Compositional Verification of Multi-Station Interlocking Systems

Macedo, Hugo Daniel dos Santos; Fantechi, Alessandro; Haxthausen, Anne Elisabeth

Published in:
Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2016)

Link to article, DOI:
10.1007/978-3-319-47169-3_20

Publication date:
2016

Document Version
Peer reviewed version

Compositional Verification of Multi-Station Interlocking Systems

Hugo D. Macedo[1], Alessandro Fantechi[1,2], and Anne E. Haxthausen[1]

[1] DTU Compute, Technical University of Denmark
[2] DINFO - University of Florence, Via S. Marta 3, Firenze, Italy

Abstract. Because interlocking systems are highly safety-critical complex systems, their automated safety verification is an active research topic investigated by several groups, employing verification techniques to produce important cost and time savings in their certification. However, such systems also pose a big challenge to current verification methodologies, due to the explosion of state space size as soon as large, if not medium sized, multi-station systems have to be controlled.
For these reasons, verification techniques that exploit *locality* principles related to the topological layout of the controlled system to split in different ways the state space have been investigated. In particular, compositional approaches divide the controlled track network in regions that can be verified separately, once proper assumptions are considered on the way the pieces are glued together.
Basing on a successful method to verify the size of rather large networks, we propose a compositional approach that is particularly suitable to address multi-station interlocking systems which control a whole line composed of stations linked by mainline tracks. Indeed, it turns out that for such networks, and for the adopted verification approach, the verification effort amounts just to the sum of the verification efforts for each intermediate station and for each connecting line.

1 Introduction

An interlocking system is responsible for guiding trains safely through a given railway network. It is a vital part of any railway signalling system and has the highest safety integrity level (SIL4) according to the CENELEC 50128 standard [1]. Automated safety verification of interlocking systems is hence an important issue and an active research topic, investigated by several research groups, see e.g., [3,4,5,9,21]. Model-checking techniques are considered for this purpose, but, notwithstanding the important advancements witnessed for these techniques, they fail to give results on large interlocking systems due to the state space explosion problem.

In this paper we elaborate on previous results of the RobustRailS research project[3], in which an automatic method for the formal verification of interlocking

[3] In Denmark, in the years 2009-2021, new interlocking systems that are compatible with the standardised European Train Control System (ETCS) Level 2 [2] will be de-

systems was developed [18,19,20]. Using a combination of SMT based bounded model checking (BMC) and inductive reasoning, the method succeeded to verify interlocking systems of large sizes. Indeed, in a comparison with other model checkers, the proposed method succeeded to verify a multi-station interlocking controlling a whole line, while the other tools failed to conclude the verification with the given time and resources available. Nevertheless, the verification of the considered systems just reached the limits of time and memory showing that larger systems cannot be addressed any more, and hence these verification methods need to be further improved.

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, by limiting the state space on which to verify it.
This principle has been exploited in [22] to define domain-oriented optimizations of the variable ordering in a BDD-based verification. Locality can be used also for slicing, as suggested in [3] and [10,8]. The idea is to consider only the portion of the model that has influence on the property to be verified, by a topological selection of interested track elements (therefore closely related to the *cone of influence* of the property): this allows for a much more efficient verification of the single property, but comes at the price of repeating the verification for every property (in principle a single verification of the conjunction of all safety properties on the whole model would otherwise suffice), and of separately checking that verifying slices does actually imply the satisfaction of desired properties for the whole system. Nevertheless, it appears that when automated, this process can offer significant time and memory savings.

A compositional approach that also exploits locality is the one used in [11], where the interlocking of a quasi-symmetrical station is divided in two halves, and the verification of one half takes into account assume/guarantee conditions at the interface with the other half. The verification effort is hence repeated for the two halves, with the extra effort of proving that assume/guarantee conditions do hold at the interface: locality allows such conditions to be rather simple so that they do not add much time to the verification.

