






Decomposing the Verification of Interlocking Systems

Anne E. Haxthausen¹ , Alessandro Fantechi² , and Gloria Gori² 

¹ DTU Compute, Technical University of Denmark, Lyngby, Denmark
aeha@dtu.dk

² University of Florence, Firenze, Italy
{alessandro.fantechi, gloria.gori}@unifi.it

Abstract. This paper considers model checking the safety for members of a product line of railway interlocking systems, where an actual interlocking system is modelled as an instance of a generic model configured over the network under its control. For models over large networks it is a well-known problem that model checking may fail due to state space explosion. The RobustRailS tools that combine inductive reasoning with SMT solving using Jan Peleska’s powerful RT-Tester tool suite have pushed considerably the limits of the size of networks that can be handled. To further push these limits, we have proposed a compositional method that can be combined with RobustRailS to reduce the size of networks to be model checked: the idea is to divide the network of the system to be verified into two sub-networks and then model check the model instances for these sub-networks instead of that for the full network. In this paper we propose a strategy for applying such network divisions repeatedly to achieve a fine granularity decomposition of a given network into a number of small sub-networks. Under certain conditions, these sub-networks all belong to a library of pre-verified elementary networks, so model checking of the sub-networks is no longer needed.

Keywords: Formal Methods · Model Checking · Compositional Verification · Interlocking Systems

1 Introduction

Formal methods have successfully been applied to development and verification of railway systems [3, 5, 6]. In particular, it has been popular to use model checking techniques for formal verification of *interlocking systems* (controlling train movements inside a railway network) as these are fully automated. Interlocking systems are configured with *application data* that reflect the elements and topology of the railway network layout. Hence, formal verification aims to verify both the *generic application* with its algorithms for safe allocation of routes to trains, and the *specific application* produced by the configuration with application data for the network under control.

Model checking is subject to *state space explosion*, which limits scalability of the approach so that automatic verification of interlocking systems for large networks is demanding in terms of computing resources, and may even fail [4].

Abstraction techniques have typically been adopted to limit state space explosion in model checking: abstraction should preserve the desired properties, hence the adopted abstraction technique should be defined specifically for the kind of system and properties under examination. For interlocking systems, a convenient abstraction can be based on the *locality* principle: properties concerning the allocation of a route to a train are typically not influenced by train movements over networks elements that are distant from, and not interfering with, the considered route. Locality of a safety property can be used to limit the state space by abstracting away such “distant movements”. In [26] this principle supports domain-oriented optimisation of the variable ordering in a BDD-based verification; it also enables property-directed *model slicing*, ([4, 10, 11]), in which verification is performed only over the portion of the model that concerns the property of interest (*cone of influence*), allowing for an efficient verification of a property, but requiring to perform slicing and verification for every property (plus checking that that slicing preserves the related property).

It has also been suggested to use *bounded model checking* to perform *k-induction* proofs of safety properties expressed as state invariants to avoid exploring the whole state space. In the RobustRailS verification tools [25] for interlocking systems this technique was implemented using the powerful SMT-based bounded model checker of Jan Peleska’s RT-Tester tool¹; this made it possible to considerably push the bounds of the size of networks that can be verified without state space explosion [25].

Locality has also enabled our proposal of a *compositional* approach for addressing verification for very large networks: the idea is to divide the network to be verified into two (or more) sub-networks and then model check the model instances for these sub-networks instead of model checking the model instance of the full network [2, 8, 15, 16]. For model checking, we use the RobustRailS verification tools. The soundness result for compositional safety verification given in [8] guarantees that, when properly cutting a network, proving safety for the sub-networks suffices to prove safety for the full network. In this way, the task of proving safety for a large network can be reduced to the task of verifying safety for sub-networks of a size manageable by the model checker.

The idea of compositional verification is also shared by the approach described in [12–14]. This approach that is based on the criteria of functional decomposition of interlocking systems defined by the Belgian railways in order to deal with the control of large networks by dividing the network into sub-networks, each possibly controlled by separate interlocking systems. A comparison of this approach with ours is presented in [1]. Indeed, it appears that decomposition of a network in this approach is grounded on pragmatic domain-related criteria, while our approach is more general. Furthermore, this approach uses an

¹ <https://www.verified.de/products/rt-tester/>.

assume-guarantee approach for verification which requires not only verification for the sub-networks as in our approach, but also verification of contracts.

The question of where to divide a network during compositional verification has triggered the contribution of this paper: an iterative decomposition strategy to achieve a fine granularity decomposition of a network into a number of small sub-networks, that under certain conditions belong to a library of pre-verified elementary networks. The soundness result for compositional safety verification guarantees that safety for the full network is given by the pre-verified safety of sub-networks. Therefore, to verify a network, it is in principle no more needed to run a model checker, independently of the size of the network, if specific network conditions are met.

The paper is structured as follows: First, in Sects. 2 and 3, short descriptions of the RT-Tester tool suite and of the RobustRailS verification method, built on top of RT-Tester, are given. Then, in Sect. 4, our compositional method using the RobustRailS tools is presented and a strategy for performing decomposition is discussed. The latter is the main, novel contribution of this paper. Section 5 draws some conclusions and ideas for future work.

