highest SILs, that design and verification is carried independently by two separate teams. Relying only on testing shifts an enormous effort (usually more than 50% of the total development effort) on the shoulders of the testing department, which is often considered less important (and hence less funded) than the design department. This becomes a risk for a company that is more and more required by the market to be CENELEC compliant, and the only solution is to shift back the effort to the design team, by introducing formal methods in the specification and design phases. This is a necessity that companies begin to realize, and success stories and CENELEC guidelines have had an important part in the raising of this consciousness.

3 Software Procurement in Railway Signaling

This section reports on experiences in a joint project between Politecnico di Milano and Italian State Railway FS, Infrastructure Department (which recently became Rete Ferroviaria Italiana S.p.A.). The purpose of the project was to define procedures and rules for managing software procurement for safety-critical signaling equipment [33]. The latter includes a broad range of devices, governing lines and tracks in stations, railway/road crossings, and train movements. Goals of the project, which were imposed as additional constraints, were:

- to cover all phases of system development, from requirements elicitation to implementation, final validation, approval and acceptance;
- to provide requirements on methods, languages and tools to be used during software development, without any bias towards any particular technology or tool provider. The only general requirement is technical soundness and being up-to-date with respect to the current advances in computer science;
- to provide results consistent with, and acceptable against, international standards (mainly the EN50128 standard, see Section 2);
- to choose methods that are sufficiently mature for industrial usage, are supported by automated tools, and are likely to gain acceptance by average engineers, both in the railway and computer technology domains.

The main working group in the project was composed of two researchers from Politecnico di Milano, expert of formal methods, and two engineers from Rete Ferroviaria Italiana, skilled in the modelling and analysis of railway signaling equipment. No formal procedure or metric was adopted for the evaluation of the various formal methods; these were analyzed through a detailed, careful investigation of the available technical documentation and scientific literature and on the basis of previous experiences in using the notations and tools. To validate the obtained results and provide some empirical support, a small-scale experiment was conducted, consisting of the formal specification of a simple signaling apparatus. The descriptions obtained in the various notations were compared with respect to compactness and readability of the produced artifacts, as these qualities were considered the most important to favor the practical application of the method.

As a result of the project, requirements and recommendations were issued, tailored to various kinds of systems. The main contribution of the project concerns requirements on specification, verification and validation. In particular, adoption of formal methods in the specification phase is recommended, when supported by suitable tools and validation and verification techniques. Here we report on the results concerning a comparative evaluation of methods, tools and notations for formal requirements specification.

3.1 System Classification

The recommendations were based on a system classification according to three categories: complexity (low, medium, high), criticality, and presence of temporal requirements. Complexity was assumed to be conventionally determined by the purchaser, while the degree of criticality was determined according to the SIL (see Section 2) of the application. The temporal requirements were classified in three categories: time independent, qualitative time, and quantitative time. The time independent category refers to systems without any particular temporal constraints, e.g. performing pure data or signal elaborations. The qualitative time category refers to systems that can restrict to time-ordered values and actions without quantitative information about time instants and time distances. Some improper real-time systems are in this category: systems with strict and binding requirements, but designed to adequately manage every possible delay or anticipation, or the absence of expected events. The quantitative time category comprises hard real-time systems. These systems interact with processes that are not completely manageable or controlled, and that cannot avoid a quantitative expression of their temporal constraints (instead of just an order relation among events) without severe consequences. It is worth pointing out that the category of qualitative time is quite different from the so-called soft real-time systems, i.e., systems where missing some (quantitatively of qualitatively stated) time constraint is undesirable or annoying but does not cause unacceptable dam-age; also it does not correspond to high-throughput systems, which must have the capability of processing high quantities of data, but with time requirements that are expressed in statistical terms. This is because the systems under consideration were in any case critical for safety and economic reasons, so that missing time requirements (even when these are qualitative) is not admitted.

3.2 Requirements Analysis and Specification

Table 1 shows the prescription on specification and validation techniques and generation of functional test cases, depending on the SIL of the system, its complexity and its temporal features.

The table is divided into four parts: analysis, syntax checks, degree of specification coverage, and validation accuracy degree. *Analysis* is itself divided into two kinds of activities for the validation of specifications: simulation or trace generation, and proof of properties. (1) *Simulation* of the behavior of a system means generating (possibly in an interactive or semi-automatic way) events and actions in a chronological order, while *trace generation* means generating (again possibly in a semi-automatic way) execution traces of the system, and verifying automatically whether these traces are compatible with the specification. Unlike simulation, in trace generation, events and actions are not necessarily generated in chronological order. (2) *Proof of properties* indicates in how far it is possible to prove mathematically (by means of logical demonstrations or exhaustive analysis) that a system possesses properties like safety, absence of deadlock, etc. Such proofs are classified according to four categories: without abstraction, with abstraction, generality, and automation degree. (2.1) *Without abstraction* means that proofs can be executed on the complete specification of the system. They therefore have a total degree of certainty: the specified system without doubt possesses the proved property. (2.2) *With abstraction* means that proofs can be