In this paper we adopt a similar compositional approach to chop a large interlocking system into smaller fragments, but by considering a different way of dividing the network in fragments. The presented approach is particularly suitable to address multi-station interlocking systems, which control a whole line composed of stations linked by mainline tracks. Indeed, it turns out that for such networks, and for the adopted verification approach, the verification effort amounts just to the sum of verification efforts for each intermediate station and for each connecting line. Since in reality the intermediate stations often share

---

ployed in the entire country within the context of the Danish Signalling Programme. In the context of the RobustRailS project accompanying the signalling programme on a scientific level, the proposed method will be applied to these new systems.

an identical layout, the gain in verification is made even larger by factoring out the identical stations.

The paper is structured as follows: in Sect. 2 we present the verification approach on which we have built our compositional approach; in Sect. 3 we show how a network can be divided in smaller fragments, and how the verification of the whole network is obtained from the verification of the single elements, using as running example a simple network. In Sect. 4 we report on the results given by the application of our decomposition approach to a small example and to the EDL line that nearly reached the capacity bounds of the adopted tools when proved as a whole. In both cases the results show significant gains in verification effort can be achieved. Sect. 5 summarizes the achieved results and discusses possible future extensions and improvements of the work presented here, especially in the direction of addressing interlocking systems that control large stations.

2 The new Danish Route-based Interlocking Systems

In this section we introduce briefly the new Danish interlocking systems and the domain terminology. The subsequent subsections explain (2.1) different components of a specification of an interlocking system which is compatible with ERTMS/ETCS Level 2 [2], and (2.2) how the safety properties are verified, respectively.

2.1 Specification of Interlocking Systems

The specification of a given route-based interlocking system consists of two main components: (1) a railway network, and (2) a corresponding interlocking table.

Railway Networks A railway network in ETCS Level 2 consists of a number of track and track-side elements of different types[4]: linear sections, points, and marker boards. Figure 1 shows an example layout of a railway network having six linear sections (`b10`,`t10`,`t12`,`t14`,`t20`,`b14`), two points (`t11`,`t13`), and eight marker boards (`mb10`, ..., `mb21`). These terms, and their functionality within the railway network, will be explained in more detail in the next paragraphs.

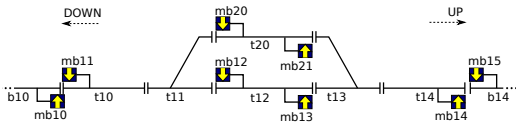

Fig. 1: An example railway network layout.

[4] Here we only show types that are relevant for the work presented in this article.

A *linear section* is a section with up to two neighbours: one in the *up* end, and one in the *down* end. For example, the linear section `t12` in Figure 1 has `t13` and `t11` as neighbours at its up end and down end, respectively. In Danish railway's terminology, *up* and *down* denote the directions in which the distance from a reference location is *increasing* and *decreasing*, respectively. The reference location is the same for both up and down, e.g., an end of a line. For simplicity, in the examples and figures in the rest of this article, the *up* (*down*) direction is assumed to be the left-to-right (right-to-left) direction.

A *point* can have up to three neighbours: one at the *stem*, one at the *plus* end, and one at the *minus* end, e.g., point `t11` in Figure 1 has `t10`, `t12`, and `t20` as neighbours at its stem, plus, and minus ends, respectively. The ends of a point are named so that 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. When a point is in the PLUS (MINUS) position, its *stem* end is connected to its *plus* (*minus*) end, thus traffic can run from its *stem* end to its *plus* (*minus*) end and vice versa. It is not possible for traffic to run from *plus* end to *minus* end and vice versa.

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. Note that sections are bidirectional, i.e., trains are allowed to travel in both directions (but not at the same time).

Along each linear section, up to two *marker boards* (one for each direction) can be installed. A marker board can only be seen in one direction and is used as reference location (for the start and end of routes) for trains going in that direction. For example, in Figure 1, marker board `mb13` is installed along section `t12` for travel direction up. Contrary to legacy systems, there are no physical signals in ETCS Level 2, but interlocking systems have a *virtual signal* associated with each marker board. Virtual signals play a similar role as physical signals in legacy systems: a virtual signal can be OPEN or CLOSED, respectively, allowing or disallowing traffic to pass the associated marker board. However, trains (more precisely train drivers) do not see the virtual signals, as opposed to physical signals. Instead, the aspect of virtual signals (OPEN or CLOSED) is communicated to the onboard computer in the train via a radio network. For simplicity, the terms *virtual signals*, *signals*, and *marker boards* are used interchangeably throughout this paper.

