Towards Automatic Design and Verification for Level 3 of the European Train Control System

Robert Wille*†   Tom Peham*   Judith Przigoda‡   Nils Przigoda‡
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
‡Siemens Mobility GmbH, Braunschweig, Germany
{robert.wille, tom.peham}@jku.at   {judith.przigoda, nils.przigoda}@siemens.com

***Abstract*—For centuries, block signaling has been the fundamental principle of today's railway systems to prevent trains from running into each other. But the corresponding infrastructure of physical blocks each requiring train detection methods is costly. Therefore, initiatives such as the *European Train Control System* (ETCS) and, here, particularly Level 3 of ETCS aim for the utilization of virtual sections which allow for a much higher degree of freedom and provide significant potential for increasing the efficiency in today's train schedules. However, exploiting this potential is a highly non-trivial task which, thus far, mainly relied on manual labor. In this work, we provide an initial automatic methodology which aids designers of corresponding railway networks and train schedules. The methodology utilizes design automation expertise (here, in terms of satisfiability solvers) to unveil the potential of ETCS Level 3. Case studies (including a real-life example inspired by the Norwegian Railways) confirm the applicability and suitability of the proposed methodology.**

I. Introduction

Since the opening of the first public railway line in 1825, railways have quickly developed into a vital component of the public traffic system. Even with the appearance of cars they stay fundamental for both human as well as freight transport and, furthermore, they will continue to become even more important as the world strives towards a carbon neutral future [1]. Since the beginning around 200 years ago, signaling systems are a crucial part for an efficient but also safe operation of railways [2].

In the earliest days of railways – long before radio or other means of remote communication were available – trains were protected from running into each other by separating them through time. After a train left a station, there was a defined time to wait until the next train could leave. Obviously, this led to problems if anything went wrong, e. g., the train had to stop for any reason. Therefore, in the late 1800s, the fundamental principle on which almost every railway in the world still relies on was introduced: *block signaling*. Here, the whole railway network is divided into sections (so-called *blocks*), and at most one train is allowed to occupy a given block at any given time. Based on this principle, interlockings use *Trackside Train Detection* (TTD) systems that collect information on the occupation of blocks and, with that, control the trains.

In the past, each country implemented their own signaling principles – leading to the establishment of different signaling characteristics as well as different realizations of trackside or trainside solutions for train control [3], [4]. This obviously leads to problems as soon as border-crossing traffic is considered. For example, in the 27 countries of the European Union, more than 15 different signaling systems are used that are incompatible with each other. In order for a train to travel through these countries, it would need to be equipped with *all* those systems. This is not only expensive, but also challenging as the space under a locomotive is limited. In order to remedy this unsatisfying situation, the European Union started the *European Railway Traffic Management System* (ERTMS) [5] initiative to develop the means for interoperable railway traffic all through Europe. An essential part of this is the *European Train Control System* (ETCS) [6], [7] which strives to harmonize and improve train control systems throughout Europe.

Originally, ETCS mainly aimed for harmonizing those systems – with ETCS Level 1 unifying the way trains locally communicate with the trackside and ETCS Level 2 removing the physical signals from the trackside and conducting signaling purely via radio [8]. With ETCS Level 3, additionally new concepts for improving existing systems are introduced. In fact, for the first time since the 19th century, ETCS Level 3 deviates from the principle of physical blocks each equipped with TTD means and additionally allows for the consideration of virtual sections within blocks. This allows for a much higher degree of freedom in the utilization of existing railway networks and, hence, provides significant potential for increasing the efficiency in today's train schedules.

However, the implementation and utilization of ETCS Level 3 is just beginning. Several papers have recently been published which considered the formalizing of the underlying concepts, e. g., in iUML-B [9], [10], Electrum [11], SysML/KAOS [12], Event-B [13], [14], or SPIN [15]. Case studies and demonstrations of the ETCS Level 3 can be found, e. g., in [16] and [17]. Even first simulations based on ETCS Level 3 have been conducted [18], [19]. However, all these methods still heavily rely on manual labor and hardly aim at the automation of the design tasks that need to be tackled to fully exploit the potential of this new concept.

In this work, we propose an initial methodology which aids designers in this domain by *automatically* designing and verifying railway layouts and train schedules exploiting the potential of ETCS Level 3. The proposed methodology allows to tackle several design and verification tasks such as to verify whether train schedules indeed work on new ETCS Level 3 railway networks, to determine alternative layouts if this is not the case, and/or to additionally optimize the layout in order to improve a train schedule. To this end, we formulate the corresponding task in terms of a satisfiability instance and utilize corresponding solving engines to resolve them. Case studies considering different railway networks and train schedules (including a real-life example inspired by the Norwegian Railways) confirm the applicability and suitability of the proposed methodology.

