Compositional Verification of Railway Interlockings: Comparison of Two Methods

Fantechi, Alessandro; Gori, Gloria; Haxthausen, Anne E.; Limbrée, Christophe

Published in:
Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification

Link to article, DOI:
10.1007/978-3-031-05814-1_1

Publication date:
2022

Document Version
Peer reviewed version

Citation (APA):
Fantechi, A., Gori, G., Haxthausen, A. E., & Limbrée, C. (2022). Compositional Verification of Railway Interlockings: Comparison of Two Methods. In *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification* (pp. 3-19). Springer. https://doi.org/10.1007/978-3-031-05814-1_1

Compositional verification of railway interlockings: comparison of two methods

Alessandro Fantechi[1], Gloria Gori[1], Anne E. Haxthausen[2], and Christophe Limbrée[3]

[1] University of Florence, Firenze, Italy
[2] DTU Compute, Technical University of Denmark, Lyngby, Denmark
[3] Belgian Railway Infrastructure Manager, Brussels, Belgium

Abstract. Formal verification of safety of interlocking systems and of their configuration on a specific track layout is conceptually an easy task for model checking. Systems that control large railway networks, however, are challenging due to state space explosion problems. A possible way out is to adopt a compositional approach that allows safety of a large system to be deduced from the formal verification of parts in which the system has been properly decomposed. Two different approaches have been proposed in this regard, differing for the decomposition assumptions and for the adopted compositional verification techniques. In this paper we compare the two approaches, discussing the differences, but also showing how the different concepts behind them are essentially equivalent, hence producing comparable benefits.

Keywords: Compositional Verification · Model Checking Railway · Railway Interlocking Systems

1 Introduction

Railway signalling is one of the domains in which formal methods have been applied in industry with multiple success stories since decades. In particular, *interlocking systems*, that control the train movements inside a railway network, called for a direct application of model checking, since required safety properties can be conveniently expressed in temporal logic. These systems need to be configured with *application data* that are closely related to the network layout in terms of tracks, points, signals, etc. Formal verification aims to verify both the generic algorithms for safe allocation of routes to trains, and the specific configuration for the network at hand, given by the application data. However, due to the high number of variables involved, automatic verification of sufficiently large networks typically incurs in combinatorial state space explosion [9].

State space explosion in model checking has been addressed by several techniques, e.g., adopting abstraction techniques, that preserve the validity of model checking results: the abstraction technique to be chosen typically depends on the nature of the system. Indeed, interlocking systems typically exhibit a high degree of *locality*: if we consider a typical safety property desired for an interlocking system, e.g., that the same track element shall not be reserved by more than one train at a time, it is likely that this property is not influenced by a train moving on a distant, or parallel, track element. Locality of a safety property can be exploited for verification purposes, so limiting the state space on which to verify it, by abstracting away the said "movement of distant trains". This principle has been exploited in [26] to define domain-oriented optimisation of the variable ordering in a BDD-based verification. Locality has been used also for *model slicing*, as suggested in [9, 14, 13], where only the portion of the model that has influence on the property to be verified (*cone of influence*) is considered: this allows for a more efficient verification of the single property, at the price of repeating the slicing and the verification for every property, and of separately checking that this abstraction preserves property satisfaction.

Locality also enables the adoption of a *compositional* approach that separately verifies portions of a network layout that are shown to be rather independent by the said locality principle. The residual dependency between the portions needs however to be properly addressed.

In [10, 18, 19, 8] a compositional verification approach based on dividing the network layout into two (or more) portions has been proposed. Extra track sections and signals are added at the border between two portions in order to abstract in one portion the behaviour of the other one.

A different approach [15–17] has addressed the same issue by resorting to the criteria for functional decomposition of interlocking systems as defined by Belgian railways; these criteria address the control of large networks by dividing the layout into subnetworks, each possibly controlled by separate interlocking systems. The cited study has considered the very same criteria as efficient as well as a basis for a compositional verification approach.

We were therefore interested in comparing the two compositional approaches. Indeed, they appear at a first glance quite different both for the definition of possible decomposition procedure and for the formal verification machinery adopted. The comparison presented in this paper shows in the end, instead, a high degree of similarity, that strengthens the confidence on the actual applicability of compositionality to attack formal verification of interlocking systems of large networks.

The paper is structured as follows: Sect. 2 introduces some necessary terminology for interlocking systems. With reference to the two different approaches, Sect. 3 describes "normal" formal verification of interlocking systems, while compositional reasoning is introduced in Sect. 4. Sect. 5 compares the two compositional approaches and discusses the differences. Sect. 6 introduces a case study, that has been useful to compare the methods, and which shows the advantages of the compositional methods. A related work section (Sect. 7) and a concluding section (Sect. 8) follow.

2 Background

In this section we briefly introduce the main notions of interlocking systems: we restrict to assumptions consistent with the European ETCS Level 2 resignalling program, for which we refer to [24, 25] for a more detailed introduction, and to Belgian signalling principles. Different terminology and assumptions are often used as well, as can be found in most of the literature on interlocking systems cited in this paper. We also take a simplified view, that is sufficient for our discussion, although ignoring some details of real systems.