2 The RT-Tester Tool Suite

In 1998 Jan Peleska and Cornelia Zahlten founded the company *Verified Systems International GmbH*, and Jan has been head of Research & Development in the company since then. The company provides tools and services in the field of safety-critical system development, verification, validation and test, and has a wide variety of customers including Siemens, Airbus and its suppliers. Verified's flagship product is *RT-Tester*², a very comprehensive model-based test automation tool suite for automatic test generation, test execution and real-time test evaluation. RT-Tester can not only be used for testing (see e.g. [19]), but also for bounded model checking (BMC) of which we will give an example in next section. RT-Tester's automation capabilities are discussed in [18], and special test case generation strategies implemented in RT-Tester are described in [9]. In 2015, the company was awarded the runner-up trophy of the EU Innovation Radar Prize due to the special testing strategy that was developed by Jan Peleska and Wen-ling Huang.

3 The RobustRailS Verification Method and Tools

In the RobustRailS research project³ that was accompanying the Danish signalling programme on a scientific level in 2012–2017, a formal method with tools support for automated, formal verification of railway interlocking systems was developed [22–25] by Linh Hong Vu under supervision of Jan Peleska and Anne Haxthausen. This section gives a short description of the RobustRailS method and tools.

² <https://www.verified.de/products/rt-tester/>.

³ <http://robustrails.man.dtu.dk>.

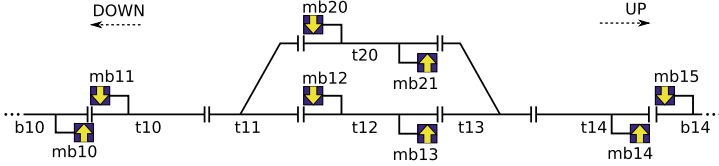


Fig. 1. A railway network layout example. From [23].

The Considered Interlocking Systems. An interlocking system is a signalling system component that is responsible for safe routing of trains through (a fraction of) a railway network under its control. An interlocking system is traditionally specified by a layout of the railway network that it controls and a so-called *interlocking table* that specifies allowed routes through the network and conditions for these routes to be exclusively reserved by a train. In Fig. 1 an example of a railway network layout for a small station is given. As it can be seen it consists of (1) train detection sections that are either linear sections (like $t10$) or switchable points (like $t11$) having a stem side and two branching sides (e.g. $t11$ has its stem next to $t10$ and its branches next to $t20$ and $t12$, respectively) and (2) markerboards⁴ (like $mb10$) placed at the ends of linear sections and only visible in one direction (e.g. $mb10$ is visible in direction UP). As a general rule for the networks considered in this paper, there is at most one markerboard in each end of a linear section and it can only be seen when leaving the section. Furthermore, at the borders of a network, there are always two linear sections (like $b10$ and $t10$) with a signal configuration having an *entry signal* on the border section and an *exit signal* on the section next to the border section. Furthermore, networks are assumed to be *loop-free*⁵.

The Tools and Method. The RobustRailS tools are centred around two inter-related DSLs (domain-specific languages):

- IDL: a DSL [23] for specifying (1) a generic, behavioural, formal model of a product line of interlocking systems and their environment and (2) generic safety properties in the form of state invariants, and
- ICL: a DSL [22] for specifying configuration data (a railway network layout and an interlocking table) that can be used to instantiate generic models and properties.

The RobustRailS tools can be used to formally verify the design of an interlocking system in the following steps, summarized in Fig. 2:

1. A generic model and generic properties are specified in IDL.

⁴ We are considering modern ERTMS level 2 based interlocking systems for which there are no physical signals. They are replaced by markerboards, and in the control system there are virtual signals associated with the markerboards. Throughout the paper we use the term *signal* as a synonym for *markerboard*.

⁵ A network is *loop-free*, if there are no physically possible path through the network containing the same section more than once.

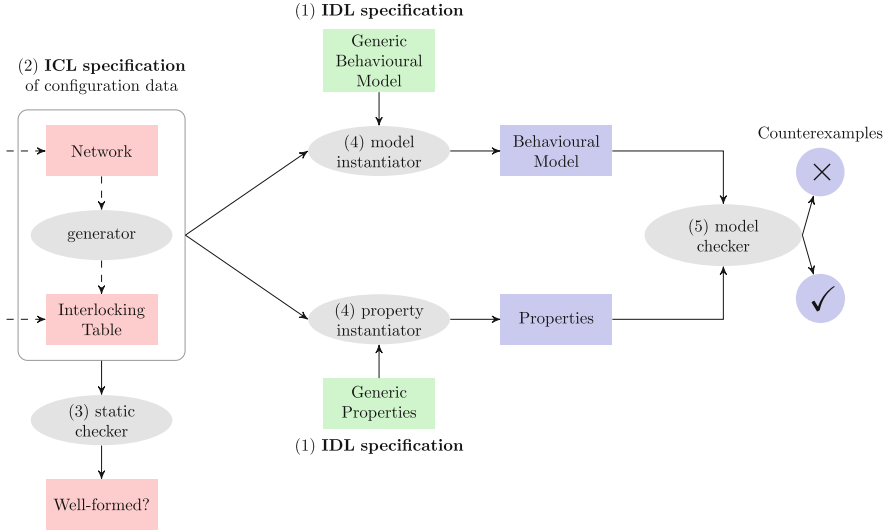


Fig. 2. The RobustRailS tool suite. From [23].

