HAL open science

Strategy Synthesis for Autonomous Driving in a Moving Block Railway System with Uppaal Stratego

Davide Basile, Maurice Beek, Axel Legay

To cite this version:

Davide Basile, Maurice Beek, Axel Legay. Strategy Synthesis for Autonomous Driving in a Moving Block Railway System with Uppaal Stratego. 40th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE), Jun 2020, Valletta, Malta. pp.3-21, 10.1007/978-3-030-50086-3_1 . hal-03283240

HAL Id: hal-03283240

https://inria.hal.science/hal-03283240v1

Submitted on 9 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Strategy Synthesis for Autonomous Driving in a Moving Block Railway System with Uppaal Stratego

Davide Basile[1], Maurice H. ter Beek[1], and Axel Legay[2]

[1] ISTI–CNR, Pisa, Italy
{davide.basile,maurice.terbeek}@isti.cnr.it
[2] Université Catholique de Louvain, Louvain-la-Neuve, Belgium
axel.legay@uclouvain.be

Abstract. Moving block railway systems are the next generation signalling systems currently under development as part of the Shift2Rail European initiative, including autonomous driving technologies. In this paper, we model a suitable abstraction of a moving block signalling system with autonomous driving as a stochastic priced timed game. We then synthesise safe and optimal driving strategies for the model by applying advanced techniques that combine statistical model checking with reinforcement learning as provided by Uppaal Stratego. Hence, we show the applicability of Uppaal Stratego in this concrete case study.

1 Introduction

Next generation railway systems are based on distributed inter-organisational entities, such as on-board train computers and wayside radio-block centres and satellites, which have to interact to accomplish their tasks. A longstanding effort in the railway domain concerns the use of formal methods and tools for the analysis of railway (signalling) systems in light of the sector's stringent safety requirements [27,29,30,17,28,11,41,42,7,10,31]. Due to their distributed and inter-organisational nature, their formal verification is still an open challenge. Whilst model-checking and theorem-proving techniques are predominant, to the best of our knowledge, applications of controller synthesis techniques are largely lacking.

We describe a formal modelling and analysis experience with Uppaal Stratego of a moving block railway signalling system. This work was conducted in the context of several projects concerned with the use of formal methods and tools for the development of railway systems based on moving block signalling systems, in which train movement is no longer authorised based on sections of the railway track between fixed points, but computed in real time as safe zones around the trains. Most notably, the H2020 Shift2Rail projects AST-Rail: SAtellite-based Signalling and Automation SysTems on Railways along with Formal Method and Moving Block Validation (http://www.astrail.eu) and 4SECURail: FORmal Methods and CSIRT for the RAILway sector (http://www.4securail.eu). The European Shift2Rail initiative (http://shift2rail.org) is

a joint undertaking of the European Commission and the main railway stakeholders to move the European railway industry forward by increasing its competitiveness. This concerns in particular the transition to next generation signalling systems, including satellite-based train positioning, moving-block distancing, and automatic driving. With a budget of nearly 1 billion euro, it is unique in its kind.

Previously, in [8,6], we introduced a concrete case study of a satellite-based moving block railway signalling system, which was developed in collaboration with industrial partners of the ASTRail project and which was modelled and analysed with Simulink and Uppaal SMC (Statistical Model Checker). While those models offered the possibility to fine tune communication parameters that are fundamental for the reliability of their operational behaviour, they did not account for the synthesis of autonomous driving strategies.

Building on such efforts, in this paper we present a formal model of a satellite-based moving block railway signalling system, which accounts for autonomous driving and which is modelled in Uppaal Stratego as a stochastic priced timed game. The autonomous driving module is not modelled manually, but it is synthesised automatically as a strategy, after which both standard and statistical model checking are applied under the resulting (safe) strategy. The starting point for deriving the strategy is a safety requirement that the model must respect. We moreover consider reliability aspects, and the autonomous driving strategy also provides guarantees for the minimal expected arrival time. The model and experiments are available at https://github.com/davidebasile/FORTE2020.

Related work At last year's FORTE, parametric statistical model checking was applied to Unmanned Aerial Vehicles (UAV) [4]. The model was formalised as a parametric Markov chain with the goal of reducing the probability of failure while varying parameters such as precision of the position. The UAV follows a predefined flight plan, whereas we aim at automatically synthesising a strategy to safely drive the train. It would be interesting to investigate the possibility of synthesising flight plans under safety constraints.

A decade ago at FORTE'10, one of the first applications of statistical model checking (using the BIP toolset) to an industrial case study was presented, namely the heterogeneous communication system for cabin communication in civil airplanes [9]. The goal was to study the accuracy of clock synchronisation between different devices running in parallel on a distributed application, i.e. a time bound within which communication must occur. An implementation of this case study in Uppaal SMC would allow a comparison of the results.

Statistical model checking has also been used to verify the reliability of railway interlocking systems [19] and Uppaal has been used to verify railway timetables [34]. Uppaal Stratego has been applied to a few other case studies belonging to the transport domain, such as traffic light controllers [3], cruise control [38], and railway scheduling [37]. We conjecture that the Uppaal Stratego model in [37] could be paired with our model to study railway scheduling for autonomous trains, with the goal of synthesising improved strategies for both the scheduler and the autonomous driver.

Finally, there have been several recent attempts at modelling and analysing ERTMS Level 3 signalling systems (in particular Hybrid Level 3 systems with virtual fixed blocks) with Promela/Spin, mCRL2, Alloy/Electrum, iUML, SysML, ProB, Event-B, and real-time Maude [43,2,5,21,25,40,14,33]. None of these concern quantitative modelling and analysis, typically lacking uncertainty, which is fundamental for demonstrating the reliability of the operational behaviour of next generation satellite-based ERTMS Level 3 moving block railway signalling system models. One of the earliest quantitative evaluations of moving block railway signalling systems can be found in [36], based on GSM-R communications.

Structure of the paper After some background on Uppaal Stratego in Sect. 2, we describe the setting of the case study from the railway domain in Sect. 3. In Sect. 4, we present the formal model, followed by an extensive description of the conducted analyses in Sect. 5. Finally, we discuss our experience with Uppaal Stratego and provide some ideas for future work in Sect. 6.

2 Background: Uppaal Stratego

In this section, we provide some background of the tools and their input models used in this paper, providing pointers to the literature for more details.

Uppaal Stratego [24] is the latest tool of the Uppaal [12] suite. It integrates formalisms and algorithms coming from the less recent Uppaal Tiga [13] (synthesis for timed games), Uppaal SMC [22] (Statistical Model Checking), and the synthesis of near optimal schedulers proposed in [23].