A railway network consists of a number of track and track-side elements of different types, of which we limit to consider *linear sections, points, and signals.* Fig. 1 shows an example railway network layout, introducing the graphic representation of these elements:

– A *linear section*, identified with a magenta box in Fig. 1, is a track section with up to two neighbours: one in the *up* end, and one in the *down* end. For simplicity, in the following examples and figures, the *up* (*down*) direction is assumed to be the left-to-right (right-to-left) direction.
– A *point*, identified with a red box in Fig. 1, can have up to three neighbours: one at the *stem*, one at the *plus* end, and one at the *minus* end. The *stem* and *plus* ends form the straight (main) path, and the *stem* and *minus* ends form the branching (siding) path. A point can be switched between two positions: PLUS and MINUS, selecting the main or siding paths, respectively.
– Linear sections and points are collectively called *(train detection) sections*, as they are provided with train detection equipment used by the interlocking system to detect the presence of trains.
– Along each linear section, up to two *signals* (one for each direction) can be installed. Two signals, one for the up direction and one for the down direction, are indicated in Fig. 1 using two green boxes. A signal can only be seen in one direction and has two aspects (OPEN or CLOSED). In ETCS Level 2, signals are actually virtual and their aspect is communicated to the onboard computer via a radio network.
– A *route*, identified with a blue arrow in Fig. 1, is a path from a *source* signal to a *destination* signal in the given railway network.
– By *setting* a route we denote the process of allocating the resources – i.e., sections, points, and signals – for the route, and then *locking* it exclusively for only one train when the resources are allocated. When the train has left all the sections of the allocated route, the route is free again, to be allocated to another train.

Further examples of network layouts are deferred to Sect. 5.

Typical safety properties required of an interlocking system can be reduced to the following generic safety conditions:

1. **No collisions:** Two trains must never occupy the same track section at the same time.

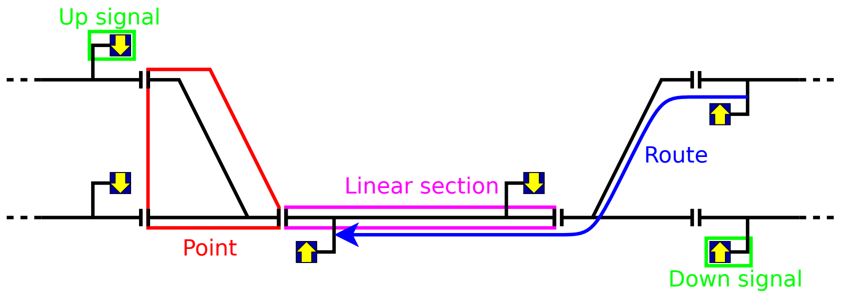


Fig. 1: Railway network: track and track-side elements.

2. **No derailments:** A point must not be switched, while being occupied by a train.

All required safety properties are expressed as *generic* conditions leading to *specific* conditions for each specific case of a network. The *No collisions* property is enforced by a mutually exclusive allocation of a route to a train asking for it. Notice that considering such typical safety properties, a route defines the maximal subset of elements whose status affects the safety property, that is, no element outside a route, or, at most, two conflicting routes, can affect a safety property for that route(s).

In this paper we focus on No collisions and No derailments safety properties. Other properties, e.g., liveness, can be proven using both methods.

3 Formal verification by model checking

Interlocking systems called for a direct application of model checking, since required safety properties can be conveniently expressed in temporal logic. The verification process based on model checking can be represented as in Fig. 2, where dashed boxes represent artefacts related to a specific network topology: typically, from the network layout a *control table* (aka *interlocking table* or *application data*) is derived, that contains information about routes, their sections, points and signals, their conditions for safe allocation, and their conflicts with other routes. From this data, a behavioural model in the form of a transition system is derived, according to realistic assumptions and principles of train movements, that also follow specific national regulations.

The two derivation steps of a model from the network layout can be automated, but the generation of the control table may ask for a manual intervention of signalling engineers to take into account specific physical constraints or other peculiarities [11].
The network layout guides the instantiation of generic safety properties as well. As usual, the model checker verifies whether the properties are satisfied by the model, returning a diagnostic counterexample in the case they are not.

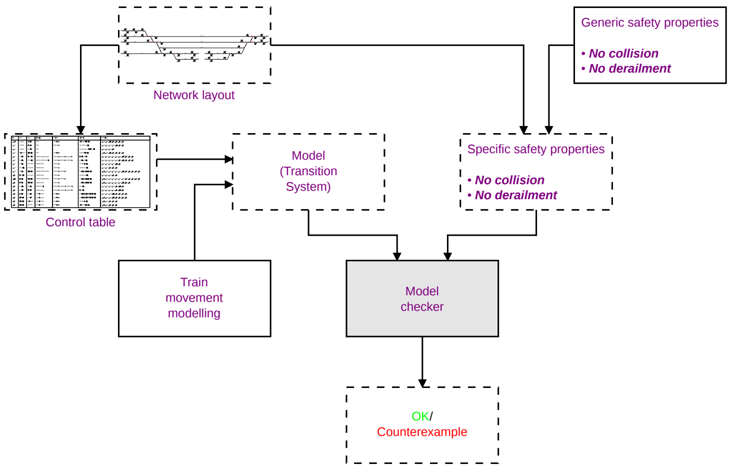


Fig. 2: Monolithic verification process.