The remainder of this paper is structured as follows: First, Section II gives a more detailed review on the ETCS Level 3 and its potential as well as the resulting design tasks. Afterwards, we introduce the proposed design automation solution in Section III. Section IV illustrates by means of several case studies how the methodology can be used for certain design tasks within the ETCS Level 3 domain. Finally, Section V concludes the paper.

II. Background & Motivation

This section reviews and illustrates the new potential that comes with the introduction of ETCS Level 3. Afterwards, we discuss design and verification tasks which result from this new standard and for which no automatic solutions are available yet.

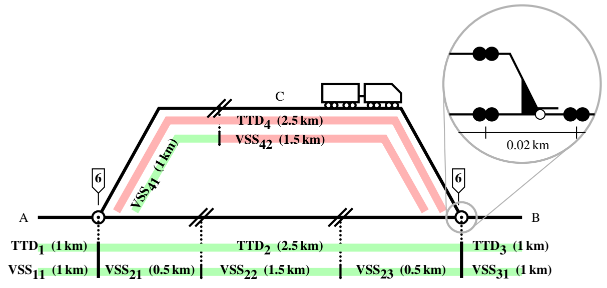
A. Potential of ETCS Level 3

In order to prevent trains from running into each other, signaling systems are essential for any railway system. Currently, almost every railway system in the world relies on *block signaling*, where the whole railway network is divided into *blocks* which are equipped with means of train detection in order to determine whether a given block is currently occupied by a train or not. Those blocks are eventually observed and controlled by interlockings which manage the train movement [20], i. e., move the points for every train in the appropriate direction as well as signals the drivers whether and with what speed they can proceed or if they have to stop. To this end, *Trackside Train Detection* (TTD) systems are used that collect corresponding information on the occupation of blocks. They utilize, e. g., *axle counters* to gather the respectively required information. Accordingly, corresponding blocks are frequently also called *TTD sections*.

Thus far, TTD sections often have been defined by geography, economical considerations, or as a trade-off between efforts of installing axle counters and possible benefits. Because of this, the length of the corresponding TTD sections range from some meters (e. g., around points) or some hundred of meters (e. g., at train stations) to several kilometers (e. g., in remote areas). Of course, this significantly affects the efficiency of the underlying railway networks.

With the introduction of the ETCS Level 3, those principles change for the first time since the 19th century. Rather than relying on fix blocks (which require, e. g., axle counters at the beginning and end of each TTD section), now also virtual sections are possible. More precisely, the TTD sections which already exist are divided into smaller *Virtual Subsections* (VSSs), which do not require physical axle counters anymore and, hence, allow for a much higher degree of freedom in the utilization of existing railway networks[1]. The following example illustrates the new concept and its potential.

Example 1. *Figure 1 shows a railway network which is used as running example in the remainder of this paper. Originally, this network was divided into 4 TTD sections (denoted by TTD_1–TTD_4) – each with corresponding axle counters at their beginning and end*[2]*. Using the ETCS Level 3, this layout can additionally be enriched by VSSs (as shown in Fig. 1 through the 7 sections VSS_{11}–VSS_{42}). This allows much more freedom in the utilization of railway networks without the need of adding more axle counters. For example, if a train occupies the virtual subsection VSS_{42}, just a portion of the network is blocked (highlighted red), while, thus far, the entire TTD section TTD_4 would be blocked.*



(a) Example railway network illustrating TTDs and VSSs

Train	Start	Goal	Speed[km/h]	Length[m]	Departure Time	Arrival Time
1	A	B	180	400	0:00	0:04:30
2	B	A	120	700	0:00	0:04:00
3	A	C	120	100	0:01	0:03:00
4	B	A	180	250	0:01	0:05:00

(b) Example schedule

Fig. 1: Example layout and schedule

B. Resulting Design & Verification Tasks

Using ETCS Level 3 in general and VSS in particular allows for a much more flexible use of train systems and provides significant potential for increasing the efficiency in today's train schedules. At the same time, this poses significant challenges for designers of corresponding railway networks and/or schedules aiming to exploit this potential. For example, the following design tasks emerge with the introduction of ETCS Level 3[3]:

1) *Verification of Train Schedules on ETCS Level 3 Layouts*, in which it is verified whether a given train schedule indeed can be executed on an also given TTD/VSS layout.
2) *Generation of VSS Layouts*, in which a given railway network with existing TTD sections is extended by VSS sections so that an also given train schedule can be realized.
3) *Schedule Optimization Using the Potential of VSS*, in which a given railway network with existing TTD sections is extended by VSS sections such that the best possible train schedule is determined, i. e., a schedule which minimizes the arrival times of all trains (given corresponding departure times).