Uppaal Tiga [13,20] implements an efficient on-the-fly algorithm for the synthesis of strategies extended to deal with models of *timed games*. These are automata modelling a game between a player (the controller) and an opponent (the environment). Transitions are partitioned into controllable and uncontrollable ones. The controller plays the controllable transitions, while the opponent plays the uncontrollable ones. The controller is only allowed to deactivate controllable transitions. The goal is to synthesise a strategy for the controller such that, no matter the actions of the opponent, a particular property is satisfied. Generally, uncontrollable transitions are used to model events such as delays in communication or other inputs from the environment. On the converse, controllable transitions characterise the logic of the controller, generally related to actuators. The strategy synthesis algorithm uses a suitable abstraction of the real-time part of the model, through *zones* that are constraints over the real-time clocks. Strategy synthesis allows an algorithmic construction of a controller which is guaranteed to ensure that the resulting system satisfies the desired correctness properties, i.e. reachability and safety.

Uppaal SMC is a statistical model checker based on models of *stochastic timed automata*. These are automata enhanced with real-time modelling through *clock* variables. Moreover, their stochastic extension replaces non-determinism with probabilistic choices and time delays with probability distributions (uniform

for bounded time and exponential for unbounded time). These automata may communicate via (broadcast) channels and shared variables. Statistical Model Checking (SMC) [39,1] is based on running a sufficient number of (probabilistic) simulations of a system model to obtain statistical evidence (with a predefined level of statistical confidence) of the quantitative properties to be checked. SMC offers advantages over exhaustive (probabilistic) model checking. Most importantly, SMC scales better since there is no need to generate and possibly explore the full state space of the model under scrutiny, thus avoiding the combinatorial state-space explosion problem typical of model checking, and the required simulations can be easily distributed and run in parallel. This comes at a price: contrary to (probabilistic) model checking, exact results (with 100% confidence) are out of the question.

The method proposed in [23] extends the strategy synthesis of [13] to find near-optimal solutions for *stochastic priced timed games*, which are basically stochastic timed automata enhanced with controllable and uncontrollable transitions, similarly to timed games. In short, the method starts from the most permissive strategy guaranteeing the time bounds, computed with the algorithms in [13]. This strategy is then converted into a stochastic one by substituting non-determinism with uniform distributions. Finally, reinforcement learning is applied iteratively to learn from sampled runs the effect of control choices, to find the near-optimal strategy.

Uppaal Stratego uses stochastic priced timed games as formalism whilst integrating (real-time) model checking, statistical model checking, strategy synthesis, and optimisation. It thus becomes possible to perform model checking and optimisation under strategies, which are first-class objects in the tool. Internally, abstractions that allow to pass from stochastic priced timed games to timed games similar to those in [13] are used to integrate the various algorithms.

3 Context and Case Study

The European Railway Traffic Management System (ERTMS) is a set of international standards for the interoperability, performance, reliability, and safety of modern European rail transport [26]. It relies on the European Train Control System (ETCS), an automatic train protection system that continuously supervises the train, ensuring to not exceed the safety speed and distance. The current standards distinguish four levels (0–3) of operation of ETCS signalling systems, depending largely on the role of trackside equipment and on the way information is transmitted to and from trains. The ERTMS/ETCS signalling systems currently deployed on railways throughout Europe concern at most Level 2.

Level 2 signalling systems are based on fixed blocks starting and ending at signals. The block sizes are determined based on parameters like the speed limit, the train's speed and braking characteristics, drivers' sighting and reaction times, etc. But the faster trains are allowed to run, the longer the braking distance and the longer the blocks need to be, thus decreasing the line's capacity. This is

because the railway sector's stringent safety requirements impose the length of fixed blocks to be based on the worst-case braking distance, regardless of the actual speed of the train. For exact train position detection and train integrity supervision, Level 2 signalling systems make use of trackside equipment (such as track circuits or axle counters). However, communication of the movement authority (MA), i.e. the permission to move to a specific location with supervision of speed, as well as speed information and route data to and from the train is achieved by continuous data transmission via GSM-R or GPRS with a wayside radio block centre. Moreover, an onboard unit continuously monitors the transferred data and the train's maximum permissible speed by determining its position in between the Eurobalises (transponders on the rails of a railway) used as reference points via sensors (axle transducers, accelerometer and radar).

The next generation Level 3 signalling systems currently under investigation and development, no longer rely on trackside equipment for train position detection and train integrity supervision. Instead, an onboard odometry system is responsible for monitoring the train's position and autonomously computing its current speed. The onboard unit frequently sends the train's position to a radio block centre which, in turn, sends each train a MA, computed by exploiting its knowledge of the position of the rear end of the train ahead. For this to work, the precise absolute location, speed, and direction of each train needs to be known, which are to be determined by a combination of sensors: active and passive markers along the track, and trainborne speedometers. The resulting moving block signalling systems allow trains in succession to close up, since a safe zone around the moving trains can be computed, thus considerably reducing headways between trains, in principle to the braking distance. This allows for more trains to run on existing railway tracks, in response to the ever-increasing need to boost the volume of passenger and freight rail transport and the cost and impracticability of constructing new tracks. Furthermore, the removal of trackside equipment results in lower capital and maintenance costs [32].

One of the current challenges in the railway sector is to make moving block signalling systems as effective and precise as possible, including satellite-based positioning systems and leveraging on an integrated solution for signal outages (think, e.g., of the absence of positioning in tunnels) and the problem of multipaths [44]. However, due to its robust safety requirements the railway sector is notoriously cautious about adopting technological innovations. Thus, while GNSS-based positioning systems are in use for some time now in the avionics and automotive sectors, current train signalling systems are still based on fixed blocks. However, experiments are being conducted and case studies are being validated in order to move to Level 3 signalling systems [15,2,5,21,25,40,6,14,8,33].

The components of the moving block railway signalling case study considered in this paper are depicted in Fig. 1. The train carries the location unit and onboard unit components, while the radio block centre is a wayside component. The location unit receives the train's location from GNSS satellites, sends this location (and the train's integrity) to the onboard unit, which, in turn, sends the location to the radio block centre. Upon receiving a train's location, the radio

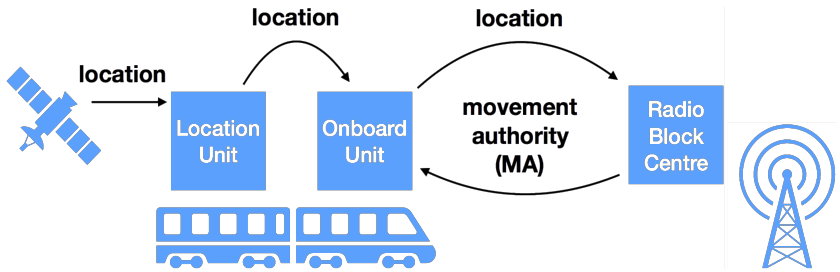

Fig. 1. ERTMS Level 3 moving block railway signalling (adapted from [8,31])

block centre sends a MA to the onboard unit (together with speed restrictions and route configurations), indicating the space the train can safely travel based on the safety distance with preceding trains. The radio block centre computes such MA by communicating with neighbouring radio block centres and exploiting its knowledge of the positions of switches and other trains (head and tail position) by communicating with a Route Management System. We abstract from the latter and from communication among neighbouring radio block centres: we consider one train to communicate with one radio block centre, based on a seamless handover when the train moves from one radio block centre supervision area to an adjacent one, as regulated by its Functional Interface Specification [45].