In this paper we consider two verification approaches that implement, with a few differences, the process of Fig. 2.

3.1 The RobustRailS method

The RobustRailS verification method [22, 24, 25, 23] is based on a combination of formal methods and a domain-specific language (DSL) to express network diagrams and interlocking tables. A tool is provided by the RobustRailS environment to transform the DSL description into the transition system model and the required safety properties given as Linear Temporal Logic (LTL) formulae.

The RobustRailS tools can be used to verify the design of an interlocking system in the following steps:

1. A DSL specification of the configuration data (a network layout and its corresponding interlocking table) is constructed in the following order:
   (a) first the network layout,
   (b) and then the interlocking table (this is either done manually or generated automatically from the network layout).
2. The static checker [12] verifies whether the configuration data is statically well-formed according to the static semantics [24] of the DSL.
3. The generators instantiate a generic behavioural model and generic safety properties with the well-formed configuration data to generate the model input of the model checker and the safety properties.
4. The generated model instance is then checked against the generated properties by the bounded model checker, performing a k-induction proof.

The static checking in step (2) is intended to catch errors in the network layout and interlocking table, while the model checking in step (4) is intended to catch safety violations in the control algorithm of the instantiated model.

The tool chain associated with the method has been implemented using the RT-Tester framework [20, 21].

3.2 The Louvain method

The Louvain verification method [3] exploits a set of tools to automatically verify safety properties on a railway interlocking system model generated from the application data. The Louvain verification process can be described on the basis of Fig. 2 and consists of the following steps:

1. Generate a model of the interlocking based on its application data.
2. Generate a model of the train and the safety properties applicable to a specific network layout from the description of the topology of the network.
3. Combine the models of the interlocking with two instances of the train in a SMV model and verify the properties with nuXmv.

4 Compositional verification

Fig. 3 represents a generic compositional verification method, in which a complex network layout is divided into two or more subnetworks / components, and the previous process is applied to each of them, including control table generation, model generation, safety properties instantiation and model checking. A formal proof allows to extend the model checking results obtained on the subnetworks to the whole network: typically, if all the subnetworks satisfy the related safety properties, then the full network satisfies its own ones.
Within both the RobustRailS and the Louvain methods a compositional approach that instantiates the process of Fig. 3 has been developed.

4.1 The RobustRailS compositional method

In [18, 19, 8, 10] a method for performing compositional verification in connection with RobustRailS has been developed. It provides a general definition of allowed network cuts that divide a network into multiple subnetworks. Using such a network cut the compositional verification is done in the following steps:

1. Cut the network N into n subnetworks $N_1, \ldots, N_n$, applying allowed network cuts.
2. For $i = 1, \ldots, n$, use the RobustRailS tools verification steps described above to create a model m_i and properties ϕ_i and verify that m_i satisfies ϕ_i.

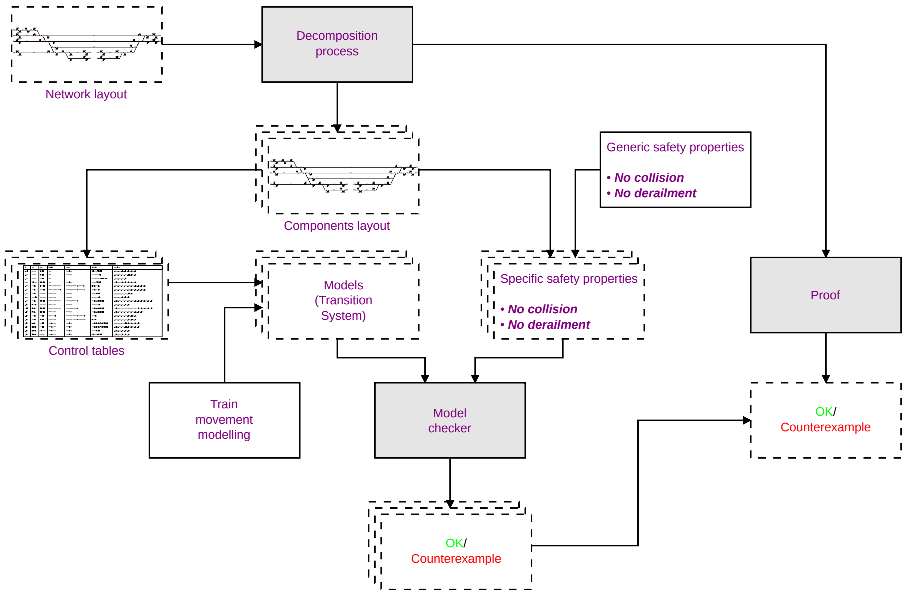


Fig. 3: Compositional verification process.

The identification of the points where to cut a network is manual[4], while a tool has been developed to generate the two subnetwork descriptions from the whole network and the identified cut points. Note that an interlocking system controlling a subnetwork (e.g. a station) is connected to the rest of the railway network by means of incoming/outcoming tracks, which are not under the control of the interlocking. The RobustRailS method assumes that a subnetwork includes at each of its connections with the outside a *border* section and a pair of signals: an *exit signal* which is not controlled by the interlocking, since the authority to exit the subnetwork area is not a responsibility of the interlocking, and an *entry signal* under the control of the interlocking. In the RobustRailS compositional method, a cut needs to add border track segment and signals in order to maintain the previously mentioned assumptions for the subnetworks as well (as shown in Fig. 4a, 4b). Under these assumptions, it is demonstrated that proving (by model checking) safety of both subnetworks implies the safety of the full network [10]. The “Proof” box in Fig. 3 is therefore in this case an a priori proof.