			SIL		COMPLEXITY			TIME		
			1,2	3,4	LOW	MED	HIGH	IND	QUAL	QUANT
Analysis	Simulation/Trace Generation		YES	YES	*	*	YES	*	YES	YES
	Property Proof	No Abstraction	*	YES(3)	*	*	*	*	*	*
		Abstraction	*	YES	*	*	*	*	*	*
		Generality	*	YES(3)	*	*	*	*	*	*
		Automation	*	S	*	*	*	*	*	*
Syntax Checks			YES	YES	*	YES	YES	*	YES	YES
Degree of Specification Coverage			T	T	*	*	*	*	T(1)	T(1)
Validation Accuracy Degree			β	$\gamma(2)$	0	α	β	0	$\beta(1)$	$\gamma(1)$

Table 1. Validation prescriptions. **Legend** $*$: there is no recommendation, neither in favor nor against adoption of the technique; (1): the indicated coverage or accuracy is a minimum requirement for the temporal parts alone; (2): it is recommended, but not mandatory at the current state of the art, to use a method with degree of accuracy δ; (3): recommended, but not mandatory at the current state of the art.

executed by introducing suitable approximations (abstractions) of the original specification, e.g. by ignoring the actual data in the system. Abstractions make proofs simpler, but may reduce the degree of certainty of the result. (2.3) *Generality* means that properties to be proved can be chosen by the user in a general and flexible way, using a suitable, sufficiently expressive mathematical notation. (2.4) *Automation degree* evaluates the support offered by the proof tools: S indicates that at least a semi-automatic support is required (the tools support the verification that the proof is correct, for instance by preparing a structure for the proof obligations or by automating the trivial parts and sub-proofs, but must be guided by expert users), otherwise proofs can be carried out manually.

Syntax checks expresses whether there is tool support for checking that a specification is syntactically correct. For *Specification coverage* the requirement T stands for *total*: all requirements have the same relevance and must therefore be specified, otherwise it can be the case that some requirements, identified in an unambiguous way and totally isolated from the others, do not have any influences on safety and need not be formally specified. Finally, *validation accuracy* measures the accuracy of the validation of the requirements specification. Values are, in order of increasing accuracy and hence of preference: 0 (informal inspections, walkthrough), α (syntactic control of types, coherence between definition and use of the entities that compose the specification, i.e. the typical static controls carried out by the compilers of modern programming languages), β (at least one of the following: simulation, animation, generation of traces, symbolic analysis, reachability analysis, proofs of certain properties like absence of deadlock, proof of properties with abstraction), γ (same as β, but with a combination of at least two techniques of a different nature, and adoption of suitable metrics in order to measure the coverage degree of a system analysis), and δ (statement and proof of general properties).

Notice that the columns of table 1 deal separately with the various system features, but their prescriptions are intended to be applied in conjunction (in other terms, the more severe requirement applies): for instance, for a system of medium complexity and qualitative temporal features the required validation accuracy degree is β; for a system of low complexity but highly critical (SIL 3 or 4) the required validation accuracy degree is γ.

Finally, Table 2 shows the application of the prescriptions of Table 1 to a set of widely used formalisms, considering both language features and current tool support. The analyzed notations and formal methods are Z [46], TRIO [35], Statecharts [37], SDL [24], UML [14], LOTOS [22], Petri nets [45], SCADE [15], and B [1]. Table 2 is obtained by comparing, for each notation and corresponding method and tool environment, the characteristic features and the tool support with the requirements expressed in Table 1. Not surprisingly, the state of the art is still unsatisfactory, even for methods and tools that received the best score, in the case of systems with quantitative timing and a high level of complexity. In this case there is no "strongly recommended" method and tool, the existing ones being only "recommended". This is due to the fact the currently available tools for analysis and verification of formal models lack a rigorous formal basis, or are neither certified nor validated by repeated and long-lasting application in an industrial setting. However, as already mentioned in the introduction, in railway signaling there are in general no complex computations or hard real-time constraints.

	SIL		COMPLEXITY			TIME		
	1,2	3,4	LOW	MED	HIGH	IND	QUAL	QUANT
Z	YES	NO(1)	YES	YES	NO(1)	YES	YES	NO
TRIO	NO(1)	NO(1)	YES	NO(1)	NO(1)	YES	YES	YES
STATE CHARTS	YES	YES	YES	YES	YES	YES	YES	YES(2)
SDL	YES	YES	YES	YES	YES	YES	YES	YES(2)
UML	NO(3)	NO	NO(3)	NO	NO	YES	YES	NO
PN	YES	NO	YES	NO	NO	YES	YES	YES(2)
LOTOS	YES	NO	YES	YES	NO	YES	YES	YES(2)
SCADE	YES	NO(1)	YES	YES	YES	YES	YES	YES(2)
B	YES	YES	YES	YES	YES	YES	YES	NO

Table 2. Prescription on specification methods. **Legend** (1): the NO answer derives from the unavailability of tools of a sufficiently consolidated level, that possess all the features required for the YES value; (2): the method is recommended for systems with SIL 1 or 2, while for systems with SIL 3 or 4 the method is acceptable at the current state of the art, but not strongly recommended; (3): the method is recommended only for systems with SIL 1.

4 A Success Story: The B Method

In the comparative case study from Section 3, the B method [1] was shown to be one of the strongest verification approaches. B has rigorous mathematical foundations and a well-developed underlying methodology, and is supported by a reasonably advanced toolset. A series of railway signaling products have benefited from the application of the B method in the design process. The success of B has had a major impact in the sector of railway signaling by influencing the definition of the EN50128 guidelines (see Section 2).