4 Formal Model

In this section, we describe the formal model of the case study introduced before. It consists of a number of SPTGs, which are basically timed automata with prices (a cost function) and stochasticity, composed as a synchronous product.

We briefly describe the model's components, followed by details of the onboard unit. Delays in the communications are exponentially distributed with rate `1:4` to account for possible delays. This is a common way of modelling communication delays. Moreover, all transitions are uncontrollable, except for the controllable actions of the driver in the `TRAIN_ATO_T` component, which are used to synthesise the safe and optimal strategy.

Component `OBU_MAIN_GenerateLocationRequest_T` initiates system interactions by generating a request for a new location to send to the location unit. The location unit component `LU_MAIN_T` receives a new position request from the onboard unit, replying with the current train location (computed via GNSS). The main component `OBU_MAIN_SendLocationToRBC_T` of the onboard unit performs a variety of operations. It receives the position from the location unit, sends the received position to the radio block centre, and eventually implements a safety mechanism present in the original system specification. In particular, at each instant of time, it checks that the train's position does not exceed the MA received from the radio block centre; if it does, it enters a failure state. The failure is also

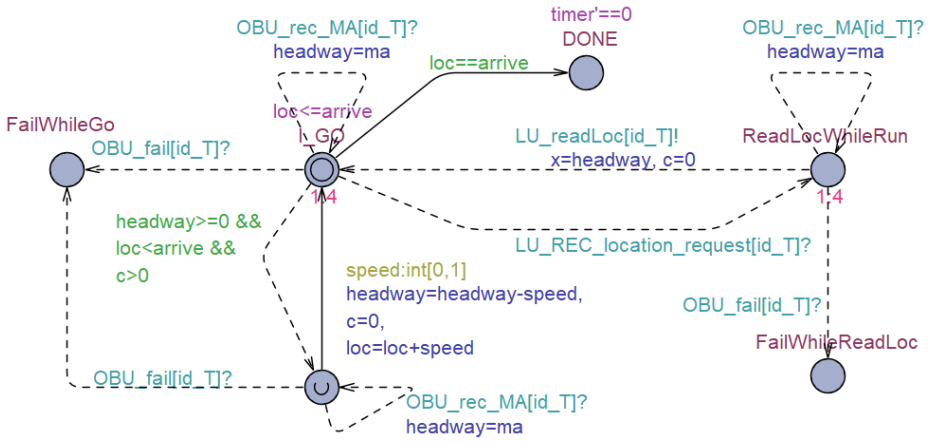

Fig. 2. The `TRAIN_ATO_T` component

broadcast to all other components. In fact, all components can enter a failure state if a failure is triggered. The `RBC_Main_T` component model of the radio block centre receives MA requests from the onboard unit. Once received, the radio block centre repeatedly sends a MA message until the corresponding acknowledgement from the onboard unit is received. Also `OBU_MAIN_ReceiveMA_T` models the logic of the onboard unit. It receives a MA from the radio block centre, and sends back a corresponding acknowledgement. Finally, the `TRAIN_ATO_T` component was defined to synthesise a strategy for moving the train in a safe and optimal way. In particular, the position of the train (variable `loc`) is stated in a unidimensional space and identified by one coordinate (representing the position along its route), and the train is allowed at each cycle to either move one unit or stay idle. To allow state-space reduction, the value of `loc` represents a range of the space in which the train is located, rather than a specific point in space. Next, we describe this component, depicted in Fig. 2, in detail.

The initial state of `TRAIN_ATO_T` is the nominal state `I_GO`, drawn with two circles. Two failure states (`FailWhileGo` and `FailWhileReadLoc`) are reached in case the MA is exceeded in `OBU_MAIN_SendLocationToRBC_T`. The initial state has an invariant to guarantee that the train has not passed its destination. Note that invariants can be constraints on clocks or variables. This is done by checking that the location of the train, which is encoded by the integer variable `loc`, is less than or equal to the integer constant `arrive`, which is an input parameter of the model to perform experiments. From the initial state it is possible to transit to state `ReadLocWhileRun`, upon a location request coming from `LU_MAIN_T`, and coming back from `ReadLocWhileRun` to `I_GO` by replying to such a request. Variable `x` is a buffer for value-reading messages. To reduce the model's state space, the value transmitted by `TRAIN_ATO` is the remaining headway, i.e. the difference between the MA and the location. Indeed, such value has a fixed range if compared to the location (under the assumption that the arrival point is greater

than the initial headway value, otherwise the train will never exceed its MA before arriving to the destination). In turn, `OBU_MAIN_SendLocationToRBC_T` checks if such transmitted value (`headway`) is negative for triggering a failure, since in that case the train has exceeded its MA.

From both states `I_GO` and `ReadLocWhileRun`, an inner loop is used to receive the new MA (`movaut`) from `RBC_Main_T`. The `movaut` should be relative to the current location `loc` of the train, i.e. `movaut = loc +` fixed number of meters that the train is allowed to travel. However, to reduce the state space, such a message simply resets the `headway` variable to its initial value, which is an integer constant called `ma`. Thus, `movaut` is not stored in the state space because its value can be retrieved as `loc+ma`. The constant `ma` is another input parameter of the model. The reason such a loop is also present in state `ReadLocWhileRun` is that otherwise the MA message would be lost in this state, and similarly for the urgent state (marked with the symbol U, described below).

We now discuss the two controllable transitions in the model. The first is used to move a train. In Uppaal Stratego a controller cannot delay its actions (whereas the environment can), hence the movement of the train is split into an uncontrollable transition followed by a controllable one, with an intermediate urgent state. An intermediate urgent state is such that a transition must be taken without letting time pass. This is a workaround to force the controller to perform an action at that instant of time. From the initial state `I_GO`, an uncontrollable transition targeting the urgent state is used to check that the conditions for moving the train are met. In particular, if the headway is non-negative and the train has not arrived, the transition for moving the train is enabled. Additionally, a test `c>0` on the clock `c` is used to forbid Zeno behaviour. Indeed the clock `c` is reset to zero after the train has moved, hence time is forced to pass before the next movement. Such a condition cannot be stated directly on the controllable transition, otherwise a time-lock (i.e. time is not allowed to pass) would be reached in case the condition is not met.