2. A railway network layout and its corresponding interlocking table are specified in ICL in the following order: first the network layout, and then the interlocking table. The creation of the latter is either done manually or generated automatically from the network layout.
3. A static checker verifies whether the configuration data is statically well-formed [7] according to the static semantics [24] of ICL.
4. Generators instantiate a generic behavioural model and generic safety properties with the well-formed configuration data to generate a model and safety properties for the network and routes described in the configuration data.
5. The generated model instance is then checked against the generated properties by a bounded model checker performing a k -induction proof.

The static checking in step (3) is intended to catch errors in the network layout and interlocking table, while the model checking in step (5) 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 Jan Peleska’s RT-Tester framework [18, 21]. The bounded model checker in RT-Tester uses the SONOLAR SMT solver [20] to compute counterexamples showing the violations of the base case or induction step.

Applications. The RobustRailS method and tools have been used to successfully verify the safety of several interlocking systems. The first application was the Danish interlocking system for EDL, the first regional line in Denmark commissioned in the Danish Signalling Programme. First, the IDL language was used to specify a generic model for the novel family of Danish interlocking systems and generic safety conditions expressing that there are *no train collisions* (i.e. there must at most be one train on each section at the same time) and *no*

derailments (i.e. when a train traverses a point, the point must be switched in the right direction for the train to pass). Then the network for the complete EDL line consisting of eight stations of various complexity was specified in the ICL language and an interlocking table was automatically generated from this. Then method steps 3–5 were performed. The verification metrics can be found in Table 1. For more details on this case study, see [25]. Other applications are mentioned in Sect. 4.2.

This achievement of model checking an interlocking system for such a big railway network was quite remarkable. A key reason for that was the use of RT-Tester’s SMT based bounded model checker to perform an induction proof. That pushed considerably the limits of the size of networks for which interlocking systems can be verified.

4 Compositional Verification

However, networks of very large stations still exceed the model checking capacity. Therefore, to be able to perform verification for *any* size of networks, we have previously [2, 8, 15, 16] suggested to use a compositional verification method on top of the RobustRailS verification method.

The idea of our compositional method is as follows: Assume given a generic model and generic safety properties for no collisions and no derailments. To verify an interlocking system instance for a specific network N , divide the network into two parts (sub-networks) N_1 and N_2 , and then verify the interlocking system instances for these two networks using the RobustRailS method and tools. This division process can be applied repeatedly until all sub-networks are small enough to be verified.

In Sect. 4.1, we explain the compositional method in more detail, and in Sect. 4.2 we report on some case studies applying the method. Using our compositional method rises the question: which decomposition of a given network should be made? In Sect. 4.3 we explain an idea for that.

4.1 A Method for Compositional Verification

To introduce the compositional method, we first need to define what is a *cut* of a network, and how the sub-networks should be generated by the cut.

Cut Specifications. A *single cut* is a cut that can be performed between any two neighbouring, non-border sections $t1$ and $t2$ in a network N . An example of a single cut is shown in Fig. 3. The *specification* of that single cut is the pair $(t1, t2)$. To divide a network into two parts, it is not always enough to perform a single cut, but a *cluster cut* consisting of several single cuts may be needed. An example of a cluster cut is shown in Fig. 4. The *specification* of a *cluster cut* is the set of specifications of each of its single cuts. A cut is *legal*, if it divides the network into exactly two parts, no route is cut by more than one single cut, and no flank/front protecting elements⁶ are separated by the cut from the sections they protect. In this paper we assume that flank/front protecting is not adopted.

⁶ In the end of Sect. 4.3 the notion of flank protection is explained.

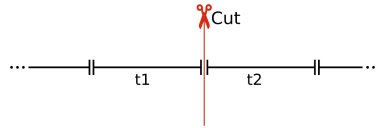


Fig. 3. An example of a single cut. From [8].

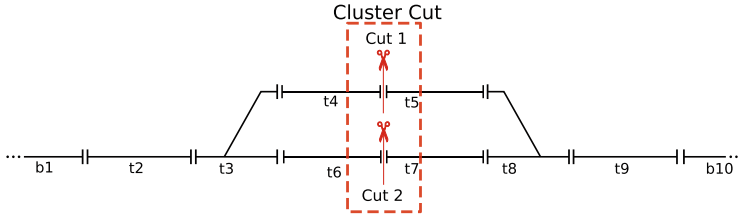


Fig. 4. An example of a cluster cut. From [8].