[4] The automatization of cut placements for RobustRailS tool is currently an undergoing activity.

4.2 The Louvain compositional method

A compositional verification process has been introduced in the Louvain method as well [15, 17]. The peculiar aspects of the Louvain compositional method can be summarised as:

1. The decomposition of a network into subnetworks is guided by five different decomposition patterns, inspired by the already adopted practice in Belgian railways to divide the interlocking logic for a large station in several zones;
2. These patterns basically define mutual exclusion interface variables that must be exchanged between the subnetworks in order to control the access from one to the other;
3. A specific tool, named *Component retriever*, takes the network topology, generates the components based on the said decomposition patterns, and specifies their interfaces and binding properties (contracts). Those specifications are expressed in the OCRA input language (Othello) [5];
4. Compositional verification is obtained by an *assume-guarantee* approach, supported by the OCRA framework [4]: the tool checks i) whether the contracts concerning bounded subnetworks (two by two) are coherent with the safety properties that their composite shall satisfy, and ii) that the exposed contracts of each subnetwork are satisfied by their implementation (model in SMV), according to the contracts-refinement proof system for component-based systems proposed in [6].
5. In a third verification step, safety properties are verified on the SMV model representing each subnetwork with the NuXmv model checker (Sect. 3.2).

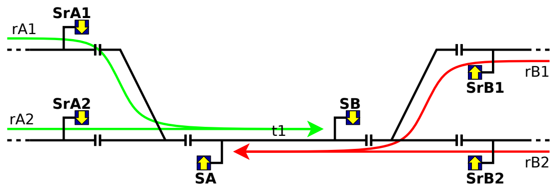
The "Proof" box in Fig. 3 in this case refers to the contract verification by OCRA, while the component verification activities are run employing nuXmv, taking advantage of k-liveness and ic3 algorithms [7, 2] in order to verify LTL properties on the components.

5 Comparison of the two methods

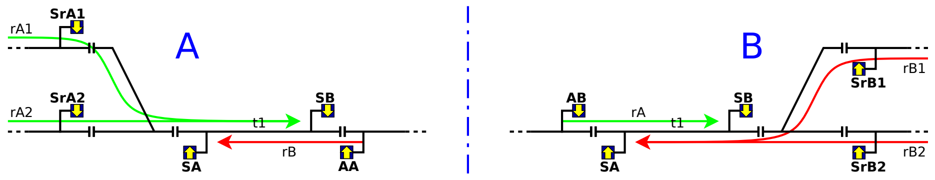
To discuss the details of each compositional method we refer to a simple example station shown in Fig. 4.

Both methods decompose the station into two components (A and B, respectively left and right) with a similar cut. The two models differ in the way the interactions between the components are managed. On one hand, RobustRailS adds linear sections and signals to abstract the other part of the network. On the other hand, Louvain retrieves and uses mutual exclusion variables (called `BSP` variables) already defined in the interlockings in order to define binding properties between subnetworks.

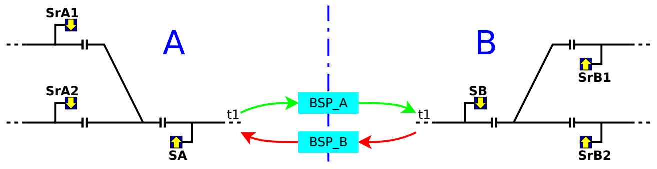
Let us consider the simple network of Fig. 4a, where the drawing represents the network layout in terms of tracks, points and signals. Two tracks converge from the left on a single track (up direction), and symmetrically from the right (down direction).



(a) Example station layout.



(b) RobustRailS cut implementation.



(c) Louvain cut implementation.

Fig. 4: Cut methods.

The coloured arrows represent the routes that make possible to reach the central track $t1$ from the left and the right. Obviously, routes $rA1$ and $rA2$ are in conflict with routes $rB1$ and $rB2$ to gain access to the central track. The No-collision property reads as “No two trains can enter track $t1$ together” and $t1$ is actually a shared resource with mutually exclusive access by the trains. The interlocking system guarantees the property by granting only one of the conflicting routes to be allocated to a train, and communicating this to the trains by means of signals at the beginning of the routes ($SrA1, SrA2, SrB1, SrB2$ for $rA1, rA2, rB1, rB2$ respectively).

When applying a decomposition that cuts the network into two symmetrical halves A and B, the two halves are managed as if they were controlled by two different interlocking systems. Hence, mutual exclusion on $t1$ is distributed among the two halves.

The definition of cut[5] given by the RobustRailS method includes $t1$ in both subnetworks and adds the signals AA and AB to A and B subnetworks, respectively (Fig. 4b); focusing on A, the extra signal adds a route rB from AA to

[5] Note that the operated cut is the one defined in [19], which slightly differs from the *single cut* defined in [10], since the $t1$ section is present in both subnetworks. The compositionality proof of [10] covers this case as well.