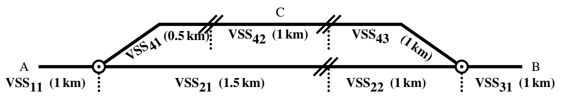
Example 2. *Consider again the track layout as well as the given TTD sections shown in Fig. 1a. Additionally considering the train schedule as shown in Fig. 1b raises the question, whether this schedule indeed can be conducted on this TTD layout. In fact, in this particular case, this is not possible because, after all four trains have departed, all four TTDs are blocked and no train can move on. As this might not be obvious at a first glance, automatic methods doing this verification would be beneficial.*[4] *At the same time, the schedule in Fig. 1b is not completely impossible for the given railway network. In fact, employing a VSS layout as shown in Fig. 1a, the schedule indeed can be conducted. Because trains do not block entire TTDs, Trains 1 and 3 can both move into "Station C", freeing the way for Trains 2 and 4. However, determining such a layout might not be trivial for more complex networks and, hence, designers would benefit from automatic methods for that as well. Finally, the overall schedule can further be improved by choosing a VSS layout as shown in Fig. 2a. With this, we can execute the schedule shown in Fig. 2b which allows Trains 1, 2, and 3 to arrive their final destination much earlier. To detect further potential like that, automatic methods would be helpful again.*

[1]Note that VSSs have originally been introduced in the ECTS Hybrid Level 3 but, according to *Shift2Rail* [21] (one of the biggest European projects within the ETCS domain), got established in ECTS Level 3 as well.

[2]Note that, in reality, TTD borders at points are never equipped with just one axle-counter but with one at each outgoing track (as depicted in the right-hand side of Figure 1) to be able to determine the direction the train is moving to. However, for sake of simplicity, we consider a single axle counter at each point in the following.

[3]Note that also design tasks beyond that are emerging and the methodology proposed in the next section is rather flexible in handling other design tasks as well. However, to keep the following descriptions concise and, due to the page limitation, we focus on the following design tasks as representatives.

[4]Note that, for sake a clarity, we chose a rather simple example here. In real world applications, however, trains may be able to take different paths, employ different speeds, different waiting times, etc., which substantially increases the complexity – making it indeed a verification task (covering all possibilities) rather than just a simulation task.



(a) Optimal layout

Train	Start	Goal	Speed[km/h]	Length[m]	Departure Time	Arrival Time
1	A	B	180	400	0:00	0:03:30
2	B	A	120	700	0:00	0:02:30
3	A	C	120	100	0:01	0:02:30
4	B	A	180	250	0:01	0:03:30

(b) Improved schedule

Fig. 2: Improved VSS layout and schedule

Solving these design tasks is not trivial. In the worst case, all possible combinations, e. g., of VSS sections, train routes, schedules, etc. need to be considered – an infeasible task when conducted manually and still challenging if done in an automatic fashion. Motivated by that, it is of severe interest to investigate how design automation can help here.

III. Automatic Design and Verification of ETCS Layouts

In this section, we propose an initial methodology that solves the design and verification tasks reviewed and illustrated above in an automatic fashion. To cope with the underlying complexity, we propose to utilize the deductive power of satisfiability solvers. Their intelligent decision heuristics, powerful learning schemes, and fast implication methods allow for efficiently traversing large search spaces and have been proven to be very effective for many practically relevant problems such as model checking [22], stimuli generation [23], test pattern generation [24], and more [25]. Also in the train domain, satisfiability solvers already have been used for certain tasks such as infrastructure verification [26] or railway capacity analysis [27] (but outside of the ETCS domain).

However, in order to use satisfiability solvers, the problem needs to be formulated in a discrete and symbolic fashion. This section first introduces a corresponding formulation. Afterwards, we describe how constraints need to be employed to ensure that all solutions obtained from the formulation are indeed valid. Finally, we show how the resulting formulation eventually can be used to solve the considered design and verification tasks.

A. Discretization and Symbolic Formulation

To comprehensibly describe the considered design and verification tasks, a discrete and symbolic formulation is required which represents

- all existing tracks as well as all possible VSS sections,
- all trains as well as their possible positions in all possible moments of time, and
- the input positions, output positions, as well as (if applicable) positions of intermediate stops for each train together with their corresponding arrival/departure times.

For the tracks and possible VSS sections, we start with a graph representation $G = (V, E)$, where edges $e \in E$ represent different sections of the given railway network and nodes $v \in V$ represent their respective connection points. Naturally, switches and/or axle counters serve as logical connection points within a given layout. Additionally considering VSS, all tracks can be further partitioned in an arbitrary number of sections. Since this, in theory, allows for an infinite number of VSS sections, we discretize the setting by additionally introducing a spatial resolution r_s which defines the smallest section length that is assumed in all further considerations. Each track in the railway network is then partitioned into segments of length r_s, i. e., a track of length l would yield $\frac{l}{r_s}$ segments.