Decomposing a Network According to a Cut Specification. Given a net N and a legal cut specification, the network is decomposed into two networks as follows:

- if a single cut is between linear sections $t1$ and $t2$, first divide the network N between $t1$ and $t2$, obtaining two sub-networks N_{-1} and N_{-2} , and then add to N_{-1} and N_{-2} at the respective cut a border section and also an entry and an exit signal at that border, if there were not already signals placed around the cut. By doing so, two well-formed networks are obtained: N_1 and N_2 . Figure 5 shows how a network is decomposed into two networks by a single cut $(t1, t2)$. It can be seen how N_1 is obtained from the sub-network N_{-1} on the left-hand side of the cut by adding a border section $b1$ and border signals s_{entry_1} and s_{exit_1} . N_2 is obtained in a similar way. When it is clear from the context, sometimes we also call the resulting networks N_1 and N_2 *sub-networks*;
- if a single cut is between a linear section $t1$ and a point p , the decomposition is treated as if there was an additional linear section $t2$ between $t1$ and p , and the cut specification was $(t1, t2)$;
- if a single cut is between two points $p1$ and $p2$, the decomposition is treated as if there were two additional linear sections $t1$ and $t2$ between $p1$ and $p2$, and the cut specification was $(t1, t2)$.
- if the cut is a cluster cut, the above rules are simultaneously applied to each of its single cuts.

A tool that takes a network and a cut specification as arguments and returns the two networks obtained by decomposing the network according to the cut specification has been developed [17]. This tool is called the *RobustRailS Network Cutter*.

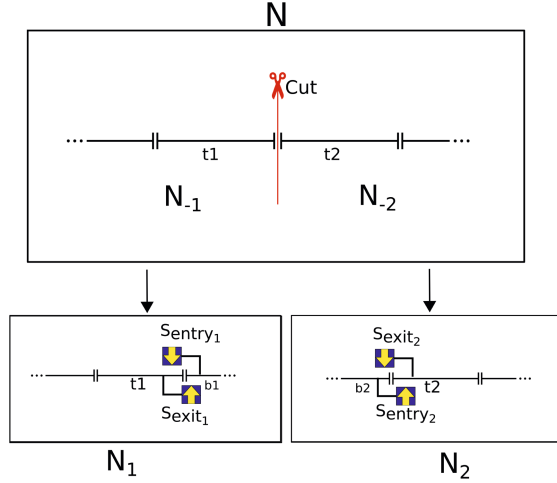


Fig. 5. An example of a decomposition of a network into two networks. From [8].

Method Steps. Using a legal cut allows to perform compositional verification in the following steps:

1. Decompose a network N according to a legal cut specification, achieving two networks N_1 and N_2 .
2. For $i = 1, 2$, apply the interlocking table generator to N_i , check the resulting specification by the static checker, and generate a model m_i and properties ϕ_i from that.
3. For $i = 1, 2$, verify that m_i satisfies ϕ_i .

In [8] it is proved that this method is *sound*. This means that in order to prove safety of the model generated from the whole network, it is sufficient to verify safety for each of the models generated from the two sub-networks formed by a legal cut.

4.2 Case Studies

A number of case studies applying the presented compositional verification approach to different networks with different characteristics and layouts have been carried out. Table 1 shows the savings in verification time and needed memory obtained applying the compositional method to non-trivial cases. For each case, the statistics are shown first for each sub-network, then the global consumption of time and memory of the compositional approach and its reduction are shown in comparison with that of a monolithic verification for the full network. The first three examples have been presented at international conferences [1, 2, 16]; in particular the first one is the already mentioned EDL line, which has been decomposed in sub-networks related to each station of the line, among which the

Table 1. Verification statistics for the compositional verification method applied to some interlocking examples.

Example	Linears	Points	Signals	Routes	Time (s)	Memory (MB)
<i>NFM2017</i> [16]						
Gadstrup	14	3	16	21	62	567
Havdrup	10	2	12	14	19	264
L. Skensved	15	3	16	21	72	616
Køge	58	23	62	75	5170	9243
Herfølge	6	2	10	14	13	210
Tureby	6	2	10	14	11	203
Haslev	10	2	12	14	14	256
Holme-Olstrup	12	2	16	20	22	352
Compositional					5383	9243
Full EDL	110	39	126	179	14352	22476
Reduction %					72.49%	68.88%
<i>SEFM2017</i> [2]						
Low	28	13	26	56	12895.35	12176.6
High	25	10	24	66	8052.92	9517.9
Compositional					20948.27	12176.6
Full Fismn	49	23	46	124	51770.64	42483.7
Reduction %					59.54%	71.34%
<i>RSSRail22</i> [1]						
LVR7A Left	20	7	31	30	670	2083
LVR7B Right	15	5	23	18	108	846
Compositional					778	2083
Full LVR7	26	12	42	48	2387	5467
Reduction %					67.41%	61.90%
<i>Tramway line</i>						
Down	12	5	12	12	81.42	462.8
Middle	9	4	8	12	55.77	392.2
Up	8	3	8	10	22.40	266.7
Compositional					159.59	462.8
Full line	22	12	20	62	28206.00	22762.7
Reduction %					99.43%	97.97%
<i>Flying junction</i>						
Each of 4 subnetworks	12	4	12	20	108.47 <i>max</i>	600.2 <i>max</i>
Compositional					369.69	600.2
Full junction	24	16	16	40	55853.76	23587.2
Reduction %					99.34%	97.45%

Køge station maintains its own high complexity. The second example is a single cut of a large network whose layout has been extracted from a portion of the main Florence station, while the third is a Belgian station on which a cluster cut has been applied, with the aim to compare the method with the decompositional approach of [14]. The remaining two have been purposely defined to explore different layout characteristics: one is inspired by a tramway network, that is, a single track tramway line with several branches and passing loops; the other is a complex flying junction, that allows grade-separated crossing of two double track lines, as well as full interconnection among the tracks of the two lines.