Interlocking Tables An interlocking system monitors constantly the status of track-side elements, and sets them to appropriate states in order to allow trains traveling safely through the railway network under control. The new Danish interlocking systems are route-based. A *route* is a path from a *source* signal to a *destination* signal in the given railway network. A route is called an *elementary route* if there are no signals that are located between its source signal and its destination signal, and that are intended for the same direction as the route.

In railway signalling terminology, *setting* a route denotes 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.

An *interlocking table* specifies the elementary routes in the given railway network and the conditions for setting these routes. The specification of a route r and conditions for setting r include the following information:

- src(r) – the source signal of r,
- dst(r) – the destination signal of r,
- path(r) – the list of sections constituting r's path from src(r) to dst(r),
- overlap(r) – a list of the sections in r's *overlap*[5], i.e., the buffer space after dst(r) that would be used in case trains overshoot the route's path,
- points(r) – a map from points[6] used by r to their required positions,
- signals(r) – a set of protecting signals used for flank or front protection [15] for the route, and
- conflicts(r) – a set of conflicting routes which must not be set while r is set.

that is needed while verifying the expected properties.

2.2 The RobustRailS Verification Approach and Toolkit

This section describes shortly the RobustRailS verification method and tool set that we use as verification technology. For detailed information, see [17,18,19,20].

The method for modelling and verifying railway interlocking systems is a combination of formal methods and a domain-specific language (DSL) to express network diagrams and interlocking tables. According to this, an environment consisting of the following components is provided.

- *An editor and static checker* [6] for editing and checking that a DSL specification (describing an interlocking system) follows certain well-formedness rules.
- The bounded *model checker* of RT-Tester [12,16] which we use for making k-induction proofs as described in [20].
- *Generators* transforming a DSL specification into inputs to the model checker:
  - a behavioural model of the interlocking system and its environment, and
  - the required safety properties given as linear temporal logic formulae.

The 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,

[5] An overlap section is needed when, for the short distance of a marker board to the end of the section, there is the concrete danger that a braking train stops after the end of the section, e.g. in adverse atmospheric conditions.

[6] These points include points in the path and overlap, and points used for flank and front protection. Sometimes it is required to protect tracks occupied by a train from another train not succeeding to brake in due space. For details about flank and front protection, see [15].

   (b) and then the interlocking table (this is either done manually or generated automatically from the network layout).
2. The static checker verifies whether the configuration data is statically well-formed according to the static semantics [19] 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 toolchain associated with the method has been implemented using the RT-tester framework [12,16]. The bounded model checker in RT-tester uses the SONOLAR SMT solver [13] to compute counterexamples showing the violations of the base case or induction step.

3 Method

We now proceed to describe the details of how we use the locality features of railway networks to verify large interlocking systems in a compositional approach. The idea is to decompose the networks in chunks that are separately verified for safety properties, and to show how under given conditions such separate verifications are enough to guarantee that the whole network satisfies the safety properties as well. We show that a multi-station interlocking system satisfies such conditions if a suitable (and natural) divide strategy is applied. Indeed, the strategy can be easily automated, providing a completely automated method to verify this kind of interlocking systems.

3.1 Border assumptions

The approach of Sect. 2 is able to verify an interlocking system when immersed in an *environment* that satisfies some assumptions on the borders of the system network. The borders of the network (in the diagram of Figure 1, they are partially dotted) are defined as track elements which are not under control of the interlocking system for the network under consideration. Thus our method assumes the following:

- *assumptions on the network*
  1. the border elements include a marker board (signal), controlled by the interlocking that protects the entrance to the network;
  2. the routes of the network are elementary: start at one and end at another of the marker boards of the network (including the border ones), and do not include intermediate marker boards;
- *assumptions on the trains' behaviour*

3. trains can "materialize" on a border element at any time, unless a train has already materialized on it and has not yet moved away;
4. trains on a border track element can enter the network if they are allowed to do (that is, if the border virtual entry signal is OPEN);
5. trains cannot enter in the middle of the network;
6. trains that pass from an adjacent track element to a border one can "dematerialize" at any moment.