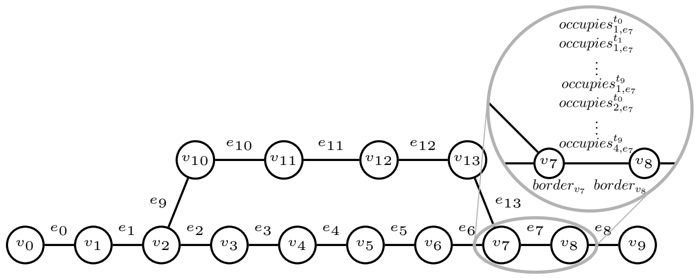


Fig. 3: Symbolic formulation

Example 3. *Consider again the original railway network shown in Fig. 1 and, additionally, assume that a spatial resolution of r_s = 0.5 km is employed. Then, the graph representation shown in Fig. 3 serves as starting point for the generation of the symbolic formulation.*

By defining G this way, we can think of the vertices $v \in V$ as potential VSS borders. From this representation, we can encode every possible VSS layout. To this end, we introduce Boolean variables $border_v$ for each $v \in V$ representing whether or not v separates two VSS sections. That is, $border_v = 0$ states that the two edges connected to v belong to the same VSS section, while $border_v = 1$ states that v is the border between two VSS sections.

Example 4. *Consider again the graph representation shown in Fig. 3 as well as the corresponding $border_v$-variables (exemplarily annotated at the right-hand-side). Additionally assume that $border_{v_2}$, $border_{v_3}$, $border_{v_6}$, $border_{v_7}$, $border_{v_{11}}$ are set to 1 and all other $border_v$-variables are set to 0. Then, this assignment represents a VSS layout composed of 7 VSS sections as already discussed before in Example 1 and shown in Fig. 1. Assigning the $border_v$-variables with other values allows to define all VSS layouts possible in this railway network.*

Having a symbolic formulation for all tracks and possible VSS layouts, we can now provide a formalization for all possible train positions within this discrete network for all possible moments of time. To this end, let's assume that the set $Trains$ denotes a set of trains to be considered. At any given point in time, a train $tr \in Train$ may occupy a segment of the track layout or not. This is encoded by Boolean variables $occupies^{t_i}_{tr,e}$, where $occupies^{t_i}_{tr,e} = 1$ ($occupies^{t_i}_{tr,e} = 0$) states that, at a time step t_i, a train $tr \in Trains$ occupies (does *not* occupy) the segment $e \in E$.

To properly formulate "time step" in this context, we discretize the notion of time in a similar fashion as we discretized track sections before, i. e., we introduce a temporal resolution r_t which defines the smallest amount of time that is assumed in all further considerations. Based on that, all further considerations are assumed to take place within an interval of t_{max} time steps $[t_0, t_1, \ldots, t_{t_{max}-1}]$, where t_{max} is defined by the real time a scenario should be considered divided by r_t. Finally, we additionally consider that each train $tr \in Train$ comes with a certain length and certain maximum speed provided through the variables l_{tr} and s_{tr}, respectively. Those values are accordingly discretized with respect to r_s and r_t, respectively.

Example 5. *Consider again the running example from above and, additionally, assume that a temporal resolution of $r_s = 0.5\,\mathrm{min}$ is employed. Since the entire scenario of* *the schedule shown in Fig. 1 spans over* $5\,\mathrm{min}$*, a total of* $t_{max} = \frac{5}{0.5} = 10$ *time steps is considered in all further considerations (each of these time steps represent* $30\,\mathrm{sec}$ *in real life). Together with the symbolic formulation of all sections, this leads to the introduction of occupies-variables as sketched in Fig. 3 (exemplarily annotated at the right-hand-side). Then, setting* $occupies^2_{1,e_5}$*,* $occupies^2_{1,e_6}$*, and* $occupies^2_{1,e_7}$ *to 1 represents that, at time step 2 (i. e.,* $1\,\mathrm{min}$ *after the start of the scenario), Train 1 occupies the sections represented by edges* e_5, e_6, e_7.

The resulting formulation eventually represents all possible VSS sections as well as all trains and their possible positions in all time steps. Additionally, the input positions, output positions, and positions of intermediate stops (together with their corresponding arrival/departure times) can now easily be defined by setting the corresponding $occupies$-variables for the respective positions (i. e., sections/edges) and time steps to 1 when working with a fixed schedule.

This way, all possible combinations (including possible solutions for the respectively considered design and verification tasks) are formalized in a symbolic fashion.

However, passing this symbolic formulation to a satisfiability solver basically yields an arbitrary assignment to all $border_v$- and $occupies$-variables – representing an arbitrary (and most likely invalid) scenario. Hence, in order to make sure that (1) valid solutions only and (2) solutions which solve a certain design task (such as the ones discussed in Section II-B) are obtained, we finally have to add constraints and objectives which accordingly restrict the variable assignments. This is covered next.