The B method targets software development from specification through refinement, down to implementation and automatic code generation, with verification at each stage. It includes a notation - Abstract Machines - for specifying a system: an Abstract Machine is defined as a set of states and a set of operations that modify the values of state variables; an invariant predicate is defined on states; for each operation a precondition and a postcondition are defined, so describing the effects of operations on state variables. It must be proved that when executing an operation in a state that satisfies both the precondition and the invariant, the state after the execution of this operation satisfies both the postcondition and the invariant. Moreover, at each refinement step if must be proved that the required safety properties of the system are preserved. So writing a specification produces a series of proof obligations that need to be discharged by formal proofs. The B method is accompanied by support tools, which include tools for the derivation of proof obligations, theorem provers, and Ada code generation tools.

The B method has been successfully applied to railway signaling systems, especially by Matra Transport and Alstom, mainly in France. The first application has been at the end of the eighties, concerning the SACEM system for the control of a line of Paris RER [19]. B was introduced while the project was already in progress, in order to ensure the two railway companies exploiting the line (SNCF and RATP) about the correctness of the design. B has been adopted for many later designs of similar systems by the same companies (especially Matra, which is now absorbed by Siemens). One of the most striking application has been for the Paris Météor metro line, which is in operation since October 1998. This line was designed to reach traffic of 40,000 passengers per hour with an interval between trains down to 85 sec. during peak hours. It is being managed by the Automatic Train Operation system developed by Matra, consisting of 86,000 lines of Ada (see [3]).

Of the 27,800 proven lemmas during the B development, around 90% were proven automatically by support tools, leaving around 2,000 lemmas to be interactively proven. Many errors were found during proof activities. By contrast, no further bugs were detected by the various testing activities that followed the B development. Moreover, no bugs have been reported since the line is in operation.

5 Classes of Railway Signaling Equipment

Railway signaling equipment can be roughly divided in two main categories. Train control systems guarantee safe speed and braking control for trains, while interlocking systems establish safe routes through the intricate layout of tracks and points within a railway station, yard or section. Several other minor signaling systems can be considered, which are often used to provide input for main signaling systems. Some of them may share the criticality level of main systems. Moreover, some signaling systems actually merge features of both categories above. For the purpose of our discussion, it will however be useful to concentrate on the nature of the two major classes identified.

5.1 Train Control Systems

A variety of train control systems exist, which may depend on the degree of authority over the driver (ranging from giving a mere support to the driver, to the completely automatic, driverless, systems), on different means to convey information to the train (either to the driver or to on-board equipment) and on the nature of this information. However, the basic general principle on which train speed control is based is common: the *braking curve* concept (Figure 1). The preceding train, or a fixed obstacle, defines a curve for the maximal safe speed of the train in any point of the line at a given time. The train has to maintain its own speed below the curve. Since the preceding train moves, the curve follows it, giving free headway to the train. The main challenge is to make sure that knowledge on board of the train regarding the curve is sufficiently accurate.

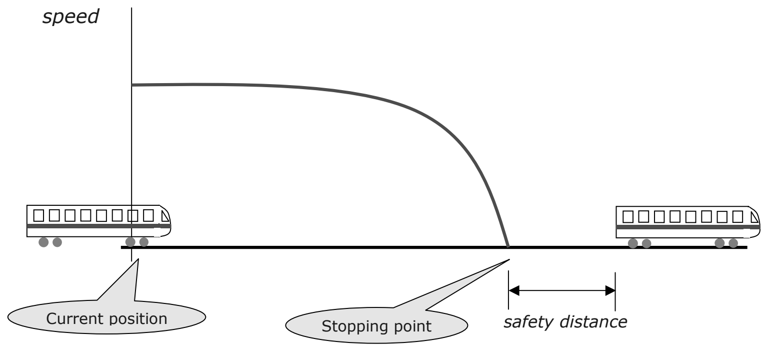


Fig. 1. The braking curve principle

When information is continuously exchanged between on-board and wayside computers, a guaranteed bandwidth is required, and safe communication protocols need to be used. Hence, in modern train control systems, complexity is shifting from basic safety rules to communication protocols.

The examples of SACEM and Météor systems show that formal methods can be applied to the entire system development. However, when dealing with main lines, matters get complicated by heterogeneity, compatibility and interoperability issues. The rolling stock is in general intended to be capable of running on differently equipped lines, with mixed freight/passenger traffic. Every national railway has its own tradition, especially for what concerns signaling rules and procedures. Before the advent of the open European market, every national company maintained contacts with (mostly national) signaling equipment providers, and the strict relation, typical of a protected market, between the first and the latter favored a nation-wide specific approach to signaling. This has had the effect that different countries in Europe have different train control systems, which require different on-board equipment. The new rules of the open railway market require instead that a train of a company is given equal access to all tracks in Europe. This means that a train should be interoperable. The only way to achieve this goal is currently to equip the cab with several versions of on-board equipment, one for each traversed nation, or to change the locomotive at each border.

For this reason the ERTMS/ETCS (European Rail traffic Management Systems / European Train Control System) project was launched, aiming at a single train control system for the future transeuropean railway network. The project plans, after the current initial test phase, to gradually install the ERTMS/ETCS equipment side by side to the traditional national equipment, also exploiting the three successive ERTMS/ETCS levels, referring to the increasing degree of information flowing from way-side to on-board equipment. In level 2 and 3, GSM-R (GSM radio communication specific to the railway industry) is adopted to continuously transfer information to the train on the status of the line ahead.