The highest savings are obtained when, in the full network, several routes do not conflict and therefore can be used concurrently, contributing to the state space explosion, due to interleaving of concurrent train movements over such routes: if the cut is made such that the number of independent routes inside a sub-network is low, the concurrency degree is dramatically decreased. This is the case of the tramway line example, divided into three sub-networks, and of the flying junction example, where the cut produces four almost isomorphic sub-networks of far lower complexity.

A deeper study on the correlation between full network topology, cut strategy, and verification savings by decomposition is planned as future work.

4.3 A Decomposition Strategy

Using the presented compositional verification method leaves the question: which cuts should be made in order to decompose a network into small networks that are fast to verify? In this section we will exploit the idea of providing a library of pre-verified, elementary networks and a strategy for dividing a given network into sub-networks of which as many as possible are elementary.

Elementary Networks. As elementary networks we allow one of the network patterns shown in Fig. 6: an elementary network can be a sequence of linear sections having only the required signals at the two borders (see *a*) and *b*)). It can alternatively (see *c*) and *d*)) be a network containing just one point surrounded by at least two linear sections on each of its three sides. There are only the required signals at the three borders and optionally zero, one, two or three of the signals shown directly facing the point. All patterns admit an unbounded number of linear elements at specific positions. In *c*) there is only one linear section between the the point and each of the three border sections, while in *d*), there are two (or more) linear sections between the point and the border section on the stem side. In a similar way it is allowed to have two (or more) linear sections between the point and the border sections on the branching sides of the point.

Model instances of the networks of Fig. 6 have been model checked to be safe, for all the admitted combinations of presence of markerboards, but without the presence of the admitted extra linear sections. Moreover, a result from [8] allows us to add an unbounded number of linear sections at the indicated specific

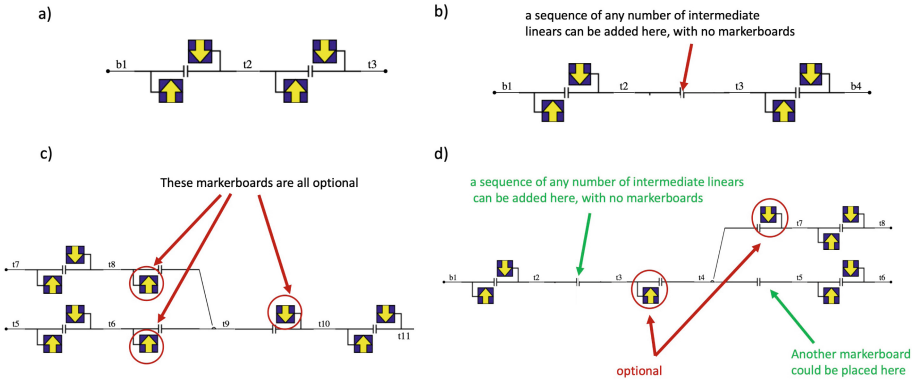


Fig. 6. Patterns for elementary networks.

positions without impacting safety. Hence, we can conclude that *model instances for all elementary networks are safe*.

Decomposing a Network. Given a network, now the idea is to search for places to make legal cuts, one by one, such that the network can be divided into parts that are either elementary networks or non decomposable networks (that is, they cannot be cut without breaking the rules for legal cuts). In the ideal case that the decomposition leads to networks that are all elementary, no model checking is needed.

As an example, consider the network shown in Fig. 7. By making the three cuts (two single cuts (083, PM02U) and (PM02U, PM03U) and the cluster cut $\{(802, PM04U), (801, PM04U)\}$ shown by green lines, one by one, one achieves the four elementary networks N_1^1 , N_1^2 , N_1^3 , and N_2^3 shown in Fig. 8.

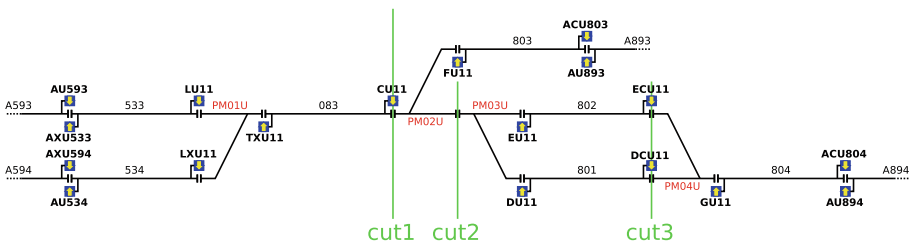
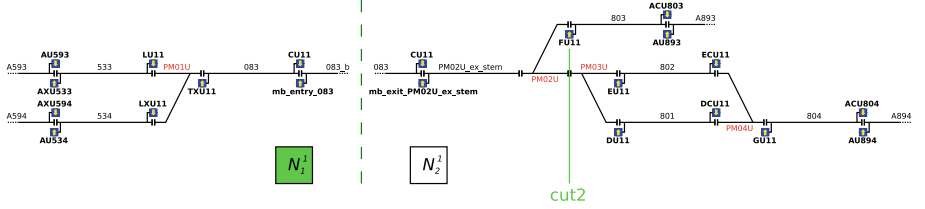
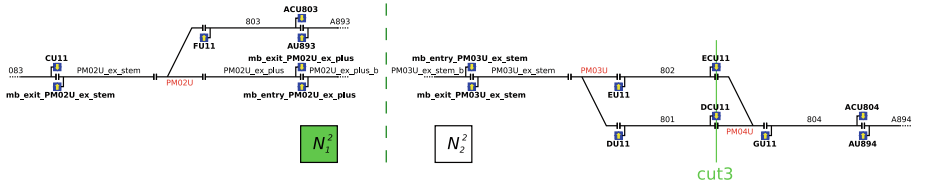