B. Enforcing Constraints

The most obvious constraint that needs to be enforced on the symbolic formulation from above is that all considered trains always occupy at most one place of the network (and are, e. g., not at two different places at the same time). This can properly be enforced by adding a constraint forcing one $occupies$-variable of a train at a time step to be set to 1[5], while all others are forced to be set to 0. More precisely, for each train $tr \in Trains$ and for each time step $t_i \in \{t_0, \cdots, t_{t_{max}-1}\}$, we employ

$$\bigvee_{c \in chains(l^*_{tr})} \Big(\bigwedge_{e \in c} occupies^{t_i}_{tr,e} \bigwedge_{f \in E \setminus c} \neg occupies^{t_i}_{tr,f} \Big),$$

where $chains(l) \subset \mathcal{P}(E)$ denotes all chains of edges of length l in the graph G and l^*_{tr} is the discrete notation of the length of the train tr defined by $l^*_{tr} = \lceil \frac{l_{tr}}{r_s} \rceil$.

In a similar fashion, we have to make sure that only those movements of trains are allowed which are in accordance with the track layout, the maximum speed of each train, as well as the progress of time (i. e., that a train is not in one section of the railway network in one time step and in a completely different one at the other end of the network in the next one). This can properly be enforced by adding a constraint which ensures that, if a train occupies one segment of the network in one time step (i. e., if one particular $occupies$-variable is set to 1), it either has to occupy the same segment in the next time step or any segment which can be reached with the train's speed in one time step (i. e., in the next time step, either the same $occupies$-variable from before has to be set to 1 or one of the "neighboring" ones). More precisely, for each $tr \in Trains$, for each segment $e \in E$, and for each time step $t_i \in \{t_0, \cdots, t_{t_{max}-2}\}$, we employ

$$occupies^{t_i}_{tr,e} \implies \bigvee_{f \in reachable(e,tr)} occupies^{t_{i+1}}_{tr,f},$$

where $reachable(e, tr)$ denotes all segments that the train tr can reach when starting from e (including e itself).

Train movement is also constrained by the TTD/VSS layout. Recall that a VSS section can only be occupied by at most one train. Trains occupying different TTDs are automatically separated because TTD borders are always also VSS borders. Trains occupying the same TTD must be placed in separate VSS. If two trains occupy the same TTD at the same time, we can enforce this by setting the $border$-variable for one of the nodes between the occupied segments to 1. More precisely, for each pair of trains $tr_1, tr_2 \in Trains$ with $tr_1 \neq tr_2$, for each segment $e \in E, f \in TTD(e)$, and for each time step $t_i \in \{t_0, \cdots, t_{t_{max}-1}\}$, we employ

$$\big(occupies^{t_i}_{tr_1,e} \wedge occupies^{t_i}_{tr_2,f}\big) \implies \bigvee_{v \in \text{between}(e,f)} border_v,$$

where $TTD(e)$ is the set of all segments belonging to the same TTD as e and $between(e, f)$ denotes all $v \in V$ that belong to the chain connecting e and f. This constraint also ensures that only valid VSS layouts are generated when the VSS layout is not fix.

With the constraints from above, we almost ensured a correct formulation of the train's behavior. But up to now, it is still technically possible that two trains moving in opposite directions can "go through one another". More precisely, if a train is in segment $e \in E$ in time step t_i and in segment $f \in E$ in time step t_{i+1}, the segments on the path from e to f must not be occupied by any other train at time step t_i or t_{i+1}. Since we cannot know how the train moved from e to f, we need to block all possible paths. To this end, we use the notation $paths(e, f, tr)$ which denotes all paths from e to f that can be traversed by the train tr. Then, "collisions" like that can be prevented by enforcing

$$occupies^{t_i}_{tr_1,e} \wedge occupies^{t_{i+1}}_{tr_1,f} \implies \bigwedge_{g \in \text{paths}(e,f,tr_1)} \neg occupies^{t_i}_{tr_2,g} \wedge \neg occupies^{t_{i+1}}_{tr_2,g}$$

for trains $tr_1, tr_2 \in Trains, tr_1 \neq tr_2$, segments $e \in E$, and for each time step $t_i \in \{t_0, \cdots, t_{t_{max}-2}\}$.