The controllable transition (drawn as a solid arc) from the urgent state can either set the integer speed to 1 or to 0, allowing the train to proceed to the next interval of space or to remain in the previous interval, respectively. Recall that `loc` is not a coordinate but rather an abstraction of a portion of space. The controllable transition also updates the headway. To reduce the state space, the only negative value allowed for the variable `headway` is -1.

Finally, the second controllable transition is used to reach a sink state `DONE`. To further reduce the state space, the train is not allowed to move once `loc` has reached value `arrive`. A hybrid clock `timer` is used as a stop-watch to measure the time it takes for the train to arrive in state `DONE`. To this aim, the invariant `timer'==0` in state `DONE` sets the derivative of clock `timer` to zero. A hybrid clock can be abstracted away during the synthesis of a safe strategy.

5 Formal Analysis and Experiments

In this section, we report on analysis of the formal model. The main objective is to synthesise a safe strategy such that the train does not exceed the MA. Additionally, the train should be as fast as possible, within the limits imposed by the safety requirements. To this aim, an optimal and safe strategy is synthesised.

The experiments were carried out on a machine with a processor Intel(R) Core(TM) i7-8500Y CPU at 1.50GHz, 1601Mhz, 2 cores, and 4 logical processors with 16GB of RAM, running 64bit Windows 10. The development version of Uppaal Stratego (academic version 4.1.20-8-beta2) was used. Indeed, when developing the model and its analysis, minor issues were encountered (more later). This version of Uppaal Stratego contains some patches resulting from a series of interactions between the first author and the developers team at Aalborg University.

The set-up of the parameters of the statistical model checker was chosen to provide a good confidence in the results and is as follows: probabilistic deviation set to $\delta = 0.01$, probability of false negatives and false positives set to $\alpha = 0.005$ and $\beta = 0.5$, respectively, and the probability uncertainty set to $\epsilon = 0.005$. As anticipated in Sect. 3, we focussed on one radio block centre, one onboard unit, and one location unit, i.e. one train communicating with one radio block centre. Finally, we set `ma` $= 5$ and `arrive` $= 20$.

To begin with, we want to check if the hazard of exceeding the MA is possible at all in our model. If such a hazard were never possible, the safe strategy would simply allow all behaviour. To analyse this, we perform standard model checking:

```
A[] not (OBU_MAIN_SendLocationToRBC.MAexceededFailure)
```

This formula checks if for every possible path, the state `MAexceededFailure` of the component `OBU_MAIN_SendLocationToRBC` is never visited. Indeed, this particular state is reached exactly when the hazard occurs, i.e. the MA is exceeded, thus triggering a failure. After 0.016s, using 38,200KB of memory, Uppaal Stratego reports that this formula is not satisfied, thus such hazard is possible without a proper strategy to drive the train. We would like to check the likelihood to reach this failure, given this specific set-up of parameters. First, the average maximal time in which the train reaches its destination is computed. This is important to fine tune the time bound for further simulations. To do so, we use the statistical model checker to evaluate the following formula:

$$\phi_1 = \texttt{E[<=700;10000](max: TRAIN_ATO.timer)}$$

This formula computes the average maximal value of the `TRAIN_ATO.timer` stopwatch, i.e. the arrival time. It is computed based on 10000 simulations with an experimental time bound of 700 seconds. The computed value is 377.235 ± 3.960, and its probability distribution is depicted in Fig. 3. By analysing the probability distribution it is possible to notice that the average value is lower if faults are ignored. Indeed, in case of faults the value of `timer` is equal to the end of the

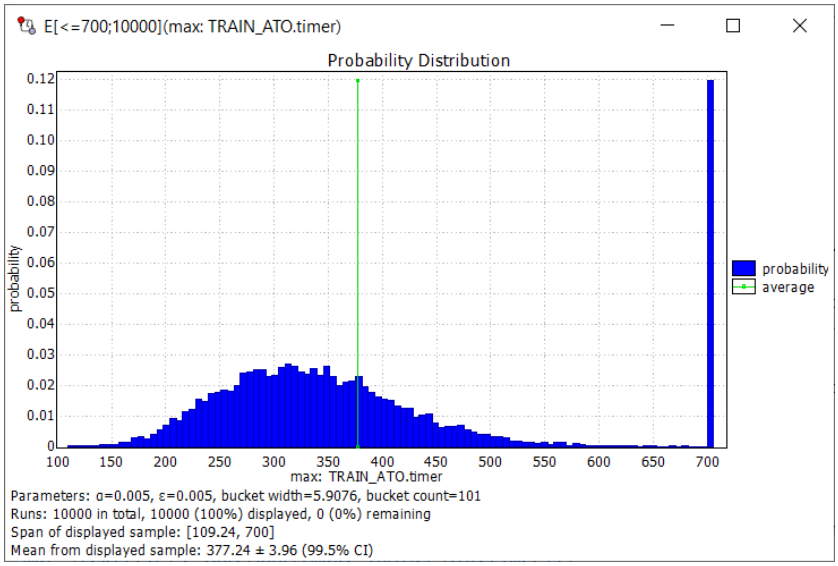

Fig. 3. Probability distribution for average maximal arrival time (ϕ_1)

simulation (i.e. 700 seconds). Hence, the time bound for the following simulations is set to 500 seconds, thus considering also worst cases of arrival time.

We now compute the likelihood of the model to reach a failure, using SMC to measure the probability of reaching the failure state with the following formula:

$$\phi_2 = \texttt{Pr[<=500] (<>OBU_MAIN_SendLocationToRBC.MAexceededFailure)}$$

Uppaal Stratego executes 33952 simulations and the probability is within the range $[0.117029, 0.127029]$, with confidence 0.995. The probability confidence interval plot for this experiment is depicted in Fig. 4. We conclude that, for this set-up of parameters, there is a relatively high probability for this hazard to occur. This is as expected, due to the absence of a strategy for driving the train and the non-deterministic choice of whether or not to move the train.