The assumptions on the status of the border track elements are an abstraction of the real environment in which the network is immersed: since the real environment puts restrictions on the actual behaviour of trains (for example, it introduces minimum time intervals between two incoming trains), these assumptions give actually an *over-approximation* of its behaviour. Hence, if safety is verified for every behaviour of the network and of its environment, it is verified also in the case the environment is substituted by a more constrained behaviour, which is the case when the network is connected to another one.

3.2 Divide strategy

We exploit the border assumptions to divide a network controlled by a single interlocking into several sub-networks, each to be controlled by its own interlocking system, such that the verification of the original interlocking system can be done by verifying the interlocking systems of the sub-networks. For the purpose of this paper, we consider that a division produces two sub-networks (a down and an up sub-network) by making one or several cuts each respecting the conditions defined below:

A network cut satisfies the *border cut conditions (BCCs)* if:

1. it separates two consecutive linear sections such that in the joint between the two there is an up markerboard on the upper part of the down section and a down markerboard on the down part of the upper section;
2. no *overlap* section for the up sub-network includes elements in the down sub-network, and vice-versa;
3. no *flank/front protection* requirement on the up sub-network depends on elements of the down sub-network, and vice-versa.

The conditions above assure that the setting (reservation) of any route depends only on the status of the elements belonging to one of the sub-networks.

A network with a cut (between the sections t14 and b14) satisfying the BCCs is illustrated in Figure 2. In such a figure one can identify two sub-networks one on the left-hand side and one on the right-hand side, thus allowing the monolithic interlocking system controlling the whole network to be safely divided into two interlocking systems controlling two networks.

Such a situation can be found when considering a multi-station interlocking, which controls several stations on a line: the long sections of track between two stations, according to the Danish signalling rules, do not carry over route

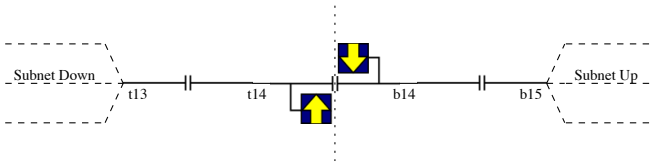

Fig. 2: A cut between sections t14 and b14 satisfies the border cut conditions.

information and present specific joints of the form described in Fig. 2, and are hence natural places in which to cut the overall network.

In the division process a network is inspected in search for regions that present candidate patterns to be cut, that is, joints of the form of Figure 2. Once such a joint has been pointed out, the cut is operated forming two sub-networks as illustrated in Figure 3 and Figure 4, where the down section of the joint (t14) will be kept as the down border element of the Up sub-network, and up section of the joint (b14) is kept as the up border element of the Down sub-network. The search is then recursively applied to the created sub-networks, until no more suitable cut points can be found.

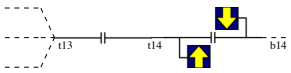

Fig. 3: Resulting sub-network Down.

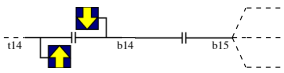

Fig. 4: Resulting sub-network Up.

On more complex network layouts, our division process would require refinements to cope with richer interfaces between the two sub-networks, that would have, anyway, to respect the BCCs.

The concepts described above allow to automate the compositional verification of multi-station interlocking systems by dividing the network in sub-networks by means of a three steps algorithm:

1. Search the network for cuts satisfying the BCCs, and divide the network into sub-networks.
2. For each of the resulting sub-networks, complete the specification of a sub-interlocking system using the generator mentioned in item 1 of Sect. 2.2.
3. Verify a model of each sub-interlocking system by following the steps in item 2, 3, and 4 of Sect. 2.2.

3.3 Soundness of the approach

Building upon the assumptions on the model and verification process we sketch an induction argument for the soundness of the compositional approach focusing on the case where we divide a network into two. First we prove that the resulting sub-networks satisfy the required network assumptions 1 and 2. Then we argue that the disjoint union of the interlocking tables generated for the sub-networks constitute the whole interlocking table for the monolithic network. Then we proceed to guarantee that from the verification results of the local sub-interlocking systems it is possible to infer the global verification result.

When dividing a monolithic network into sub-networks at joints satisfying BCC 1, it follows that the resulting sub-networks satisfy the required assumption 1 for network borders also at their new borders.