ERTMS/ETCS makes use of standardized components (European Vital Computer on board, Radio Block Center, Eurobalise, ...) and protocols (Euroradio), produced by a consortium of the main European signaling manufacturers, based on consolidated techniques. Specifications issued by ERTMS/ETCS [28] are structured as a natural language requirement document, including tables, state diagrams and sequence charts to add some formality.

Several formal modeling and verification studies have been conducted regarding ETCS protocols and components, starting from a model of ETCS given by coloured Petri nets [43], to the use of statecharts with the aim of proving safety properties by model checking an early version of the radio-based train control system [18], to the recent studies of real-time properties using stochastic Petri nets [47] or CSP-OZ-DC [30].

Formal methods have also been used by consortium companies in their development cycle. For example, Ansaldo, after having addressed with SDL and Message Sequence Charts the modeling and verification of the Radio Block Center [16], has given a formal specification of the Euroradio protocol by means of

UML State Diagrams, performing a verification by simulation following given scenarios expressed as UML Sequence Diagrams [27].

In the case of ETCS, the attention of the formal methods community has shifted from the consolidated train control logic (the braking curve principle), to the safety and real-time performance of radio-based control, which is going to be the sole mean by which the conditions of the track ahead are communicated to the train, since even signals will no more be present on the line.

5.2 Interlocking Systems

The control and management of a railway area consists of two separate tasks. First, control instructions for the track and points are devised in the logistic layer, which is usually managed by human experts. Second, it has to be guaranteed that the execution of control instructions does not jeopardize safety; that is, collisions and derailments have to be avoided. This is done by means of a so-called interlocking, which is a medium between the infrastructure at the one side, and the logistic layer and its interfaces on the other side.

An interlocking is an embedded system that controls pieces of equipment (like signals, points, track circuits, automatic blocks) so interconnected that their functions should be performed in proper sequence and for which interlocking rules are defined in order to guarantee safe operations. A simple example of an interlocking system is shown in Figure 2, taken from a real Italian interlocking system [20]. Line segments represent track segments in the infrastructure; some of them have track circuits, that is, sensors of the presence of a train, which are numbered inside circles, joints between segments represent points. Lollypop-like drawings represent signals of various type. Numbered labels are at- tached to each important part of a route. This example (a station consisting of a single track line) constitutes of eight allowed routes, two points, eight signals, six track circuits and two automatic blocks.

A route can be set free only if all points on the route have been correctly placed, and no train is present. The signals can be set to green only if the route in front is set to free. These sentences express two examples of generic principles that hold for every interlocking systems. Such rules aim at allowing only safe combinations of points positions, signals, etc., in order to avoid collisions. The signal indications, handled by the interlocking system, govern the correct use of the routes, authorizing the movement of trains. The rules usually enforce a predefined sequence of actions. For instance, issuing a route request command first triggers a check that all the track elements involved in the route are free. In that case, commands are issued for the positioning of points for that route and for locking the track elements. This phase may be followed by a global centralized control over the correct state of the commanded elements, after which the route is locked and signal indications for the route are set.

Note however that the generic rules expressed above need to be conjugated on every specific lay- out; for instance, the rules should be set for the route 1-3 in figure 1 taking into account point 1, track circuit 10, 11, I, and so on.

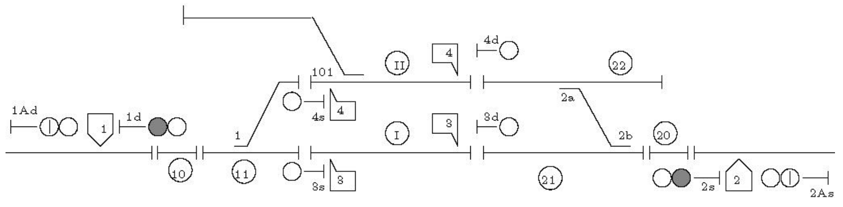


Fig. 2. The simplest track layout.

A route can be set free only if all points on the route have been correctly placed, and no train is present. The signals can be set to green only if the route in front is set to free. These sentences express two examples of generic principles that hold for every interlocking systems. Actually, the precise and complete set of such rules depends on the kind of railway station, yard or section (see, e.g., [31]), and also on national policies traditionally established by railway companies or regulatory boards. Since an interlocking system is safety-critical, the formalization of such rules is a top requirement.

In the traditional process adopted by many railway companies to develop relay-based interlocking systems, the generic principles were encoded into relay circuit templates. When a new interlocking plant was installed, these general principles had to be adapted to the particular layout of the section in use. The adaptation process was also guided by some more or less formalized rules. At the end of the process there was a diagram containing the command and control circuits for each logical or physical object in the station.

An example of this kind of diagram is shown in Figure 3; this diagram, taken from the same Italian interlocking, represents a circuit for the establishment of route 1-3 in Figure 2, taking into account point 1, track circuits 10, 11, I, and so on. The ladder diagram in Figure 3, expresses the fact that the energizing of relay CD_1_3 is dependent on many other relay contacts.