C. Enforce the Objective and Solve the Design Task

All constraints outlined above ensure that the satisfiability solver only determines variable assignments representing valid scenarios. However, additionally it has to be ensured that the actually considered design or verification task is solved as well. Using the tasks discussed in Section II-B, this can be accomplished by adding the following final constraints and/or objective functions to the formulation:

Verification of Train Schedules: Schedules can be encoded as a list of triples (tr, e, t_i) defining for each train $tr \in Trains$ what segment $e \in E$ it should occupy at a particular time step $t_i \in \{t_0, \cdots, t_{t_{max}-1}\}$. This can easily be enforced by setting the corresponding variable $occupies^{t_i}_{tr,e}$ to 1. In order to check, whether this schedule also works on a given TTD/VSS layout, we simply have to enforce this layout by setting the corresponding $border_v$-variables to 1 (to 0), if $v \in V$ separates (does not separate) two VSS sections. By this, the resulting instance becomes satisfiable, if the schedule works with the VSS layout and unsatisfiable if not.

[5]In case a train is so long that it may span over a chain of connected sections, several $occupies$-variables might be set to 1.

Generation of VSS Layouts: Also here, a schedule is given (which can be enforced as done above) but, in contrast to the previous design task, the satisfiability solver is supposed to generate a VSS layout with which the schedule can be executed. A trivial way to approach this is by setting $border_v = 1$ for all $v \in V$. This would assign each segment $e \in E$ to a different VSS. This ensures that a train of length l only occupies exactly l segments. This in turn guarantees that trains are blocking each other as little as possible. However, there might be other layouts that use less VSS and still allow the given schedule to be executed. If designers want a layout that uses as little VSS sections as possible, they can easily enforce that by adding the objective function

$$\min : \sum_{v \in V} border_v .$$

Schedule Optimization Using the Potential of VSS: For this design task, we only take the departure time and all the stops of each train but do not specify when a train should arrive at a given stop, i. e., we only enforce

$$\bigvee_{i=0}^{t_{max}-1} occupies_{tr,e}^{t_i}$$

for each $tr \in Trains$ and its corresponding stops at $e \in E$. Then, we want the satisfiability solver to determine routes for all these trains (and the corresponding arrival times) which are as efficient as possible.

The term "efficient" can thereby be interpreted in different fashions. One interpretation might be that the overall scenario should be completed in at least as possible time steps. Another might be that each single train $tr \in T$ should take all its stops and arrive its final stop as fast as possible. One of the major benefits of the proposed methodology is that all these objectives can easily be enforced by just adding a corresponding constraint and/or objective function. In the following, we illustrate that by means of the objective of minimizing the number of time steps it takes until *all* trains have reached their destination.

To this end, we introduce the Boolean variable $done_{tr}^{t_i}$ where $done_{tr}^{t_i} = 1$ ($done_{tr}^{t_i} = 0$) states that, at time step t_i, a train $tr \in Trains$ has left the network (is still within the network). Let $lastStop(tr) \in E$ be the last stop of train $tr \in Trains$. Of course, a train can only leave the network after it has arrived at its final stop. This is encoded by the constraint:

$$done_{tr}^{t_i} \implies \bigvee_{j=0}^{i} occupies_{tr,lastStop(tr)}^{t_j}$$

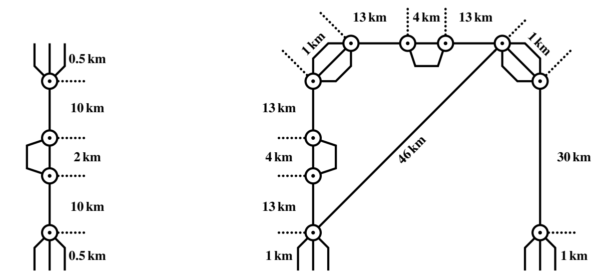
Then, we can define the fact that all trains have reached their goal at time t_i by

$$done^{t_i} := \bigwedge_{tr \in Trains} done_{tr}^{t_i}$$

The number of time steps in which not all trains have arrived at their final stop should be minimized. This can be enforced by adding the objective function

$$\min : \sum_{i=0}^{t_{max}-1} \neg done^{t_i} .$$

Combining all constraints and, if applicable, the respective objective functions from above, a symbolic formulation describing the considered design or verification task is obtained. Passing this formulation to a satisfiability solver (such as [28], [29]) yields two possible outcomes:



(a) Simple layout (b) Complex layout

Fig. 4: Considered railway networks

- The satisfiability solver determines an assignment to all *border*- and *occupies*-variables. Then, a VSS layout as well as the positions of all trains at any moment in time (and, by this, all routes of the trains) can be derived from this assignment. This can show that a given schedule indeed can be realized on a given VSS layout (in case of the verification task) or deliver the desired VSS layout and/or a schedule (in case of the generation or optimization design task).
- Otherwise, the satisfiability solvers proves that no such assignment exists for the given constraints/objectives. Then, it has been proven that no routes of the trains exists which can realize a given schedule (in case of the verification task) or that no VSS layout/train routes exists which can realize a given schedule on the original track layout within the considered time interval.