When dividing a monolithic network satisfying network assumption 2 (that routes are elementary) into sub-networks at joints satisfying BCC 1, the generated routes for the sub-networks will be disjoint and "internal" either to one or to the other sub-networks, and they will also satisfy network assumption 2 for the sub-networks. Furthermore, the union of the routes generated for the sub-networks is the set of routes of the original monolithic interlocking system.

When the cutting interfaces also satisfy conditions BCC 2 and BCC 3, the conditions for setting any route depends only on the status of the sole elements belonging to the sub-network to which the route belongs, and therefore the disjoint union of the interlocking tables generated for the sub-networks constitutes the whole interlocking table for the monolithic network.

Due to the above conclusions, the behaviour of the interlocking for the big network is the sum of the behaviours of the interlocking systems for the sub-networks. Only the train behaviour at the new borders is different. Due to the fact that trains' behaviour assumption 3 allows the materialization of trains in the border elements of the interlocking systems for the sub-networks, our compositional approach to the monolithic network over-approximates the verification of the safety property by allowing more trains to "materialize" in sections that are not border of the monolithic network. Thus terminating our argument.

4 Experiments

In this section we present the results of applying our approach to a case study invented for the sake of explanation and to a real world case study we adapted from our group's previous work: the Early Deployment Line (**EDL**) in the Danish Signalling Programme. In Sect. 4.1 we describe our experimental approach, then in Sect. 4.2 we describe and summarize the results obtained in the invented case study, and the results for the (**EDL**) example are shown in Sect. 4.3.

4.1 Experimental approach

For each of the case studies, we put the method described in Sect. 3 in practice by first obtaining sub-networks (in XML format) according to the divide strategy

explained in Sect. 3.2. Then for each sub-network, we use the RobustRailS verification tool described in [18,19,20] and Sect. 2.2 to generate a model instance and safety properties, and then to verify that the generated safety properties hold in the model.

We also use the RobustRailS verification tool to monolithically verify the railway network (without decomposing it) such that we can compare verification metrics for the compositional approach with verification metrics for the monolithic approach.

While verifying each instance we measure (in seconds) the real time taken to obtain the verification result and what was the total memory (in MB) used by the verification tool. In addition we collect some statistics about the model instances as presented in Table 1 and Table 2. Such statistics provide a basis for complexity comparison and include: the number of linear and point sections for each instance, the number of marker boards, routes, and the state space dimension (in logarithmic scale).

All the experiments for both case studies have been performed on a machine with an Intel(R) Xeon(R) CPU E5-2667 @ 2.90Gz, 64GB RAM, and running CentOS 6.6, Linux 2.6.32-504.8.1.el6.x86_64 kernels.

4.2 Mini-Tiny-Fork: A small case study

To illustrate and evaluate our approach we have devised a case study based on existing networks from [18,19] inspired by the typical examples from real world used in other studies about formal verification of railway interlocking systems [5,7,9,22]. The used network, although invented, represents a realistic case.

The case study is based on three networks: **M**ini, **T**iny, and **F**ork. **M**ini is the network shown in Figure 1 and it represents a typical pattern of a station found on lines with small stations. **T**iny, a network expressing a typical line pattern between stations, is depicted in Figure 5, and **F**ork, a common network shape for a terminal station, is depicted in Figure 6.

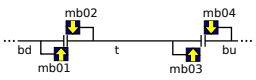

Fig. 5: **T**iny Network.

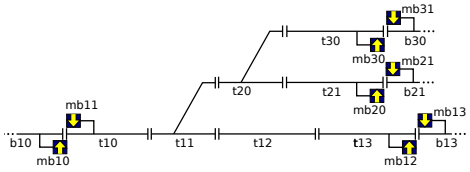

Fig. 6: **F**ork Network.

These networks, when connected left to right in the respective order, form an example of a monolithic network of a line with two stations: **M**ini and **F**ork, connected by a single track: **T**iny. We designate this network by the acronym $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$ and show it (without labels) in Figure 7.