Such a circuit is generated based on templates supplied to help the engineers in the design of new stations. The templates are then associated to layout objects and replicated for each object. In a circuit template there are all the contacts needed to manage that kind of object; the only action to perform on it is the substitution of these generic contacts with the ones dictated by the layout. The structure of all diagrams related to routes is always the same, but the numbers and names of some contacts (serial or parallel) are different from one route to another.

The safety of old times, relay-based, interlocking systems was based on single fail-safe concepts, exploiting the intrinsic characteristics of relay technology. The introduction of computers in the control and command chain has subverted this approach to safety, since failure modes of computer-controlled equipment may be much more diverse and difficult to predict.

The first approach followed by some railway companies was to maintain the traditional and well-established relay-based principle diagram as the trusted

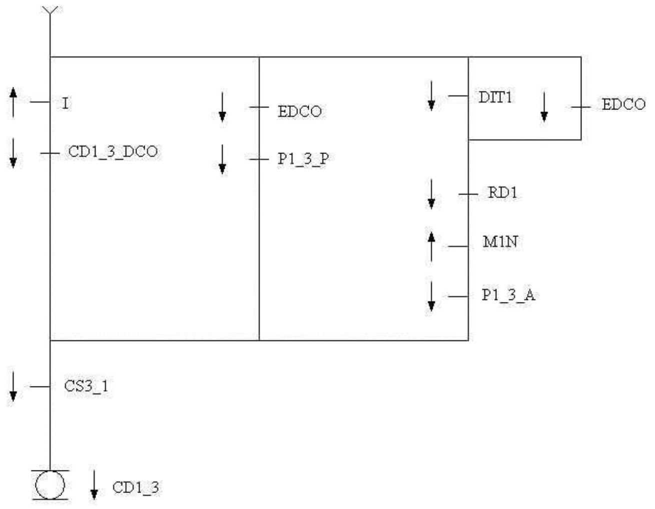


Fig. 3. An instantiated relay schema referred to route 1-3

source of information for computer-based interlocking developers, looking for conformance of the new interlocking systems to such sources, by means of costly and tedious, but possibly not exhaustive, testing. This approach has put on manufacturers' shoulders the burden of conceiving a family of interlocking products, together with means to instantiate the generic product by taking into account some proprietary, formalized version of principle diagrams suitable to be (more or less automatically) interpreted or compiled into running code, that has to be shown compliant to the trusted source. Actual approaches have varied from manufacturer to manufacturer as witnessed in [6, 34, 42, 25]. The development of computer-controlled interlocking systems has seen an increasing interest in the use of formal methods, due to their ability to precisely specify the logical rules that guarantee the safe establishment of routes. B notations (see Section 4) are not really suitable to express the interlocking logical rules, since these require the system states (logical variables, corresponding to the old relay states) to be accessed globally by many logical rules, while Abstract Machines encapsulate their state variables, which are accessible only via operations.

Rather than being adherent to old relay technology, what is needed is an innovative approach that encompasses a complete formalization of the whole system. Domain specific languages have been proposed for the formalization of interlockings [38, 44], the most prominent one being EURIS [4, 21], which will be discussed in Section 5.3. More recently, the possibility of using commercial support tools has pushed forward the use of general-purpose languages, as shown in Section 3, and a recent trend has indicated Statecharts (in their Statemate, Stateflow or UML state diagrams dialects) as a means for defining a standard formalization, see e.g. [2].

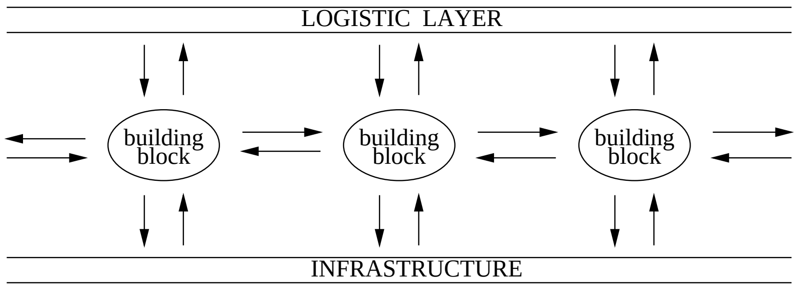
The latter trend has to do with the transition from the traditional protected market to the European open market, but on different grounds than for train control systems (see Section 5.1). With the previous national protected market, interlocking systems were developed by national manufacturers for the national railway company. Still, railway signaling is the responsibility of the national societies that are in charge of the railway infrastructure (e.g., RFF, RFI, ProRail) and not of the open market railway operators. On the other hand, traditionally national industries have been merged and reorganized in a few multi-national companies. They therefore have to merge different know-hows about railway signaling production, in order to unify their product lines. In this scenario, the strict collaboration between national railway companies and national railway industries, in which there was no need for a precise specification (every misunderstanding was resolved by phone) vanishes; there is greater reliance on precise contractual specifications to define the responsibility of each involved party.

The European railway community has come to realise that a drastic overhaul of current interlocking design is needed, for four main reasons. First, current methods to design interlockings, like SSI and VPI from Alstom and WESTRACE from Westinghouse Signals, are based on the earlier designs of interlockings using relays (i.e., electrical switches), and as a result do not fully exploit the additional capabilities of computer hardware and software. Second, due to their lack of modularity, current methods are not suitable to build interlocking systems for very large railway stations; dividing such stations into separate parts, which is the most common solution, causes undesirable communication overhead in current methods. Third, different European nations so far use different interlocking technologies, with raising costs due to the lack of standardization Fourth, formal methods cannot easily be integrated into current methods; see e.g. [36, 17, 23], for some work in this direction, mostly based on the adoption of model checking techniques for formal verification of interlocking systems. It is in this light that the Italian railway infrastructure agency has issued the document for software procurement discussed in Section 3.