Fig. 7. Cuts shown on a network (LVR1).

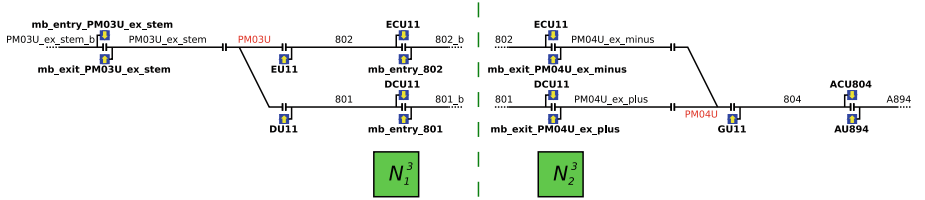
In practice, a possible process of finding such cuts for a loop-free network N is as follows, provided that there are no flank/front protecting elements:



(a) Networks $N_1^1 + N_2^1$ resulting from decomposing the LVR1 network by $cut1$.



(b) Networks $N_1^2 + N_2^2$ resulting from decomposing N_2^1 by $cut2$.



(c) Networks $N_1^3 + N_2^3$ resulting from decomposing N_2^2 by $cut3$.

Fig. 8. Decomposition of the LVR1 network in three steps according to the three cuts shown in Fig. 7. The four resulting green sub-networks N_1^1 , N_1^2 , N_1^3 , and N_2^3 are elementary.

1. Start searching from the neighbour (linear section) l of some border section b of N . The search direction is from l towards the next adjacent element in the direction opposite to b .
2. Follow the sections from l one by one as long as they are linear and do not have any signals attached until one of the following happens:
 - (a) If a linear section having an *exit signal* is found, we have reached a border and no cut should be made, as the considered network is an elementary linear network.
 - (b) If two consecutive, linear sections $l1$ and $l2$ are found, and at least one of them has a signal facing the other, then a decomposition using the cut $(l1, l2)$ should be made. By this the generated sub-network containing $l1$ will by construction be an elementary linear network. The search for further cuts should then continue from $l2$ in the other sub-network.
 - (c) If a point p is found, then we should continue to search for cuts on the two other sides of p . This search depends on from which side p was found: the

stem or one of the branching sides. In both cases the search also depends on whether the two other sides are connected or not.⁷

- i. If coming from the stem of p , and the two branching sides are not connected, then we should search for cuts in each of the two branches. The search here is similar to the search starting from a border, except that if a second point is found, a single cut must be made just *before* that point. The two searches may hence lead to totally zero, one or two single cuts, dividing the network into (1) an elementary point network containing p and (2) zero, one or two additional sub-networks in which a search for cuts must be performed. For instance, when searching for a cut in network N_2^1 in Fig. 8 (a), starting from $PM02U_ex_stem$, a single cut, $cut2$: $(PM02U, PM03U)$, will be found in the lower branch, while no cuts are found in the upper branch (as a border is met before any further points or non-border signals), so it results in two sub-networks.
- ii. If coming from the stem, and the two branching sides are connected, then a similar search is made in each of the branches. In this case two single cuts (one in each branch) will be found and these must be combined in a cluster cut (in order to divide the network into two parts) leading to an elementary point network containing p and one additional sub-network to which search for cuts must be recursively applied. That is e.g. the case when searching for a cut ($cut3$) in network N_2^2 in Fig. 8 (b), starting from $PM03U_ex_stem$.
- iii. If coming from a branching side of p , and the stem and the other branching side are not connected, searches for cuts in the other branch and on the stem side must be performed in a similar way to case i) above. That happens e.g. when searching for the first cut in Fig. 7 starting from linear section 533.
- iv. If coming from a branching side of p , and the stem and the other branching side are connected, the search to be performed is similar to case ii), except that in some cases it is not possible to find a legal cluster cut: that happens if a potential cluster cut divides a route into three parts⁸, as shown in Fig. 9, where the cluster cut shown by a red, dotted line is found when searching from $L1$ on the upper branching side of point $P1$. In such a case we should then start a search from another border to see if a cut can be found from there. It is our conjecture that it is always possible to find a border from which it is possible to find a legal cluster cut through the connected sub-component, provided that the network is loop-free. For instance, in Fig. 9, the legal cluster cut $\{(P2, P1), (L24, P4)\}$ shown by a dashed, green line can be found when searching from $L2$. Figure 10 gives an

⁷ By *connected* we mean that by navigating the graph of the not yet visited part of the network starting from the two sides we eventually reach a common point.