The $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$ network can be used to compare the monolithic approach described in Sect. 2 with the compositional approach described in Sect. 3 because the border (bd,t) between **M**ini and **T**iny, and the border (t,bu) between **T**iny and **F**ork are both satisfying the BCCs. Therefore, the $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$ network can be naturally divided into the three networks **M**ini, **T**iny, and **F**ork according to the method described in Sect. 3, hence allowing for a compositional analysis that we will refer to as **M**+**T**+**F**.

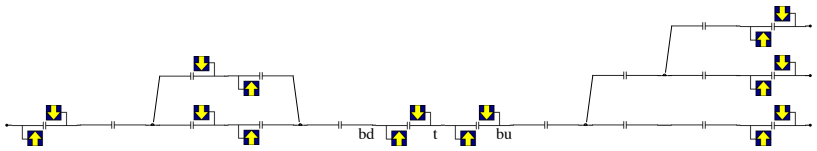

Fig. 7: $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$ Network.

Table 1 shows the verification metrics for the compositional analysis **M**+**T**+**F**: all metrics are obtained by summing the corresponding metrics for the sub-networks, except for the memory usage, which is calculated as the maximum memory usage of the sub-networks. The table also shows the verification metrics for the monolithic analysis of the monolithic network $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$. In all cases the verification tool succeeded to verify the safety properties using simple induction (k-induction with $k = 1$). As it can be observed the verification time and memory usage of the compositional analysis **M**+**T**+**F** is, as expected, much better than for the monolithic analysis of $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$: The verification time is five times faster and the memory usage (204 MB) is more than three times less. Moreover, if the verification for the sub-networks were run in parallel, our compositional approach would achieve a running time of just 18 seconds. Even though memory consumption would increase in this case, the parallelization would still use less memory resources (the sum of individual memory usages: 391 MB) than the monolithic case (759 MB).

Table 1: Verification metrics for the Mini-Tiny-fork case study.

	Linears	Points	Signals	Routes	$log_{10}(\|S\|)$	Time	Memory
Tiny	3	0	4	2	11	1	13
Fork	9	2	8	6	40	10	174
Mini	6	2	8	12	37	18	204
M+**T**+**F**	18	4	20	20	76	29	204
$\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$	14	4	16	20	76	145	759

We introduced faults in the model and ran the tool to find a counterexample witnessing a safety violation. As expected the time taken to find the witness was improved by the compositional analysis. Whereas $\mathbf{M} \cdot \mathbf{T} \cdot \mathbf{F}$ takes 6.6 hours using 20.7 GB, the compositional approach takes 40 minutes using 7.2 GB.

4.3 EDL: A real world case study

The **EDL** is the first regional line in Denmark to be commissioned in the Danish Signalling Programme. The line spreads over 55 kilometres from the station in Roskilde to Næstved's station, and the statistics shown in Table 2 gives insight on its composition.

We apply seven cuts to the network dividing it into eight sub-networks, each corresponding to an **EDL** station. The result are six networks of fairly similar complexity (Gadstrup, Havdrup, Herfølge, Tureby, Haslev, and Holme-Olstrup) plus two more complex ones (L. Skensved and Køge). With such a division we decompose the verification of the interlocking system for **EDL** into the separate verification of the eight stations.

Table 2: Verification metrics for the EDL line.

	Linears	Points	Signals	Routes	$log_{10}(\|S\|)$	Time	Memory
Gadstrup	14	3	16	21	73	86	513
Havdrup	10	2	12	14	51	20	263
L. Skensved	20	4	22	28	101	223	1212
Køge	52	22	54	66	306	6581	9393
Herfølge	6	2	10	14	39	13	191
Tureby	6	2	10	14	39	12	180
Haslev	10	2	12	14	51	22	261
Holme-Olstrup	12	2	16	20	63	27	350
Compositional	130	39	152	191	682	6984	9393
EDL	116	39	138	191	682	22793	26484

As in the Mini-Tiny-Fork case study, the verification tool succeeded to verify the safety properties for the eight sub-interlocking systems using simple induction (k-induction with $k = 1$) and the verification metrics show that for the compositional analysis (see the table entry Compositional) the verification time is approximately a third (taking approx. 2 hours) than for the monolithic analysis (taking aprox. 6 hours).