After these standard and statistical model-checking experiments, we exploit the synthesis capabilities of Uppaal Stratego to automatically fix the specification to adhere to safety constraints. Indeed, no manual intervention to fix the model is needed: it suffices to compute a driving strategy and compose it with the model. Recall that the only controllable transitions in the model are those for deciding whether or not to move the train (i.e. related to acceleration/deceleration, accordingly). This in turn depends on the stochastic delays in communication. The strategy prunes controllable transitions such that those previously reachable configurations leading to the failure state are no longer reachable. To compute

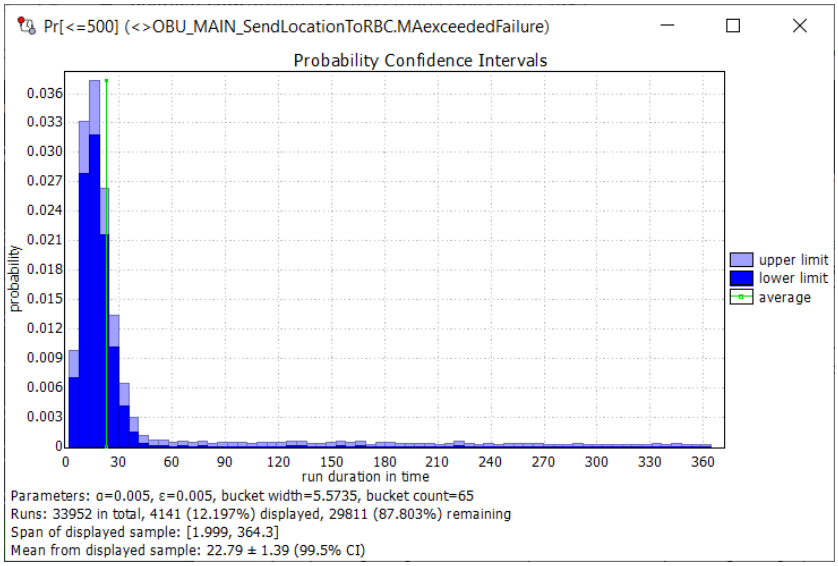

Fig. 4. Probability confidence interval for likelihood of reaching a failure (ϕ_2)

this strategy, called `safe`, we evaluate the following formula:

```
strategy safe = control :
        A[] not (OBU_MAIN_SendLocationToRBC.MAexceededFailure)
```

Uppaal Stratego successfully computes the `safe` strategy in 7.167s, using 576,888KB of memory. The strategy allows all possible behaviour that does not violate the above property. To have a glimpse of the strategy at work, we ran 50 simulations of the model composed with the `safe` strategy to visualise the variable `TRAIN_ATO.loc` (i.e. the train's location) with the following command:

```
simulate 50 [<=500] TRAIN_ATO.loc under safe
```

Figure 5 shows the trajectory of variable `TRAIN_ATO.loc` for 50 simulations, computed in 0.147s, using 576,984KB. We see that in all trajectories the train never stops before reaching its destination, i.e. no failure occurs. However, in some simulations the train is relatively slower, when compared to other simulations.

Uppaal Stratego also allows to model check the synthesised strategies. We ran a full state-space exploration by means of standard model checking to formally verify that after composing the model with the `safe` strategy the hazard of exceeding the MA is mitigated. This is checked through the following formula:

```
A[] not (OBU_MAIN_SendLocationToRBC.MAexceededFailure) under safe
```

This formula checks that in the model composed with the `safe` strategy, the 'bad' state is never reached. After 2.283s and using 599,268KB of memory, Uppaal Stratego reports that the formula is satisfied, thus confirming that we

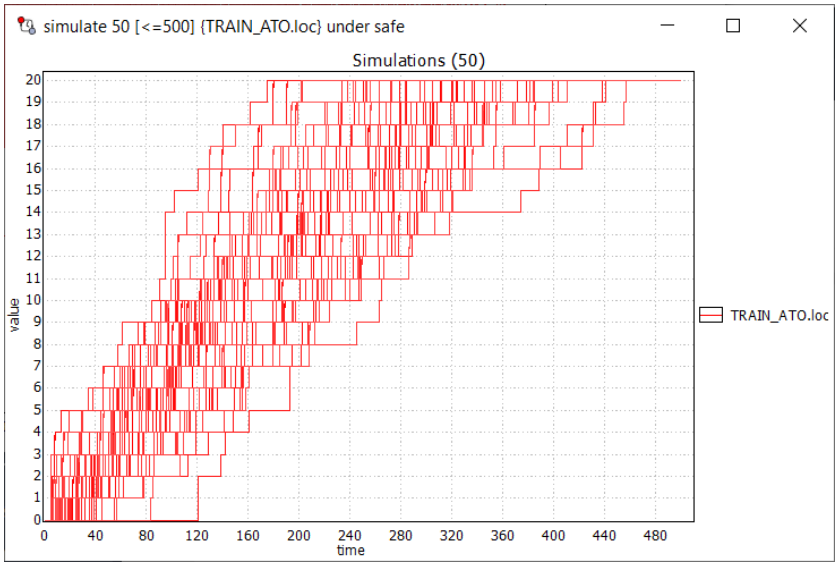

Fig. 5. Simulations of the model under the safe strategy `safe`

automatically synthesised a strategy for mitigating the hazard. However, even if not showed in Fig. 5, there exist trajectories in the composition where the train never reaches its destination. This can be formally proven with a full state-space exploration of the strategy by standard model checking of the following formula:

```
A<> (TRAIN_ATO.DONE) under safe
```

This formula checks that in all paths eventually state `TRAIN_ATO.DONE` is reached (i.e. the train reached its destination). After 0.053s, using 599,268KB of memory, Uppaal Stratego reports that the formula does not hold. Indeed, as expected, the strategy does not guarantee that such a state is always reached, but it only guarantees to avoid state `OBU_MAIN_SendLocationToRBC.MAexceededFailure`. For example, there exists also a safe strategy that allows the train to remain in its starting position.

To evaluate the probability to reach state `TRAIN_ATO.DONE` under the `safe` strategy, we ran the statistical model checker to evaluate the following formula:

$$\phi_3 = \texttt{Pr[<=500] (<>TRAIN_ATO.DONE) under safe}$$

Uppaal Stratego executes 10617 runs and estimates the probability to be in the interval $[0.960561, 0.970561]$ with confidence 0.995. The probability distribution of this formula is depicted in Fig. 6. We conclude that the likelihood for the train to not reach its destination within 500 time units under the `safe` strategy is low, and it is mainly due to the possibility of large delays in communications. These delays are indeed the only source of stochastic behaviour in the model.