⁸ Note that when coming from the stem, we do not have such a problem, as a route cannot pass through a point via its two branches.

example of a network that cannot be decomposed into elementary networks as the network is not loop-free.

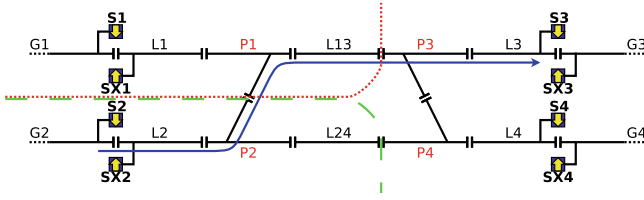


Fig. 9. The cluster cut $\{(P1, P2), (L13, P3)\}$ shown by a red, dotted line is illegal as it divides the route shown as a blue, solid arrow in three parts. The cluster cut $\{(P2, P1), (L24, P4)\}$ shown by a green, dashed line is legal. (Color figure online)

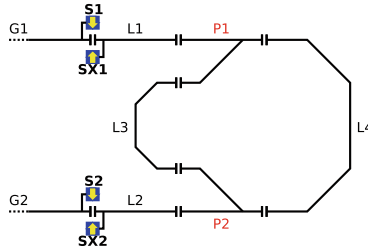


Fig. 10. An example of a non-loop-free, non-decomposable network.

In railway interlocking systems, specific additional mechanisms may be included to enforce safety also in the case in which trains do not strictly respect signals, due to a driver's misbehaviour or accidental inability to brake. In the *Flank Protection* mechanism 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. In the example of Fig. 11 locking of route r requires the point $t20$ to be in the straight position in order to protect the flank of route r by a train accidentally missing the closed $mb20$ signal. If both point $t20$ and route r lie in the same sub-network when a cut is operated, the extra condition on the point position has no impact on compositionality: but this is not the case for the drawn cut, which separates the protecting and the protected points. As discussed in [8], in this case compositional verification results do not fully hold, so we consider such a cut as not legal: both elements should instead be in the same sub-network, which is therefore not elementary, since it contains two points. In the presentation of our approach, we have assumed that there is no flank protection. If flank protection was adopted, legal cuts would not be allowed to separate the protecting and the protected points. However, then we would no longer be able to decompose a loop-free network into networks that are all elementary.

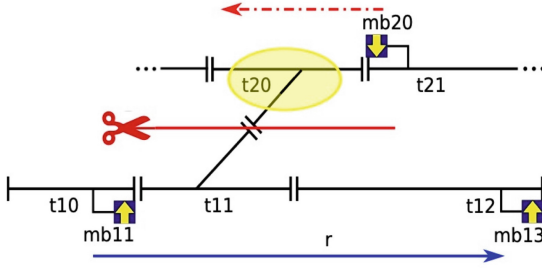


Fig. 11. Cut through a flank protection.

5 Conclusions and Future Work

In this paper we have presented a compositional method for model checking the safety of interlocking systems. The idea of the compositional method is to divide the network under control into some sub-networks and then model check the model instances for these networks instead of model checking the model instance of the full network. The paper suggests a novel strategy for decomposing a network into a number of small sub-networks that, under certain conditions, all belong to a library of pre-verified elementary networks, so no model checking is actually needed for the specific application.

This strategy will be the subject of further work, including its implementation in a tool for the automatic decomposition of a network: this will be accompanied by a deeper assessment of its soundness and completeness, as well as of its tractability, and on the other end will enable experimenting it on several complex layout examples. Also, this study will address a consolidated definition of the conditions under which the conjecture of full decomposition in elementary networks holds, and the impact of flank protection or other analogous protection mechanisms on the applicability of the decomposition algorithm.

Dedication and Acknowledgements

We dedicate this paper to Jan Peleska who we admire so much for his brilliant research in applicable formal methods for safe industrial products. The first author (Anne Haxthausen) would like to express her gratitude to Jan for more than 25 years of the most enjoyable, inspiring, and fruitful collaboration.

The RobustRailS tools used in the work presented in this paper were developed by her PhD student, Linh H. Vu, under co-supervision by Jan Peleska who came with brilliant ideas and generously provided the possibility of using RT-Tester as backend. All three authors are very indebted to Peleska and Vu. Furthermore, the authors would like to thank Hugo D. Macedo, who contributed to the initial work on the compositional method used in this paper, and to thank Anna Nam Anh Nguyen and Ole Eilgaard for their network cutter tool which we have also used in this paper.