Furthermore, the compositional analysis uses less than half of the memory resources (9393 MB) because we only need as much as the maximum value of memory used to verify each sub-interlocking. Such memory reduction is important when checking real world interlocking systems where a single station with a complex network may quickly exhaust the amount of memory available. As

already discussed, if run in parallel our compositional approach would achieve a much better running time. Even though memory consumption would increase, the parallelization would only use roughly half (the sum of the individual memory usages: 12363 MB) of the memory resources than the monolithic case.

5 Conclusions

We have presented a compositional method to address the safety verification of railway interlocking systems of large size by means of formal verification techniques. The method, built on top of tools providing support for efficient verification of this kind of systems, is tailored to the characteristics of multi-station interlocking systems, that is, systems that control a line connecting several stations.

The idea of our method is as follows: Multi-station interlocking systems exhibit a good deal of topological locality, which is exploited to easily, and automatically, decompose the layout in smaller chunks, easier to verify. Figures on the time and memory consumption gains obtained have been given for a simple example and for a large, real world case study, the **EDL** line in Denmark, showing that this approach is a good first step in the direction of defining a general compositional approach able to exploit as better as possible the inherent locality present in models of the interlocking logics of large networks.

The presented method assumes some Border Cut Conditions to be fulfilled in order to guarantee soundness. These conditions require that the cuts are made such that each route of the full network and the overlap and flank protection of that route are completely within one of the sub-networks after the decomposition. More work is needed to identify other patterns and conditions that identify a safe place to cut a network employing these features in sub-networks, maintaining some possibility of compositional verification.

As evidenced in the results for the **EDL** line, and expected from the parallelization theory, the gains of the method depend on the running time for the largest sub-interlocking system resulting from the division step. Future works should take into account such bottleneck either by applying it to suitable lines (for instance big lines with stations with relative simple complexity), or by finding better strategies to divide the interlocking systems, or both.

In general, a more systematic study is needed on which ways to cut a network allow for compositional safety verification and which do not. We are currently exploring the challenge to find a way to cut into sub-networks the layout of very large stations, which tend to appear in terminal stations. In such case, due to the intricacy of the routes allowing trains to traverse the many tracks of the station, it is likely that finding a generic, automated, way to divide the network in sub-networks will be difficult, but future solutions to such problem would benefit from compositional approaches like ours.

Acknowledgments. The authors would like to express their gratitude to Jan Peleska and Linh Hong Vu for their excellent contribution to the development of

the RobustRailS interlocking verification method and tool set and for an always very enjoyable collaboration. Furthermore, the authors would like to express their gratitude to Peter Holm Østergaard for helping us with some practical work, and to Ross Edwin Gammon and Nikhil Mohan Pande from Banedanmark (Railnet Denmark) and Jan Bertelsen from Thales for helping us with their expertise about Danish interlocking systems.

The research presented in this paper has been funded by Villum Fonden and the RobustRailS project granted by Innovation Fund Denmark.

References

1. CENELEC European Committee for Electrotechnical Standardization. *EN 50128:2011 – Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems*. 2011.
2. European Railway Agency. ERTMS – System Requirements Specification – UNISIG SUBSET-026, April 2014. Available at `http://www.era.europa.eu/Document-Register/Pages/Set-2-System-Requirements-Specification.aspx`.
3. Alessio Ferrari, Gianluca Magnani, Daniele Grasso, and Alessandro Fantechi. Model Checking Interlocking Control Tables. In Eckehard Schnieder and Géza Tarnai, editors, *FORMS/FORMAT 2010 – Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 107–115. Springer, 2010.
4. Helle Hvid Hansen, Jeroen Ketema, Bas Luttik, Mohammad Reza Mousavi, Jaco van de Pol, and Osmar Marchi dos Santos. Automated Verification of Executable UML Models. In Bernhard K. Aichernig, Frank S. de Boer, and Marcello M. Bonsangue, editors, *Formal Methods for Components and Objects*, volume 6957 of *Lecture Notes in Computer Science*, pages 225–250. Springer, 2010.
5. Anne E. Haxthausen, Marie Le Bliguet, and Andreas A. Kjær. Modelling and Verification of Relay Interlocking Systems. In Christine Choppy and Oleg Sokolsky, editors, *Foundations of Computer Software, Future Trends and Techniques for Development*, volume 6028 of *Lecture Notes in Computer Science*, pages 141–153. Springer, 2010.
6. Anne E. Haxthausen and Peter H. Østergaard. On the use of static checking in the verification of interlocking systems. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation*, volume This Volume of *Lecture Notes in Computer Science*. Springer, 2016.
7. Anne E. Haxthausen, Jan Peleska, and Ralf Pinger. Applied Bounded Model Checking for Interlocking System Designs. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops. Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 205–220. Springer, 2014.
8. Philip James, Faron Möller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, and Helen Treharne. Decomposing scheme plans to manage verification complexity. In Schnieder and Tarnai [14], pages 210–220.
9. Philip James, Faron Möller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, Helen Treharne, Matthew Trumble, and David Williams. Verification of Scheme Plans Using CSP||B. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods*, volume 8368 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2014.