SA, that abstracts all the previously incoming routes in down direction in the full network, namely $rB1$ and $rB2$. The same holds symmetrically for B.

The Louvain method instead addresses the problem by recurring to a mechanism already adopted for communication between interlocking systems controlling shared tracks, that is, interface variables. In Fig. 4c it can be seen that two interface variables BSP_A and BSP_B exist: the former communicates to the B half the reservation of $t1$ as seen from A, and vice versa for the latter.

In the RobustRailS method, in order to open signal $SrA1$ in the A subnetwork, the route $rA1$ (from $SrA1$ to SB) should successfully go through the states ALLOCATING and LOCKED, which is not possible if the conflicting route rB from AA to SA (that abstracts $rB1$ and $rB2$) is in one of the states ALLOCATING, LOCKED or OCCUPIED. This means that if $SrA1$ is OPEN, AA is CLOSED (as AA being OPEN would require that the conflicting route rB is LOCKED). This is guaranteed by model checking the A subnetwork. The same occurs symmetrically for the subnetwork B.

In the Louvain method, the two halves have each a copy of the BSP variables. The contract between the two halves is that whenever the A half allocates a route to $t1$ (such as $rA1$), opening the corresponding signal $SrA1$, the output variable BSP_A is false and the input variable BSP_B is true (B is not allowing a train to $t1$), and vice versa for the half B. Each subnetwork is then individually checked to guarantee this property. The proper implementation of the shared BSP variables in each subnetwork (model) prevents train collisions on the shared track $t1$.

We can therefore observe that in the RobustRailS method, the added signals AA and AB play the role of the variables BSP_A and BSP_B of the Louvain method, respectively.

We can conclude therefore that the two methods are equivalent for the considered cut[6]. Similar arguments can be used to show that this holds also for the other four decomposition patterns considered by the Louvain method. We claim that different decomposition patterns, addressed by the RobustRailS method, that have not been considered by the Louvain one, could be expressed by means of interface variables and contracts according to the latter by properly mimicking the added RobustRailS signals with interface mutual exclusion variables.

The two verification methods are different in the sense that while the Louvain one builds on the adoption of established compositional model checking techniques and tools, applied to the specific problem, the RobustRailS approach has been tailored in its very definition to the specific problem, hence it is a domain dependent solution: this is apparent in the fact that in the former contracts are established for each interface between two components, while in the latter the interface models are built by adding extra railway elements.

6 Case study: La Louvière-Sud

We develop further our comparison by applying the two methods on a common case study, with the main aim to confirm the advantages of compositionality in

[6] The formal equivalence of the two presented methods is out of scope of this paper.

both frameworks. Our case study concerns a real railway network: a portion of La Louvière-Sud in Belgium, which was already decomposed into three subnetworks, namely LVR1, LVR7 and LVR9 represented in Fig. 5[7].

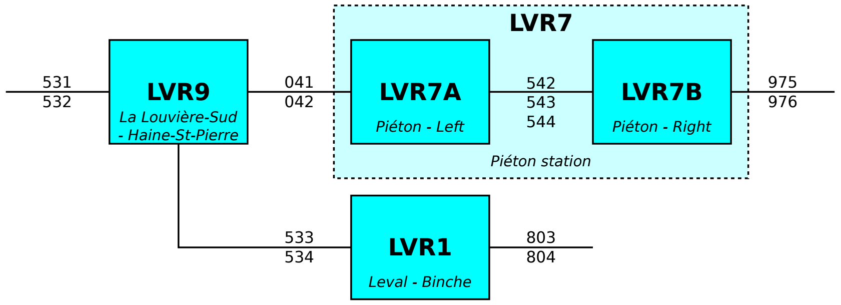


Fig. 5: La Louvière-Sud: topology of the verified components.

LVR1 and LVR9 are small and have a limited number of routes (18), so it is expected that for both methods they can be verified without recurring to decomposition. On the other hand, LVR7, the Piéton station represented in Fig. 6, has three main line platforms, a marshalling yard with two tracks, and contains many routes. So we will investigate how a decomposition of LVR7 can help the verification.

In the following subsection we describe how LVR7 can be decomposed using the two methods.

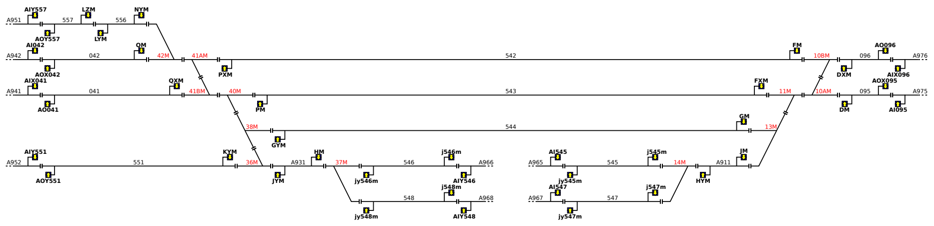
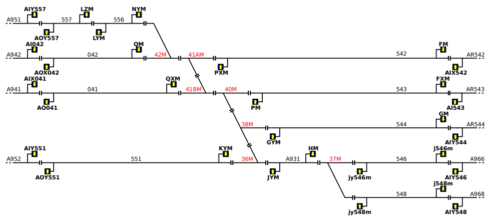


Fig. 6: Piéton station scheme represented according to the RobustRailS conventions: sections (plain label), points (red label) and signals (bold label). In particular the scheme includes the pair of signals at the borders.