In all these cases, the designers get their design or verification task solved in a completely automatic fashion.

IV. Application and Case Studies

The methodology proposed above has been implemented in C++/python and using Z3 [28] as a satisfiability solver. Afterwards, we conducted several case studies to evaluate the applicability of the resulting methodology. The implementation as well as the case studies have been made publicly available at https://iic.jku.at/eda/research/ects. In this section, we summarize a selection of those case studies in order to demonstrate the benefits of the proposed methodology. To this end, we considered the following railway networks and corresponding train schedules:

- *Running example*, i. e., the network from Fig. 1 which has been used as running example in this paper.
- *Simple Layout*, i. e., the network and from Fig. 4a which is used as a representative of a layout composed of 3 stations (at the top, middle, and bottom of Fig. 4a).
- *Complex Layout*, i. e., the network from Fig. 4b which represents a somewhat more complex layout composed of a total of 6 stations which are connected differently.
- *Nordlandsbanen*, i. e., a real-life example inspired by the Norwegian Railways network that connects Trondheim and Bodø. The network considered here is composed of 58 train stations and several connections ranging over a total of 822km tracks.

For each of those networks and their corresponding train schedules, we used the methodology proposed above to conduct the design tasks reviewed in Section II-B, i. e., to verify whether the train schedules indeed work on the given (pure) TTD layout, to determine alternative layouts if this is not the case, and/or to additionally optimize the layout in order to improve the train schedule. All case studies have been

TABLE I: Obtained results

Task	Var.	Sat.	TTD/VSS	Time Steps	Runtime [s]
Running Example ($r_t = 0.5\,\mathrm{min}$, $r_s = 0.5$ km)					
Verification	654	No	4	–	0.10
Generation	654	Yes	5	10	0.14
Optimization	654	Yes	7	7	0.25
Simple Layout ($r_t = 1\,\mathrm{min}$, $r_s = 0.5$ km)					
Verification	3910	No	10	–	3.26
Generation	3910	Yes	14	19	7.21
Optimization	3910	Yes	14	15	28.40
Complex Layout ($r_t = 3\,\mathrm{min}$, $r_s = 1$ km)					
Verification	14025	No	22	–	63.33
Generation	14025	Yes	23	17	151.80
Optimization	14025	Yes	25	14	210.70
Nordlandsbanen ($r_t = 5\,\mathrm{min}$, $r_s = 5$ km)					
Verification	21156	No	51	–	62.39
Generation	21156	Yes	53	48	82.65
Optimization	21156	Yes	57	44	79.60

conducted on an Intel Core i7-8550U machine using an 1.80 GHz processor with 16 GB of main memory running Ubuntu 18.04.5.

The results are summarized in Table I. Here, the first two columns refer to the respective design or verification task as well as the respectively required number of Boolean variables which were needed to symbolically formulate it. Afterwards, we list whether the resulting satisfiability instance was shown satisfiable by the solver, how many (TTD and VSS) sections were considered/have been generated, and how many time steps it took to complete the schedule. Finally, the total runtime of the methodology (in CPU seconds) has been listed.

The results clearly confirm the applicability and suitability of the proposed methodology. All verification and design tasks can be solved in an automatic fashion in rather feasible runtime. By this, designers could easily prove that, in all cases, the pure TTD layout was not sufficient to implement the desired schedules (the satisfiability solver did not determined a satisfying assignment for the verification task). Furthermore, the proposed methodology readily provides a solution for this situation: Considering the objective for the second design task, designers can automatically generate a VSS layout which can realize the schedule (as can be seen in the fourth column of Table I, often just a few additional virtual sections where needed to make the schedule work).

Finally, even the schedules themselves could be improved by employing the objective for the third design task. Here, the results nicely show that, by adding some further VSS sections, substantial reductions in the total number of time steps can be achieved. By this, the proposed methodology really unveils the full potential of ETCS Level 3. Recalling that all these tasks have been conducted manually thus far (and, hence, many of these tasks could not be addressed at all due to the underlying complexity), this demonstrates the impact of the contribution.

V. Conclusions

In this work, we proposed an initial methodology which addresses design and verification tasks in context of the ETCS Level 3 in an automatic fashion. To this end, design automation expertise (in terms of satisfiability solvers) has been utilized for the first time. Case studies showed that the resulting methodology allows to address relevant design tasks which, thus far, only have been conducted manually or, due to the underlying complexity, could not been addressed at all. The proposed methodology in terms of a symbolic formulation additionally allows to easily extend the approach, e. g., so that even further design tasks and/or objectives can be considered. By this, the methodology proposed in this work also provides a basis for a further exploitation of design automation in the domain of the design and verification of ETCS.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

[1] I. U. of Railways (UIC), Ed., *Carbon Footprint of Railway Infrastructure*. International Union of Railways (UIC), 2016.