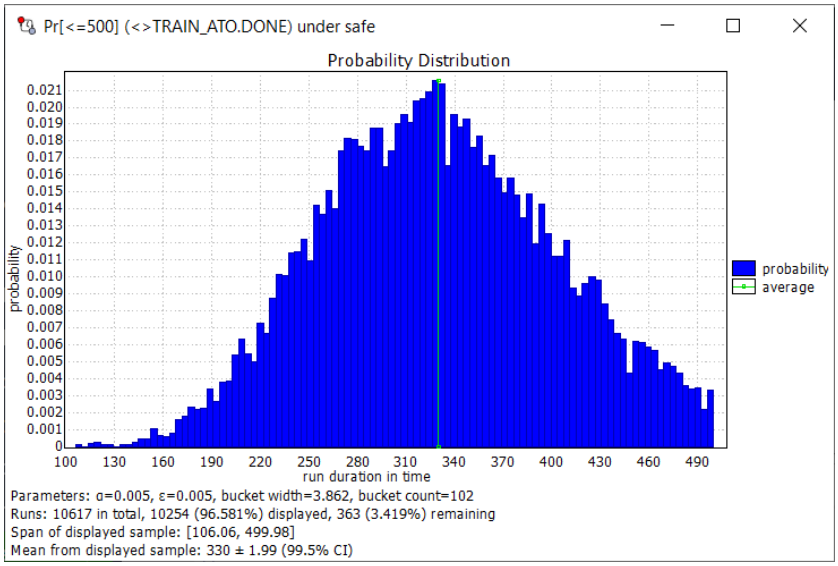

Fig. 6. Probability distribution for reaching the final destination under `safe` (ϕ_3)

We now show how Uppaal Stratego can account for dependability parameters other than safety. In particular, reliability of the system can be related to the capacity of the train to reach its destination quickly. We optimise the safe strategy to minimise the arrival time, thus increasing its reliability whilst satisfying safety. This can be done with the following query, computed in 0.015s using 580,844KB of memory:

```
strategy optsafe = minE (TRAIN_ATO.timer) [<=500] :
                                <> (TRAIN_ATO.DONE) under safe
```

This query creates a new strategy, called `optsafe`, that optimises the strategy `safe` by minimising the average value of the hybrid clock `timer` within 500 time units. Recall that this hybrid clock is used to measure the train's arrival time. The obtained strategy is thus both safe and it has an optimal speed for the train. To check this last improvement, we measure the average maximal arrival time under the strategies `safe` and `optsafe` with the following queries, respectively:

$\phi_4 =$ `E[<=700;10000](max: TRAIN_ATO.timer) under safe`

$\phi_5 =$ `E[<=700;10000](max: TRAIN_ATO.timer) under optsafe`

In particular, both queries run 10000 simulations with time bound of 700 time units. For the `safe` strategy, the estimated value is 338.473 ± 2.264. For the `optsafe` strategy, the estimated value is 331.362 ± 2.250. As expected, the optimised safe strategy has improved the arrival time of the safe strategy. The probability distribution of query ϕ_5 is depicted in Fig. 7.

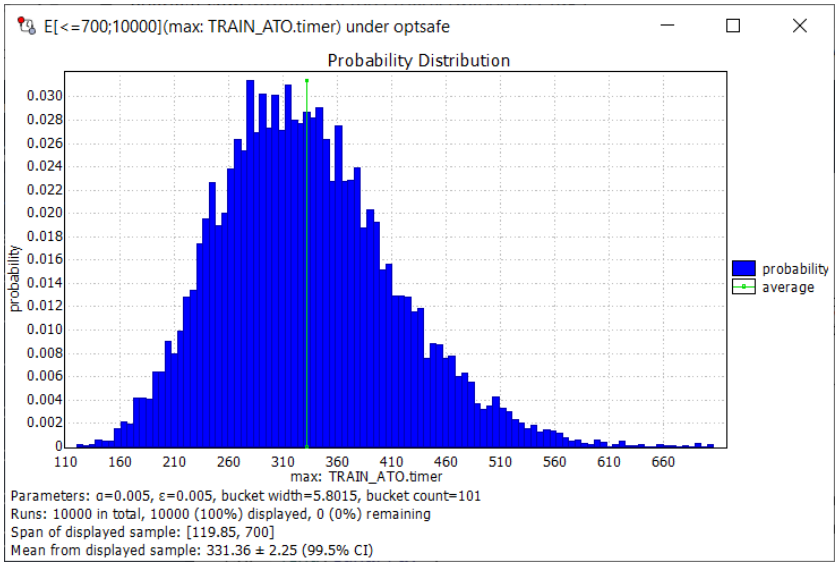

Fig. 7. Probability distribution for average maximal arrival time under `optsafe` (ϕ_5)

Sensitive analysis of maximal headway Up to this point, we evaluated the moving block railway signalling system under analysis with a specific parameter set-up. In this set-up, each time the train receives a fresh MA, its headway is reset to 5 (i.e. $\mathtt{ma} = 5$). Thus, this is the maximal possible headway.

The parameters of the model can be tuned in such a way that the analysed properties are within a desired range of values. In particular, we hypothesise that reducing the maximal headway (i.e. `ma`) results in a deterioration in performance of the optimal strategy and in an increment of the probability of reaching a failure without strategy. Indeed, with a tight headway, the train is forced to move slowly in order not to exceed its MA. In the remainder of this section, we experimentally verify our hypothesis. Table 1 reports the evaluation of properties ϕ_1–ϕ_5 in three different experiments, with values for `ma` taken from the set $\{3, 5, 10\}$, reporting also the computation times and, where appropriate, the number of runs.

By reducing the maximal headway (i.e. $\mathtt{ma} = 3$), we notice an overall deterioration of the average maximal arrival time (cf. properties ϕ_1, ϕ_4, and ϕ_5). Moreover, without strategy the probability of failure is higher when compared to $\mathtt{ma} = 5$ (cf. property ϕ_2). These results confirm our hypothesis and further corroborate the reliability of our model.

As a final experiment, we enlarged the maximal headway (i.e. $\mathtt{ma} = 10$) to evaluate the improvement in performance in case of a larger headway. We recall that a large headway is not desirable, since it would result in a lower capacity of the railway network. In this experiment, the values of ϕ_2 and ϕ_3 are similar to the case of $\mathtt{ma} = 5$. However, by observing the values of ϕ_1, ϕ_4, and ϕ_5, we note that there is only a slight improvement in arrival time, even if we have doubled

Table 1. Three sets of experiments obtained by varying the `ma` parameter

property / *headway*	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5
`ma` $= 3$	402.869 $\pm$4.449 14.7s 708,084KB	47683 runs $[0.179862, 0.189862]$ 0.3s 702,676KB	11197 runs $[0.958596, 0.968595]$ 0.078s 823,396KB	346.715 $\pm$2.189 28.25s 823,580KB	338.470 $\pm$2.207 36.899s 823,580KB
`ma` $= 5$	377.235 $\pm$3.960 14.518s 38,200KB	33952 runs $[0.117029, 0.127029]$ 39.124s 32,856KB	10617 runs $[0.960561, 0.970561]$ 17.003s 580,764KB	338.473 $\pm$2.264 25.99s 580,844KB	331.362 $\pm$2.250 34.241s 580,868KB
`ma` $= 10$	374.482 $\pm$3.935 14.722s 29,848KB	32685 runs $[0.111755, 0.121755]$ 0.172s 30,232KB	9520 runs $[0.964259, 0.974258]$ 0.187s 2,301,436KB	337.087 $\pm$2.274 26.998s 2,301,504KB	329.841 $\pm$2.282 34.172s 2,301,688KB

the maximal headway. This experiment confirms our intuition that an excessive increment of the maximal headway does not lead to a better performance. This is because the train cannot go faster than its optimal speed. On the converse, an excessive enlargement of the headway results in a deterioration of the overall track capacity. Hence, `ma` $= 5$ is a satisfactory set-up for the maximal headway.