[7] The models are available at https://github.com/gorigloria/compositionalverificationmodels


(a) Piéton left component.

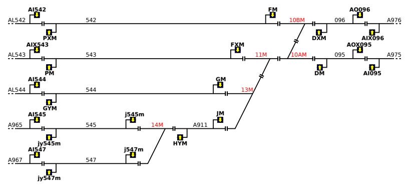

(b) Piéton right component.

Fig. 7: RobustRailS method: decomposition of Piéton station into two parts.

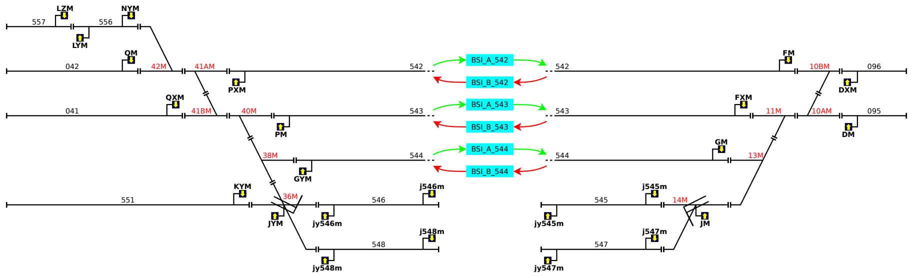


Fig. 8: Louvain method: decomposition of Piéton station into two parts.

6.1 Decomposition of LVR7 - Piéton station

Fig. 6 shows the Piéton station scheme represented according to the RobustRailS conventions.

Fig. 7 and Fig. 8 show the two subnetworks, obtained using the two decomposition methods. Note that they are slightly different. In the RobustRailS model, the signals added to abstract respectively the right (the left) component are AIX542, AI543, AIY544 (AI542, AIX543, AI544). Fig. 8 shows the decomposition according to the Louvain method, that was part of the work for the Christophe Limbrée's PhD thesis [17]. Note that only one kind of interface variable appears, out of the five kinds identified in the Louvain method [17]. The cut is, in fact, managed using the *BSI* mutual exclusion variables. The network modelled with the Louvain method relies on extensions of track-side equipment previously described in Sect. 2: in particular, the network has points with more than one branch and it uses *sectioning points*, i.e., points with an associated signal (see points 36M and 14M in Fig. 8). These extensions have been modelled with the RobustRailS methods as follows:

- Points with more than one branch have been splitted into multiple points;
- Sectioning points have been modelled as follows: 1) sectioning points have been treated as simple points; 2) an additional linear section has been placed adjacent to each sectioning point; 3) two signals, one for each direction, have been added to the additional linear section.

Another difference between the two methods is on the definition of routes. In the RobustRailS framework, every route starts at a signal and ends at the following one. This implies that no intermediate signal in the same travel direction is crossed in any given route. In the Louvain model, a route starts at a signal and ends on the destination track segment without crossing intermediate signals.

In the following subsections we report the experimentation results. The experiments were executed on a server with the following system specifications: Intel(R) Xeon(R) CPU E5-1650 @ 3.6GHz, 125GB RAM, and running Linux 4.4.0-47.x86_64 kernel. The execution time was limited to 1 day in order to fit with typically industrially acceptable times.

6.2 Verification results using the RobustRailS method

We have applied the RobustRailS method on LVR7 and its decomposed subnetworks as well as on LVR1 and LVR9. The RobustRailS control table generator allows for two options, one of which enforces the so-called *flank protection*, in which points and signals not belonging to the route are properly set in order to avoid hostile train movements into the route at an incident point. Table 1 reports the results when flank protection is chosen for all the modelled components. It can be seen that all networks were verifiable and that the time for verifying both LVR7A and LVR7B is around three times faster than that for LVR7, while the max needed memory usage (2083 MB) is around a third.

Table 1: Verification of the models for LVR1, LVR9, and LVR7 and LVR7's decomposed networks LVR7A and LVR7B using the RobustRailS tools.

ID	Name	Routes	Time (s)	Memory (MB)
LVR7	Piéton	48	2387	5467
LVR7A	Piéton - Left	30	670	2083
LVR7B	Piéton - Right	18	108	846
LVR1	Leval - Binche	18	38	413
LVR9	La Louvière-Sud - Haine-St-Pierre	18	33	415

6.3 Verification results using the Louvain method

Table 2 contains the verification metrics obtained by the OCRA/nuXmv tools for all the networks, with the same server used for RobustRailS experiments. The models of LVR1, LVR9, LVR7A and LVR7B were, as expected, verifiable, but the verification of the monolithic model of Piéton (LVR7) had to be stopped after one day, which is the maximal time considered to be feasible. The sum of the verification times of the models of the two decomposed networks is 23.210 s $\sim$ 6.5 hours, which shows that the decomposition not only made the verification feasible, but also fast (compared to more than one day). We highlight the small amount of memory occupied by the verification tasks.

Table 2: Verification of the models for LVR1, LVR9 and LVR7 and LVR7's decomposed networks LVR7A and LVR7B using the Louvain method.