References

1. Fantechi, A., Gori, G., Haxthausen, A.E., Limbrée, C.: Compositional verification of railway interlockings: comparison of two methods. In: Dutilleul, S.C., Haxthausen, A.E., Lecomte, T. (eds.) *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification: Fifth International Conference, RSSRail 2022, Paris, France, June 1–2, 2022, Proceedings. Lecture Notes in Computer Science*, vol. 13294, pp. 3–19. Springer Nature Switzerland AG (2022). https://doi.org/10.1007/978-3-031-05814-1_1
2. Fantechi, A., Haxthausen, A.E., Macedo, H.D.: Compositional verification of interlocking systems for large stations. In: Cimatti, A., Sirjani, M. (eds.) *SEFM 2017. LNCS*, vol. 10469, pp. 236–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66197-1_15
3. Ferrari, A., Ter Beek, M.H.: Formal methods in railways: a systematic mapping study. *ACM Comput. Surv.* **55**(4), 1–37 (2022)
4. Ferrari, A., Magnani, G., Grasso, D., Fantechi, A.: Model checking interlocking control tables. In: *FORMS/FORMAT 2010 - Formal Methods for Automation and Safety in Railway and Automotive Systems*. pp. 107–115. Springer (2010). https://doi.org/10.1007/978-3-642-14261-1_11
5. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H.: Systematic evaluation and usability analysis of formal methods tools for railway signaling system design. *IEEE Trans. Softw. Eng.* **48**(11), 4675–4691 (2022)
6. Ferrari, A., Mazzanti, F., Basile, D., Ter Beek, M.H., Fantechi, A.: Comparing formal tools for system design: a judgment study. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 62–74. ICSE 2020, Association for Computing Machinery, New York, NY, USA (2020)
7. Haxthausen, A.E., Østergaard, P.H.: On the use of static checking in the verification of interlocking systems. In: Margaria, T., Steffen, B. (eds.) *ISoLA 2016. LNCS*, vol. 9953, pp. 266–278. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_19
8. Haxthausen, A.E., Fantechi, A.: Compositional verification of railway interlocking systems. *Form. Asp. Comput.* **35**(1) (2023). <https://doi.org/10.1145/3549736>
9. Huang, W., Peleska, J.: Complete model-based equivalence class testing. *Int. J. Softw. Tools Technol. Transfer* **18**(3), 265–383 (2016)
10. James, P., Möller, F., Nguyen, H.N., Roggenbach, M., Schneider, S., Treharne, H.: Decomposing scheme plans to manage verification complexity. In: Schnieder, E., Tarnai, G. (eds.) *FORMS/FORMAT 2014–10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, pp. 210–220. Institute for Traffic Safety and Automation Engineering Technische Univ., Braunschweig (2014)
11. James, P., et al.: Verification of solid state interlocking programs. In: Counsell, S., Núñez, M. (eds.) *SEFM 2013. LNCS*, vol. 8368, pp. 253–268. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05032-4_19
12. Limbrée, C., Cappart, Q., Pecheur, C., Tonetta, S.: Verification of Railway Interlocking - Compositional Approach with OCRA. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) *RSSRail 2016. LNCS*, vol. 9707, pp. 134–149. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_10
13. Limbrée, C., Pecheur, C.: A framework for the formal verification of networks of railway interlockings - application to the Belgian railway. *Electr. Commun. Eur. Assoc. Study Sci. Technol.* **76** (2018)

14. Limbrée, C.: Formal verification of railway interlocking systems. Ph.D. thesis, UCL Louvain (2019)
15. Macedo, H.D., Fantechi, A., Haxthausen, A.E.: Compositional verification of multi-station interlocking systems. In: Margaria, T., Steffen, B. (eds.) ISO/LA 2016. LNCS, vol. 9953, pp. 279–293. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_20
16. Macedo, H.D., Fantechi, A., Haxthausen, A.E.: Compositional model checking of interlocking systems for lines with multiple stations. In: Barrett, C., Davies, M., Kahsai, T. (eds.) NFM 2017. LNCS, vol. 10227, pp. 146–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57288-8_11
17. Nguyen, A.N.A., Eilgaard, O.B.: Development and use of a tool supporting compositional verification of railway interlocking systems. Master’s thesis, Technical University of Denmark, DTU Compute (2020)
18. Peleska, J.: Industrial-strength model-based testing - state of the art and current challenges. In: Petrenko, A.K., Schlingloff, H. (eds.) 8th Workshop on Model-Based Testing, Rome, Italy. vol. 111, pp. 3–28. Open Publishing Association (2013)
19. Peleska, J., et al.: A real-world benchmark model for testing concurrent real-time systems in the automotive domain. In: Wolff, B., Zaïdi, F. (eds.) ICTSS 2011. LNCS, vol. 7019, pp. 146–161. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24580-0_11
20. Peleska, J., Vorobev, E., Lapschies, F.: Automated test case generation with SMT-solving and abstract interpretation. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 298–312. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_22
21. Verified systems international GmbH: RT-Tester model-based test case and test data generator - RTT-MBT - User Manual (2013). <http://www.verified.de>
22. Vu, L.H., Haxthausen, A.E., Peleska, J.: A domain-specific language for railway interlocking systems. In: Schnieder, E., Tarnai, G. (eds.) FORMS/FORMAT 2014–10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems, 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: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification: Second International Conference, RSSRail 2017, Pistoia, Italy, November 14–16, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10598, pp. 99–115. Springer Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_7
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. *Sci. Comput. Programm.* **133**, Part 2, 91–115 (2017)
26. Winter, K.: Optimising ordering strategies for symbolic model checking of railway interlockings. In: Margaria, T., Steffen, B. (eds.) *ISoLA 2012*. LNCS, vol. 7610, pp. 246–260. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34032-1_24