Euro-Interlocking is a joint project by the main European railway companies and suppliers to develop both European functional requirements and standardized interfaces for interlocking systems. In this case, interoperability is not an issue, as interlocking systems do not (directly) communicate with trains. Standardization has the sole purpose to reduce costs, by means of standardized components and standardized interlocking rules. Inside Euro-Interlocking, the EIFFRA working group [40] focuses on textual requirements and requirement management tools such as Telelogic DOORS, together with model-based requirements. This is done by means of UML state diagrams and statecharts to describe the behavior, and OCL to describe properties of the interlocking systems. In this context, SNCF-RFF modeled their (relay-based) principle schemata using Statemate, producing 90 generic statecharts for interlocking elements [41]. These have been instantiated on an example medium-size station, to obtain 115 instances of 25 statecharts (out of the 90). Simulation by Statemate and visualization of scenarios by Waveforms were used to verify the correctness of the definition and instantiations.

5.3 EURIS

The restrictions that the interlocking logics imposes on the states of the system for different railway stations, yards and sections are reasonably consistent, depending mostly on the parameters of the autonomous elements, such as signals and points. Based on this observation, Peter Middelraad from the Dutch company ProRail evolved a modular specification method/language EURIS (European Railway Interlocking Specification) [4], to describe fully automated interlocking logics. EURIS assumes an object-oriented architecture, which consists of a collection of generic building blocks, representing the elements in the infrastructure such as signals and points, and of two clearly separated entities in the outside world, representing the logistic layer and the infrastructure. The building blocks, which together make up the interlocking logic, communicate with each other by means of data structures called telegrams. The building blocks can also exchange telegrams with the two entities that model the logistic layer and the infrastructure. This model can be depicted as follows.



To give an example, suppose that the logistic layer decides that a train should be moved via route R. This request is passed on to the interlocking layer, which attempts to claim route R; this mission is divided into smaller tasks, which are performed by exchanging telegrams between building blocks. If all building blocks concerned agree that route R can be established without jeopardizing safety, then the interlocking reserves this route, after which it passes on the necessary instructions to the infrastructure.

EURIS not only denotes a specification method, it is also the name for a graphically oriented imperative specification language that is based on this method. A so-called Logic and Sequence Chart (LSC) specifies a building block. Each LSC consists of the graphical representation of procedures, which can adapt and test the values of variables, and which can ultimately trigger the transmission of a telegram. Such telegrams can be received by neighboring building blocks, by the overlying logistic layer, and by the underlying infrastructure. Conversely, each building block can also receive telegrams from neighboring building blocks, from the logistic layer, and from the infrastructure. The communication channels between the building blocks, the logistic layer, the infrastructure, and the initial values of variables, are recorded outside the LSCs.

In EURIS, the heart of a specification defines the way that different kinds of building blocks handle incoming telegrams. Intuitively, these building blocks represent the separate elements in the infrastructure, such as signals and points. As soon as all types of building blocks have been specified in full detail, the specification of a particular railway area layout is constructed by simply connecting its separate building blocks in the appropriate manner. The object-oriented approach of EURIS allows design of interlockings for large railway stations without extra effort, and makes it possible to have different interlocking design patterns for different countries.

UniSpec [4] is a particular instance of the EURIS method, which has been developed by ProRail as a complete set of generic elements to compose interlocking A simulator enables animation of the behavior of a UniSpec specification. After designing a set of LSCs, the user can join instantiations of these LSCs according to the topology of a railway area. The result is checked for design rule errors and compiled, after which situations at the controlled railway area can be simulated via a graphical interface.

In a project funded by the Dutch company Holland Railconsult, researchers from Utrecht University formulated a formal operational semantics for EURIS [5], which, following the EURIS simulator, is based on a discrete time model. This semantics was presented in the setting of discrete-time process algebra. In a follow-up project, funded by ProRail, researchers from CWI in Amsterdam devised a textual variant of EURIS called LARIS [32], with the aim of improving the clarity of the graphical LSCs. Verification efforts of EURIS specifications were undertaken at CWI in Amsterdam. A prototype compiler from EURIS to μCRL [8] was implemented, and the EURIS specification of the Dutch station Woerden-Harmelen was tackled with the help of the μCRL toolset. Thus a symbolic version of the state space of this interlocking system has become available for analysis. The correctness of a EURIS specification of a (smaller) imaginary railway station was established using the μCRL toolset.

The verification effort concerning Woerden-Harmelen resulted in several advances in the realm of formal verification. Namely, the state space belonging to the interlocking system at this station is so large that new verification techniques had to be implemented for the μCRL toolset, to cope with such large state spaces. They are based on partial order reduction [13], distributed state space generation [9], and minimization of such a distributed state space [10–12].

The ownership of EURIS has shifted from ProRail to Siemens, with the aim of guaranteeing stronger commercial support and tool development. Currently EURIS is at the heart of the GRACE toolset of Siemens [39].