ID	Name	Routes	Time (s)			Memory (MB)
			OCRA	nuXmv	Total	
LVR7	Piéton	48	Not feasible			-
LVR7A	Piéton - Left	30	15673	1997	17670	152
LVR7B	Piéton - Right	18	4791	749	5540	125
LVR1	Leval - Binche	18	287	245	532	48
LVR9	La Louvière-Sud - Haine-St-Pierre	18	3407	59	3466	81

6.4 Discussion

The shown performance figures are not meant to support an efficiency comparison between the two methods: indeed the actual verification performance depends on many factors that differ in the two methods. However, the figures on models LVR7, LVR7A and LVR7B clearly show the advantages given by compositional verification in both methods. Moreover, one interesting thing can be observed if we compare the verification times for all the networks: while the ones by the Louvain tools is generally 15-50 times longer than those by the RobustRailS tools, the time for the LVR9 component is of the order of 100 times longer. This can be explained by the fact that LVR9 is a simple junction that has few routes and a low internal complexity, but that connects with different components through several interfaces: the Louvain method requires to separately check the component w.r.t. all contracts related to the interfaces; the RobustRailS method takes the proof of compliance between components as granted once for all – when the added border sections and signals comply with the standard format (compliance assured by the static analysis engine included in RT-Tester).

Another main difference between the two methods is on the decomposition technique. On one hand, cuts are manually applied to a network for the RobustRailS method. On the other hand, Louvain method exploits automatic decomposition starting from the existing description of the network layout, which in turn is automatically generated from their application data. The automatic decomposition performed by Louvain method requires a couple of minutes, hence it can be neglected. The manual decomposition performed by RobustRailS method requires more time, but it is limited and feasible as the number of rules for cuts is low. Furthermore, for RobustRailS method, the implementation of automatic cuts is currently in progress.

7 Related work

We have already reported in the introduction how *locality* exhibited by interlocking systems has been exploited in different approaches aimed to optimise verification by model checking of large station layouts [26, 9, 14, 13, 1]. Still, in those approaches the verification process considers the full interlocking system defined over the full station layout.

The two approaches discussed in this paper are, at the best of our knowledge, the only ones that address verification of interlocking of large networks by decomposing the layout in smaller components and formally deduce safety of the whole from the safety verification of the parts.

Regarding a comparison of different formal verification methods of interlocking systems, not addressing compositionality, we can cite [11].

8 Conclusions

We have compared two different compositional approaches to address state space explosion in formal verification of railway interlocking systems: the RobustRailS compositional method and the Louvain compositional method. We made a comparison of methodological elements at a conceptual level rather than comparing concrete performance metrics of the two methods as they use different verification tools. The comparison revealed that different concepts behind the two methods are essentially equivalent when it comes to the division of the network of an interlocking system into two networks and the creation of interlocking models for these and their interfaces. However, the two methods are different in the sense that while the Louvain one builds on the adoption of established compositional model checking techniques and tools, applied to the specific problem, the RobustRailS approach has been tailored in its very definition to the specific problem, hence it is a domain dependent solution: this is apparent in the fact that in the former contracts are established for each interface between two components, while in the latter the interfaces are built by adding extra railway elements. A major difference between the two verification methods is also the amount of generated proof obligations: in both approaches, one must perform component verification (prove safety of the interlocking models for the two decomposed networks), however, for the Louvain compositional method there are additional verification obligations: the verification of the contracts, i.e. the verification that each component satisfies all contracts related to its interfaces. For the RobustRailS compositional method the soundness of the component verification has been proved a priori (once-and-for-all). A case study demonstrated that both methods had great benefits in terms of addressing state space explosion.

A further comparison of the RobustRailS method with the Louvain one could be done by extending the reasoning shown in this paper to the other four kinds of interface variable adopted in [17]. This extension is straightforward, due to the similarity of the different cases, but the detailed study is left to future work.

As a final remark, we observe that verification of interlocking systems in the end boils down to mutual exclusion verification. Operating a cut in the network typically distributes the mutual exclusion mechanisms over two or even more components, whether such decomposition is physical, to exploit the advantages of distributed computing, or logical, to remain in the low ends of the exponential state space explosion in verification. To this respect, the presented reasonings may reveal useful in other domains where some notion of distributed mutual exclusion may help verification of large systems.

Acknowledgements

The authors wish to thank Jan Peleska and Linh H. Vu with whom Anne Haxthausen developed the RobustRailS verification method and tools, and Hugo D. Macedo, who collaborated in the initial work of Anne Haxthausen and Alessandro Fantechi on the RobustRailS compositional approach.