6 Conclusion and Future Work

We have modelled and analysed an autonomous driving problem for a moving block railway signalling system. Communication between the train and the radio block centre are modelled such that the train is allowed to proceed only within the limits imposed by the radio block centre via the MA, which is based on the position of the train and updated continuously. The goal is to synthesise a strategy for the train to arrive to its destination as quickly as possible without exceeding its limits. We modelled the problem as a stochastic priced timed game. The controller is in charge of moving the train, playing against uncontrollable stochastic delays in communication. We used Uppaal Stratego to compute a strategy to enforce safety in the model. The safe strategy was statistically model checked to evaluate the mean arrival time of the train. This quantity was optimised, and the optimised strategy was compared to the safe one. We observed an improvement in the mean arrival time, whilst retaining safety. As far as we know, this is the first application of synthesis techniques to autonomous driving for next generation railway signalling systems.

This was our first experience with strategy synthesis and optimisation of a case study from the railway domain and also with Uppaal Stratego. Since this is a very recent tool there has not been much experimentation, in particular not outside of the groups involved in its development. The tool is still undergoing testing, and new versions and patches are released frequently. In fact, while

developing the model we ran into corner cases that needed interactions with the developers team at Aalborg University. Those interactions led to the release of new versions, with patches fixing the issues discovered through our model.

We did have experience in modelling and analysing railway case studies with Uppaal SMC [6,8,31]. The original model developed in [8] and statistically model checked had to be simplified considerably (cf. Sect. 4) to undergo strategy synthesis and verification. Indeed, while Uppaal SMC scales to large systems by applying simulations rather than full state-space exploration, Uppaal Stratego requires full state-space exploration of the timed game for strategy synthesis. For example, using the set-up discussed in Sect. 5 with `ma` = 10, if we double the constant `arrive` (i.e. 40 instead of 20) then during the strategy synthesis the tool terminates with an error message due to memory exhaustion.

An interesting future line of research would be to adapt the statistical synthesis techniques described in [16,35] to learn safety objectives, thus avoiding the full state-space exploration (as currently performed in Uppaal Stratego) while guaranteeing the scalability of SMC. This would enable the modelling of more complex ERTMS case studies. Also, further experiments, with different set-ups of the parameters and more trains and radio block centres need to be performed, to investigate the limits of the approach described in this paper in terms of optimisation. Finally, we intend to discuss with our railway project partners the impact of the techniques discussed in this paper.

Acknowledgements Funding by MIUR PRIN 2017FTXR7S project IT MaTTerS (Methods and Tools for Trustworthy Smart Systems) and H2020 project 4SECURail (FORmal Methods and CSIRT for the RAILway sector). The 4SECURail project received funding from the Shift2Rail Joint Undertaking under EU's H2020 research and innovation programme under grant agreement 881775.

We thank the Uppaal developers team, in particular Danny Poulsen, Marius Mikucionis, and Peter Jensen, for their assistance with Uppaal Stratego.

References

1. Agha, G., Palmskog, K.: A Survey of Statistical Model Checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018)
2. Arcaini, P., Ježek, P., Kofroň, J.: Modelling the Hybrid ERTMS/ETCS Level 3 Case Study in Spin. In: Butler et al. [18], pp. 277–291
3. B, T., Kalyanasundaram, S., Rao, M.V.P.: Coordinated Intelligent Traffic Lights using Uppaal Stratego. In: COMSNETS. pp. 789–794. IEEE (2019)
4. Bao, R., Attiogbé, J.C., Delahaye, B., Fournier, P., Lime, D.: Parametric Statistical Model Checking of UAV Flight Plan. In: FORTE. LNCS, vol. 11535, pp. 57–74 (2019)
5. Bartholomeus, M., Luttik, B., Willemse, T.: Modelling and Analysing ERTMS Hybrid Level 3 with the mCRL2 toolset. In: FMICS. LNCS, vol. 11119, pp. 98–114 (2018)
6. Basile, D., ter Beek, M.H., Ciancia, V.: Statistical Model Checking of a Moving Block Railway Signalling Scenario with Uppaal SMC. In: ISoLA. LNCS, vol. 11245, pp. 372–391 (2018)