6 Conclusions

We have seen that the history of application of formal methods to railway signaling is not disjoint from the history of the organization of railways, which has undergone dramatic changes in the last decades, due to the advent of the European Community enforced open market. A shift towards behavioral, state-machine based formalisms has been witnessed, with more attention towards formalisms supported by commercial tools. Tools that give the ability of simulating and model-checking specifications, and of generating code from them will have an added value. But these tools may be very expensive, prohibitive for small companies that produce software for major ones or produce minor equipment which should anyway satisfy directives. Still some more time will pass before a clear satisfactory indication will emerge.

Formal methods for specification and verification are– slowly and with difficulties– reaching some appreciation and use in the industrial environment: there are many notations, methods, and (prototypal) tools originating from the academia, which however lack industrial strength in terms of tool stability, documentation and user support; on the other hand, there are very few technically sound methods and tools coming from industry.

International standards like EN50128 can have a positive role in promoting the adoption of systematic and technically sound development methods, but can also be technically outdated, obscure, ambiguous or too accommodating.

Thorough verification of complex, hard real-time systems is still infeasible in practice using the (industrial strength) tools. The verification technology is however rapidly evolving.

References

1. J.R. Abrial. *The B-Book*. Cambridge University Press, 1996.
2. M. Banci and A. Fantechi. Geographical vs. functional modelling by statecharts of interlocking systems. In *Proceedings 9th Workshop on Formal Methods for Industrial Critical Systems (FMICS'04)*, Linz, Volume 133 of *Electronic Notes in Computer Science*, pp. 3–19. Elsevier, 2005.
3. P. Behm, P. Benoit, A. Faivre, and J.M. Meynadier. Météor: A successful application of B in a large project. In *Proceedings World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, Toulouse, Volume 1708 of Lecture Notes in Computer Science, pp. 369–387. Springer, 1999.
4. J. Berger, P. Middelraad, and A. J. Smith. EURIS, European railway interlocking specification. In *Proceedings IRSE'93*, pp. 70–82. Institution of Railway Signal Engineers, 1993.
5. J.A. Bergstra, W.J. Fokkink, W.M.T. Mennen, and S.F.M. van Vlijmen. *Railway Logic via EURIS*. Quaestiones Infinitae XXII, Zeno Institute of Philosophy, Utrecht, 1997. In Dutch.
6. C. Bernardeschi, A. Fantechi, S. Gnesi, S. Larosa, G. Mongardi, and D. Romano. A formal verification environment for railway signaling system design. *Formal Methods in System Design*, 12(2):139–161, 1998.

7. D. Bjørner. New results and trends in formal techniques and tools for the development of software for transportation systems - A review. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, Budapest. L'Harmattan Hongrie, 2003.
8. S.C.C. Blom, W.J. Fokkink, J.F. Groote, I.A. van Langevelde, B. Lisser, and J.C. van de Pol. μCRL: a toolset for analysing algebraic specifications. In *Proceedings 13th Conference on Computer Aided Verification (CAV'01)*, Paris, Volume 2102 of Lecture Notes in Computer Science, pp. 250–254. Springer, 2001.
9. S.C.C. Blom, I.A. van Langevelde, and B. Lisser. Compressed and distributed file formats for labeled transition systems. In *Proceedings 2nd Workshop on Parallel and Distributed Model Checking (PDMC'03)*, Boulder, Volume 89 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
10. S.C.C. Blom and S. Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. In *Proceedings 1st Workshop on Parallel and Distributed Model Checking (PDMC'02)*, Brno, Volume 68(4) of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.
11. S.C.C. Blom and S. Orzan. Distributed state space minimization. In *Proceedings 8th Workshop on Formal Methods for Industrial Critical Systems (FMICS'03)*, Trondheim, Volume 80 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
12. S.C.C. Blom and S. Orzan. Distributed branching bisimulation reduction of state spaces. In *Proceedings 2nd Workshop on Parallel and Distributed Model Checking (PDMC'03)*, Boulder, Volume 89 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
13. S.C.C. Blom and J.C. van de Pol. State space reduction by proving confluence. In *Proceedings 14th Conference on Computer Aided Verification (CAV'02)*, Copenhagen, Volume 2404 of Lecture Notes in Computer Science, pp. 596–609. Springer, 2002.
14. G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.
15. F. Boussinot and R. de Simone. The Esterel language. Another look at real time programming. *Proceedings of the IEEE*, 79(9):1293–1304, 1991.
16. A. Chiappini, A. Cimatti, C. Porzia, G. Rotondo, R. Sebastiani, P. Traverso, and A. Villafiorita. Formal specification and development of a safety-critical train management system. In *Proceedings 18th Conference on Computer Safety, Reliability and Security (SAFECOMP'99)*, Toulouse, Volume 1698 of Lecture Notes in Computer Science 1698, pp. 410–419. Springer, 1999.
17. A. Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli, and P. Traverso. Formal verification of a railway interlocking system using model checking *Formal Aspects of Computing*, 10(4):361–380, 1998.
18. W. Damm and J. Klose. Verification of a radio-based signaling system using the STATEMATE verification environment. *Formal Methods in System Design*, 19(2):121–141, 2001.
19. C. DaSilva, B. Dehbonei, and F. Mejia. Formal specification in the development of industrial applications: Subway speed control system. In *Proceedings 5th IFIP Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'92)*, Perros-Guirec, pp. 199–213. North-Holland, 1993.
20. P.E. Debarbieri, F. Valdambrini, and E. Antonelli. A.C.E.I. Telecomandati per linee a semplice binario, schemi I0/19. CIFI Collana di testi per la preparazione agli esami di abilitazione, Quaderno 12, 1987.