10. Phillip James, Andy Lawrence, Faron Moller, Markus Roggenbach, Monika Seisenberger, Anton Setzer, Karim Kanso, and Simon Chadwick. Verification of solid state interlocking programs. In Steve Counsell and Manuel Núñez, editors, *Software Engineering and Formal Methods - SEFM 2013 Collocated Workshops. Revised Selected Papers*, volume 8368 of *Lecture Notes in Computer Science*, pages 253–268. Springer, 2014.
11. Christophe Limbrée, Quentin Pecheur, Charles Moller, and Stefano Tonetta. Verification of railway interlocking - compositional approach with OCRA. In Thierry Lecomte, Ralf Pinger, and Alexander Romanovsky, editors, *Reliability, Safety and Security of Railway Systems - RSSR 2016*, volume 9707 of *Lecture Notes in Computer Science*. Springer, 2016.
12. Jan Peleska. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. In Alexander K. Petrenko and Holger Schlingloff, editors, 8th Workshop on Model-Based Testing, Rome, Italy, volume 111 of *Electronic Proceedings in Theoretical Computer Science*, pages 3–28. Open Publishing Association, 2013.
13. Jan Peleska, Elena Vorobev, and Florian Lapschies. Automated Test Case Generation with SMT-Solving and Abstract Interpretation. In M. Bobaru, K. Havelund, G. Holzmann, and Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 2011.
14. Eckehard Schnieder and Géza Tarnai, editors. *FORMS/FORMAT 2014 - Formal Methods for Automation and Safety in Railway and Automotive Systems*. Institute for Traffic Safety and Automation Engineering, Technische Universität Braunschweig, 2014.
15. Gregor Theeg, Sergeĭ Valentinovich Vlasenko, and Enrico Anders. *Railway Signalling & Interlocking: International Compendium*. Eurailpress, 2009.
16. Verified Systems International GmbH. *RT-Tester Model-Based Test Case and Test Data Generator - RTT-MBT - User Manual*, 2013. Available on request from `http://www.verified.de`.
17. Linh H. Vu, Anne E. Haxthausen, and Jan Peleska. A Domain-Specific Language for Railway Interlocking Systems. In Schnieder and Tarnai [14], pages 200–209.
18. Linh H. Vu, Anne E. Haxthausen, and Jan Peleska. Formal modeling and verification of interlocking systems featuring sequential release. In Cyrille Artho and Peter Csaba Ölveczky, editors, *Formal Techniques for Safety-Critical Systems*, volume 476 of *Communications in Computer and Information Science*, pages 223–238. Springer International Publishing, 2015.
19. Linh Hong Vu. *Formal Development and Verification of Railway Control Systems - In the context of ERTMS/ETCS Level 2*. PhD thesis, Technical University of Denmark, DTU Compute, 2015.
20. Linh Hong Vu, Anne E. Haxthausen, and Jan Peleska. Formal modelling and verification of interlocking systems featuring sequential release. *Science of Computer Programming*, 2016.
21. K. Winter. Symbolic Model Checking for Interlocking Systems. In Francesco Flammini, editor, *Railway safety, reliability, and security: technologies and systems engineering*. IGI Global, 2012.
22. Kirsten Winter. Optimising Ordering Strategies for Symbolic Model Checking of Railway Interlockings. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, volume 7610 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2012.