References

1. Bonacchi, A., Fantechi, A., Bacherini, S., Tempestini, M.: Validation process for railway interlocking systems. Science of Computer Programming **128**, 2–21 (2016)
2. Bradley, A.R.: SAT-Based Model Checking without Unrolling. In: Proc. VMCAI 2011, Austin, TX, USA. LNCS, vol. 6538, pp. 70–87. Springer (2011)
3. Busard, S., Cappart, Q., Limbrée, C., Pecheur, C., Schaus, P.: Verification of railway interlocking systems. In: Proc. ESSS 2015, Oslo, Norway, June 22, 2015. EPTCS, vol. 184, pp. 19–31. Open Publishing Association (2015)
4. Cimatti, A., Dorigatti, M., Tonetta, S.: OCRA: A tool for checking the refinement of temporal contracts. In: 28th IEEE/ACM International Conference on Automated Software Engineering, Silicon Valley, CA, USA, November 11-15, 2013. pp. 702–705. IEEE (2013)
5. Cimatti, A., Tonetta, S.: A Property-Based Proof System for Contract-Based Design. In: 38th Euromicro Conference on Software Engineering and Advanced Applications. pp. 21–28. IEEE (2012)
6. Cimatti, A., Tonetta, S.: Contracts-refinement proof system for component-based embedded systems. Science of Computer Programming) **97**, 333–348 (2015)
7. Claessen, K., Sörensson, N.: A liveness checking algorithm that counts. In: Formal Methods in Computer-Aided Design, FMCAD 2012, Cambridge, UK, October 22-25, 2012. pp. 52–59. IEEE (2012)
8. Fantechi, A., Haxthausen, A.E., Macedo, H.D.: Compositional Verification of Interlocking Systems for Large Stations. In: SEFM 2017 - 15th Software Engineering and Formal Methods, Trento, Italy, September 4-8, 2017. LNCS, vol. 10469, pp. 236–252. Springer (2017)
9. Ferrari, A., Magnani, G., Grasso, D., Fantechi, A.: Model Checking Interlocking Control Tables. In: FORMS/FORMAT 2010. pp. 107–115. Springer (2010)
10. Haxthausen, A.E., Fantechi, A.: Compositional verification of railway interlocking systems. Submitted for publication (2021)
11. Haxthausen, A.E., Nguyen, H.N., Roggenbach, M.: Comparing Formal Verification Approaches of Interlocking Systems. In: Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification. LNCS, vol. 9707, pp. 160–177. Springer (2016)
12. Haxthausen, A.E., Østergaard, P.H.: On the Use of Static Checking in the Verification of Interlocking Systems. In: ISoLA 2016, Part II. LNCS, vol. 9953, pp. 266–278. Springer (2016)
13. James, P., Möller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Decomposing scheme plans to manage verification complexity. In: FORMS/FORMAT 2014. pp. 210–220. Institute for Traffic Safety and Automation Engineering, Technische Univ. Braunschweig (2014)

14. James, P., Lawrence, A., Möller, F., Roggenbach, M., Seisenberger, M., Setzer, A., Kanso, K., Chadwick, S.: Verification of Solid State Interlocking Programs. In: SEFM 2013 Collocated Workshops. Revised Selected Papers. vol. 8368, pp. 253–268. Springer (2014)
15. Limbrée, C., Cappart, Q., Pecheur, C., Tonetta, S.: Verification of Railway Interlocking - Compositional Approach with OCRA. In: RSSRail 2016, Paris, France, June 28-30, 2016, Proceedings. LNCS, vol. 9707, pp. 134–149. Springer (2016)
16. Limbrée, C., Pecheur, C.: A Framework for the Formal Verification of Networks of Railway Interlockings - Application to the Belgian Railway. Electron. Commun. Eur. Assoc. Softw. Sci. Technol. **76** (2018)
17. Limbrée, C.: Formal verification of railway interlocking systems. Ph.D. thesis, UCL Louvain (2019)
18. Macedo, H.D., Haxthausen, A.E., Fantechi, A.: Compositional Verification of Multi-Station Interlocking Systems. In: Proc. ISoLA 2016. LNCS, vol. 9953. Springer (2016)
19. Macedo, H.D., Fantechi, A., Haxthausen, A.E.: Compositional Model Checking of Interlocking Systems for Lines with Multiple Stations. In: Proc. NFM 2017. pp. 146–162. Springer (2017)
20. Peleska, J.: Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. In: 8th Workshop on Model-Based Testing, Rome, Italy. vol. 111, pp. 3–28. Open Publishing Association (2013)
21. Verified Systems International GmbH: RT-Tester Model-Based Test Case and Test Data Generator - RTT-MBT - User Manual (2013), req. at http://www.verified.de
22. Vu, L.H., Haxthausen, A.E., Peleska, J.: A Domain-Specific Language for Railway Interlocking Systems. In: FORMS/FORMAT 2014. pp. 200–209. Institute for Traffic Safety and Automation Engineering, Technische Universität Braunschweig (2014)
23. Vu, L.H., Haxthausen, A.E., Peleska, J.: A domain-specific language for generic interlocking models and their properties. In: Proc. RSSRail 2017, Pistoia, Italy, November 14-16, 2017, Proc. LNCS, vol. 10598, pp. 99–115. Springer (2017)
24. Vu, L.H.: Formal Development and Verification of Railway Control Systems - In the context of ERTMS/ETCS Level 2. Ph.D. thesis, Technical University of Denmark, DTU Compute (2015)
25. Vu, L.H., Haxthausen, A.E., Peleska, J.: Formal modelling and verification of interlocking systems featuring sequential release. Science of Computer Programming **133, Part 2**, 91–115 (2017)
26. Winter, K.: Optimising Ordering Strategies for Symbolic Model Checking of Railway Interlockings. In: ISoLA 2012. vol. 7610, pp. 246–260. Springer (2012)