[2] L. Schnieder, *Eine Einführung in das European Train Control System (ETCS)*. Springer Vieweg, 2019.

[3] J. Pachl, *Besonderheiten ausländischer Eisenbahnbetriebsverfahren*, 2019.

[4] ——, *Railway Signalling Principles*, Braunschweig, Jun 2020.

[5] E. A. for Railways, "European rail traffic management system (ERTMS)," https://www.era.europa.eu/activities/european-rail-traffic-management-system-ertms_en, accessed: 2020-09-18.

[6] "Set of specifications 1/2/3," https://www.era.europa.eu/content/ccs-tsi-annex-mandatory-specifications, accessed: 2020-09-18.

[7] P. Stanley and I. of Railway Signal Engineers, *ETCS for Engineers*, 2011.

[8] Alcatel, Alstom, A. Signal, Bombardier, I. Rail, and Siemens, *System Requirements Specification*. European Union Agency for Railways, 2006.

[9] D. Dghaym, M. Poppleton, and C. Snook, "Diagram-led formal modelling using iUML-B for Hybrid ERTMS Level 3," in *ABZ*, 2018.

[10] D. Dghaym, S. Dalvandi, M. Poppleton, and C. Snook, "Formalising the hybrid ERTMS level 3 specification in iUML-B and Event-B," *International Journal on Software Tools for Technology Transfer*, vol. 22, Jun 2020.

[11] A. Cunha and N. Macedo, "Validating the Hybrid ERTMS/ETCS Level 3 concept with Electrum," in *ABZ*, 2018.

[12] S. J. Tueno Fotso, M. Frappier, R. Laleau, and A. Mammar, "Modeling the Hybrid ERTMS/ETCS Level 3 standard using a formal requirements engineering approach," in *ABZ*, 2018.

[13] J.-R. Abrial, "The ABZ-2018 case study with Event-B," in *ABZ*, 2018.

[14] A. Mammar, M. Frappier, S. J. Tueno Fotso, and R. Laleau, "An Event-B model of the Hybrid ERTMS/ETCS Level 3 standard," in *ABZ*, 2018.

[15] P. Arcaini, P. Ježek, and J. Kofroň, "Modelling the Hybrid ERTMS/ETCS Level 3 case study in Spin," in *ABZ*, 2018.

[16] Hoang, Thai Son and Butler, Michael and Reichl, Klaus, "The Hybrid ERTMS/ETCS Level 3 case study," in *ABZ*, 2018.

[17] D. Hansen, M. Leuschel, D. Schneider, S. Krings, P. Körner, T. Naulin, N. Nayeri, and F. Skowron, "Using a formal B model at runtime in a demonstration of the ETCS Hybrid Level 3 concept with real trains," in *ABZ*, 2018.

[18] J. Jansen, E. Quaglietta, M. Bartholomäus, A. Pot, and R. Goverde, "ETCS Hybrid Level 3: A simulation-based impact assessment for the Dutch railway network."

[19] D. Gill, "ETCS Level 3 for metro-type mainline operation," in *Aspect*, 2017.

[20] G. Theeg and S. Vlasenko, Eds., *Railway Signalling & Interlocking: International Compendium*. Eurailpress, 2009.

[21] S. M. GmbH, *X2Rail-1 Deliverable 5.1 Moving Block System Specification*. Shift2Rail, 2019.

[22] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, vol. 1579. Springer Verlag, 1999, pp. 193–207.

[23] R. Wille, D. Große, F. Haedicke, and R. Drechsler, "SMT-based stimuli generation in the SystemC verification library," in *Forum on Specification and Design Languages*, 2009, pp. 1–6.

[24] S. Eggersglüß, R. Wille, and R. Drechsler, "Improved SAT-based ATPG: more constraints, better compaction," in *Int'l Conf. on CAD*, 2013, pp. 85–90.

[25] A. Biere, M. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*. IOS Press, 2009.

[26] B. Luteberget and C. Johansen, "Efficient verification of railway infrastructure designs against standard regulations," *Formal Methods in System Design*, vol. 52, pp. 1–32, 2018.

[27] B. Luteberget, K. Claessen, and C. Johansen, "Design-time railway capacity verification using SAT modulo discrete event simulation," in *Int'l Conf. on Formal Methods in CAD*, 2018.

[28] L. M. de Moura and N. Bjørner, "Z3: an efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[29] R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler, "SWORD: A SAT like prover using word level information," in *VLSI-SoC: Advanced Topics on Systems on a Chip: A Selection of Extended Versions of the Best Papers of the Fourteenth International Conference on Very Large Scale Integration of System on Chip*, R. Reis, V. Mooney, and P. Hasler, Eds. Springer, 2009, pp. 175–192.