21. F.J. van Dijk, W.J. Fokkink, G.P. Kolk, P.H.J. van de Ven, and S.F.M. van Vlijmen. EURIS, a specification method for distributed interlockings. In *Proceedings 17th Conference on Computer Safety, Reliability and Security (SAFECOMP'98)*, Heidelberg, Volume 1516 of Lecture Notes in Computer Science, pp. 296-305. Springer, 1998.
22. P. van Eijk, C.A. Vissers, and M. Diaz. *The Formal Description Technique LOTOS*. Elsevier, 1989.
23. C. Eisner. Using symbolic CTL model checking to verify the railway stations of Hoorn-Kersenboogerd and Heerhugowaard. *Software Tools for Technology Transfer*, 4(1):107–124, 2002.
24. J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL - Formal Object-oriented Language for Communicating Systems*. Prentice Hall, 1997.
25. L.H. Eriksson. Specifying Railway Interlocking Requirements for Practical Use, In *SAFECOMP'96 – Proceedings of the 15th International Conference on Computer Safety, Reliability and Security*, Springer-Verlag, 1996.
26. `http://www.ertms.com`.
27. R. Esposito, A. Lazzaro, P. Marmo, and A. Sanseviero. Formal verification of ERTMS Euroradio safety critical protocol. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, Budapest. L'Harmattan Hongrie, 2003.
28. ETCS Specifications, 2002. `http://etcs.uic.asso.fr/specifications.html`.
29. European Committee for Electrotechnical Standardization. EN 50128, Railway Applications Communications, Signaling and Processing Systems Software for Railway Control and Protection Systems, 2001.
30. J. Faber. Verifying real-time aspects of the European train control system. In *Proceedings 17th Nordic Workshop on Programming Theory (NWPT'05)*, Copenhagen, 2005.
31. W.J. Fokkink. Safety criteria for the vital processor interlocking at Hoorn-Kersenboogerd. In *Proceedings 5th Conference on Computers in Railways (COMPRAIL'96)*, Berlin, pp. 101–110. Computational Mechanics Publications, 1996.
32. W.J. Fokkink, J.F. Groote, M.J. Hollenberg, and S.F.M. van Vlijmen. *LARIS 1.0: LAnguage for Railway Interlocking Specifications*. CWI Publications Miscellaneous, Stichting Mathematisch Centrum, 2000.
33. U. Foschi, M. Giuliani, A. Morzenti, M. Pradella, and P. San Pietro. The role of formal methods in software procurement for the railway transportation industry. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, Budapest. L'Harmattan Hongrie, 2003.
34. B. Fringuelli, E. Lamma, P. Mello, and G. Santocchia. Knowledge-based technology for controlling railway stations. *IEEE Intelligent Systems*, 7(6):45–52, 1992.
35. C. Ghezzi, D. Mandrioli, and A. Morzenti. TRIO: A logic language for executable specifications of real-time systems. *Journal of Systems and Software*, 12(2):107–123, 1990.
36. J.F. Groote, J.W.C. Koorn, and S.F.M. van Vlijmen. The safety guaranteeing system at station Hoorn-Kersenboogerd. In *Proceedings 10th IEEE Conference on Computer Assurance (COMPASS'95)*, Gaithersburg, pp. 131-150. IEEE Computer Society Press, 1995.
37. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
38. A.E. Haxthausen and J. Peleska. Generation of executable railway control components from domain-specific descriptions. In *Proceedings 4th Symposium on Formal*

Methods for Railway Operation and Control Systems (FORMS'03), Budapest, pp. 83–90. L'Harmattan Hongrie, 2003.

39. B. Jung. Die Methode und Werkzeuge GRACE. In *Formale Techniken für die Eisenbahn-sicherung (FORMS'00)*, Fortschritt-Berichte VDI, Reihe 12, Nr. 441. VDI Verlag, 2000.
40. N.H. König and S. Einer. The Euro-Interlocking formalized functional requirements approach (EIFFRA). In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, Budapest. L'Harmattan Hongrie, 2003.
41. P. Le Bouar. Interlocking SNCF functional requirements description. Euro-Interlocking Project, Paris, May 2003.
42. G. LeGoff. Using synchronous languages for interlocking. In *First International Conference on Computer Application in Transportation Systems*, 1996.
43. M. Meyer zu Hörste and E. Schnieder. Formal modelling and simulation of train control systems using Petri nets. In *Proceedings World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, Toulouse, Volume 1709 of Lecture Notes in Computer Science, p. 1867. Springer, 1999.
44. M.J. Morley. Safety in railway signalling data: A behavioural analysis. In *Proceedings 6th Workshop on Higher Order Logic Theorem Proving and its Applications (HUG'93)*, Vancouver, Volume 740 of *Lecture Notes in Computer Science*, pp. 464–474. Springer, 1993.
45. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
46. J.M. Spivey. *Introducing Z: a Specification Language and its Formal Semantics*. Cambridge University Press, 1998.
47. A.Zimmermann and G. Hommel. Towards modeling and evaluation of ETCS real-time communication and operation. *Journal of Systems and Software*, 77(1):47–54, 2005.