7. Basile, D., ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F., Piattino, A., Trentini, D., Ferrari, A.: On the Industrial Uptake of Formal Methods in the Railway Domain. In: iFM. LNCS, vol. 11023, pp. 20–29 (2018)
8. Basile, D., ter Beek, M.H., Ferrari, A., Legay, A.: Modelling and Analysing ERTMS L3 Moving Block Railway Signalling with Simulink and UPPAAL SMC. In: FMICS. LNCS, vol. 11687 (2019)
9. Basu, A., Bensalem, S., Bozga, M., Caillaud, B., Delahaye, B., Legay, A.: Statistical Abstraction and Model-Checking of Large Heterogeneous Systems. In: FMOODS/FORTE. LNCS, vol. 6117, pp. 32–46 (2010)
10. ter Beek, M.H., Borälv, A., Fantechi, A., Ferrari, A., Gnesi, S., Löfving, C., Mazzanti, F.: Adopting Formal Methods in an Industrial Setting: The Railways Case. In: FM. LNCS, vol. 11800, pp. 762–772 (2019)
11. ter Beek, M.H., Gnesi, S., Knapp, A.: Formal methods for transport systems. Int. J. Softw. Tools Technol. Transf. **20**(3), 237–241 (2018)
12. Behrmann, G., David, A., Larsen, K.G., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: QEST. pp. 125–126. IEEE (2006)
13. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: UPPAAL-Tiga: Time for Playing Games! In: CAV. LNCS, vol. 4590, pp. 121–125 (2007)
14. Berger, U., James, P., Lawrence, A., Roggenbach, M., Seisenberger, M.: Verification of the European Rail Traffic Management System in Real-Time Maude. Sci. Comput. Program. **154**, 61–88 (2018)
15. Biagi, M., Carnevali, L., Paolieri, M., Vicario, E.: Performability evaluation of the ERTMS/ETCS – Level 3. Transp. Res. C-Emer. **82**, 314–336 (2017)
16. Bønneland, F., Jensen, P., Larsen, K.G., Muñiz, M., Srba, J.: Partial Order Reduction for Reachability Games. In: CONCUR. LIPIcs, vol. 140, pp. 23:1–23:15 (2019)
17. Boulanger, J.L. (ed.): Formal Methods Applied to Industrial Complex Systems — Implementation of the B Method. John Wiley & Sons (2014)
18. Butler, M., Raschke, A., Hoang, T., Reichl, K. (eds.): ABZ, LNCS, vol. 10817 (2018)
19. Cappart, Q., Limbrée, C., Schaus, P., Quilbeuf, J., Traonouez, L., Legay, A.: Verification of Interlocking Systems Using Statistical Model Checking. In: HASE. pp. 61–68. IEEE (2017)
20. Cassez, F., David, A., Fleury, E., Larsen, K.G., Lime, D.: Efficient On-the-Fly Algorithms for the Analysis of Timed Games. In: CONCUR. LNCS, vol. 3653, pp. 66–80 (2005)
21. Cunha, A., Macedo, N.: Validating the Hybrid ERTMS/ETCS Level 3 Concept with Electrum. In: Butler et al. [18], pp. 307–321
22. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015)
23. David, A., Jensen, P.G., Larsen, K.G., Legay, A., Lime, D., Sørensen, M.G., Taankvist, J.H.: On Time with Minimal Expected Cost! In: ATVA. LNCS, vol. 8837, pp. 129–145 (2014)
24. David, A., Jensen, P.G., Larsen, K.G., Mikucionis, M., Taankvist, J.H.: Uppaal Stratego. In: TACAS. LNCS, vol. 9035, pp. 206–211 (2015)
25. Dghaym, D., Poppleton, M., Snook, C.: Diagram-Led Formal Modelling Using iUML-B for Hybrid ERTMS Level 3. In: Butler et al. [18], pp. 338–352
26. ERTMS/ETCS RAMS Requirements Specification – chap. 2 – RAM (30/9/1998), http://www.era.europa.eu/Document-Register/Documents/B1-02s1266-.pdf
27. Fantechi, A.: Twenty-Five Years of Formal Methods and Railways: What Next? In: SEFM. LNCS, vol. 8368, pp. 167–183 (2013)

28. Fantechi, A., Ferrari, A., Gnesi, S.: Formal Methods and Safety Certification: Challenges in the Railways Domain. In: ISoLA. LNCS, vol. 9953, pp. 261–265 (2016)
29. Fantechi, A., Fokkink, W., Morzenti, A.: Some Trends in Formal Methods Applications to Railway Signaling. In: Formal Methods for Industrial Critical Systems: A Survey of Applications, chap. 4, pp. 61–84. John Wiley & Sons (2013)
30. Ferrari, A., Fantechi, A., Gnesi, S., Magnani, G.: Model-Based Development and Formal Methods in the Railway Industry. IEEE Softw. **30**(3), 28–34 (2013)
31. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing Formal Tools for System Design: a Judgment Study. In: ICSE. ACM (2020)
32. Furness, N., van Houten, H., Arenas, L., Bartholomeus, M.: ERTMS Level 3: the Game-Changer. IRSE News **232**, 2–9 (April 2017), https://www.irse.nl/resources/170314-ERTMS-L3-The-gamechanger-from-IRSE-News-Issue-232.pdf
33. Hansen, D., Leuschel, M., Körner, P., Krings, S., Naulin, T., Nayeri, N., Schneider, D., Skowron, F.: Validation and real-life demonstration of ETCS hybrid level 3 principles using a formal B model. Int. J. Softw. Tools Technol. Transf. (2020)
34. Haxthausen, A.E., Hede, K.: Formal Verification of Railway Timetables - Using the UPPAAL Model Checker. In: From Software Engineering to Formal Methods and Tools, and Back. LNCS, vol. 11865, pp. 433–448 (2019)
35. Jaeger, M., Jensen, P.G., Larsen, K.G., Legay, A., Sedwards, S., Taankvist, J.H.: Teaching Stratego to Play Ball: Optimal Synthesis for Continuous Space MDPs. In: ATVA. LNCS, vol. 11781, pp. 81–97 (2019)
36. Jansen, D.N., Hermanns, H.: Dependability Checking with StoCharts: Is Train Radio Reliable Enough for Trains? In: QEST. pp. 250–259. IEEE (2004)
37. Karra, S.L., Larsen, K.G., Lorber, F., Srba, J.: Safe and Time-Optimal Control for Railway Games. In: RSSRail. LNCS, vol. 11495, pp. 106–122 (2019)
38. Larsen, K.G., Mikucionis, M., Taankvist, J.H.: Safe and Optimal Adaptive Cruise Control. In: Correct System Design. LNCS, vol. 9360, pp. 260–277 (2015)
39. Legay, A., Delahaye, B., Bensalem, S.: Statistical Model Checking: An Overview. In: RV. LNCS, vol. 6418, pp. 122–135 (2010)
40. Mammar, A., Frappier, M., Tueno Fotso, S.J., Laleau, R.: An Event-B Model of the Hybrid ERTMS/ETCS Level 3 Standard. In: Butler et al. [18], pp. 353–366
41. Mazzanti, F., Ferrari, A.: Ten Diverse Formal Models for a CBTC Automatic Train Supervision System. In: MARS/VPT. EPTCS, vol. 268, pp. 104–149 (2018)
42. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: Towards formal methods diversity in railways: an experience report with seven frameworks. Int. J. Softw. Tools Technol. Transf. **20**(3), 263–288 (2018)
43. Nardone, R., Gentile, U., Benerecetti, M., Peron, A., Vittorini, V., Marrone, S., Mazzocca, N.: Modeling Railway Control Systems in Promela. In: FTSCS. CCIS, vol. 596, pp. 121–136 (2016)
44. Rispoli, F., Castorina, M., Neri, A., Filip, A., Di Mambro, G., Senesi, F.: Recent Progress in Application of GNSS and Advanced Communications for Railway Signaling. In: RADIOELEKTRONIKA. pp. 13–22. IEEE (2013)
45. UNISIG: FIS for the RBC/RBC handover (15/6/2016), http://www.era.europa.eu/Document-Register/Pages/set-2-FIS-for-the-RBC-RBC-handover.aspx