



**HAL**  
open science

## Tuning Trains Speed in Railway Scheduling

Étienne André

► **To cite this version:**

Étienne André. Tuning Trains Speed in Railway Scheduling. 25th International Conference on Formal Engineering Methods (ICFEM 2024), Kazuhiro Ogata; Meng Sun; Dominique Méry, Dec 2024, Hiroshima, Japan. pp.37-50, 10.1007/978-981-96-0617-7\_3. hal-04822116

**HAL Id: hal-04822116**

**<https://hal.science/hal-04822116v1>**

Submitted on 9 Dec 2024

**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.

Copyright

# Tuning Trains Speed in Railway Scheduling\*

Étienne André<sup>1,2</sup> 

<sup>1</sup> Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, Villetaneuse, France

<sup>2</sup> Institut Universitaire de France (IUF)

**Abstract.** Railway scheduling consists in ensuring that a set of trains evolve in a shared rail network without collisions, while meeting schedule constraints. This problem is notoriously difficult, even more in the case of uncertain or even unknown train speeds. We propose here a modeling and verification approach for railway scheduling in the presence of uncertain speeds, encoded here as uncertain segment durations. We formalize the system and propose a formal translation to PTAs. As a proof of concept, we apply our approach to benchmarks, for which we synthesize using IMITATOR suitable valuations for the segment durations.

**Keywords:** Railway scheduling · Timed automata · Parameter synthesis · IMITATOR.

## 1 Introduction

Railway scheduling consists in ensuring that a set of trains evolve in a shared rail network without collisions, while meeting local or global, absolute or relative timing constraints. This problem is notoriously difficult, and even more in the case of uncertain or even unknown train speeds, for which the solution needs to exhibit (or *synthesize*) speeds for which the schedule constraints are met without collisions. This becomes even more tricky when the schedule constraints (specifying, e.g., the time difference between two events in the network) become themselves uncertain or unknown.

*Contributions* In this paper, we offer a modeling and verification framework for railway scheduling in the presence of uncertain speeds, modeled using uncertain segment durations. Our railway model is close to that of [KR23] with some differences and simplifications: we consider a set of trains evolving in a shared network made of a double-vertex graph modeling segments and stations. Segments have a length and a maximum speed (which can be refined using the maximum speed of trains); such lengths and speeds are here encoded using traversal durations. Compared to [KR23], we notably extend the model with the

---

\* This is the author version of the manuscript of the same name published in the proceedings of the 25th International Conference on Formal Engineering Methods (ICFEM 2024). The final version is available at [10.1007/978-981-96-0617-7\\_3](https://doi.org/10.1007/978-981-96-0617-7_3). This work is partially supported by ANR BisoUS (ANR-22-CE48-0012).

ability to express uncertain or unknown speeds (and therefore durations). As target formalism for specification and verification, we choose parametric timed automata (PTAs) [AHV93], an extension of timed automata (TAs) [AD94] with unknown timing constants, allowing to model variability and uncertainty. Our contributions are three-fold:

1. a formal modeling of the train trajectory problem under uncertain speeds;
2. a translation scheme from our formal model into PTAs; and
3. a set of experiments to show the applicability of our approach.

*Outline* We review related works in Section 2. We recall necessary preliminaries in Section 3. We formalize our railway model (and the main problem) in Section 4. Our translation to PTAs is described in Section 5. As a proof of concept, we apply our translation to benchmarks in Section 6. We conclude in Section 7.

## 2 Related works

A number of works attempt to formalize railway scheduling problems using formal methods, with different model assumptions, and different target formalisms. In [Nar+14; Ben+17], the focus is on the formalization of railway control systems using extensions of hierarchical state machines called “Dynamic STate Machines” (DSTMs). In [Van10; Van18], colored Petri nets are used to model railway interlocking tables, with applications to Thai railway stations. Recent works such as [Lut+21; KR23] use SAT techniques, with [KR23] modeling continuous dynamics in a quite involved way.

Timed automata are a particularly well-suited formalism to model such problems, due to their ability to model concurrent and timed behaviors. Therefore, a number of works (such as [YWK13; Kha+16; Avr+19; Kar+19; Naz+19; LTH20]) are interested in scheduling or train interlocking problems. Timing uncertainty is not considered though.

In [Cha+18], so-called parametric timed automata (differing from usual PTAs [AHV93], as events can be parametrized too) are used to build monitors with variability in order to perform runtime verification of computer-based interlocking systems; an application to Beijing metro line 7 is briefly studied.

In contrast to these works, we address here uncertain or unknown segment traversal durations; we allow in addition for parametric schedule constraints.

Beyond the specific application to railways, planning and scheduling using TAs was considered in, e.g., [KMH01; AAM06; AM12]. Scheduling in the presence of uncertainty was addressed in some works using parametric timed automata, including scheduling problems with applications to the aerospace [Fri+12; And+21], or schedulability under uncertainty for uniprocessor environments [And17].

## 3 Preliminaries

We denote by  $\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}, \mathbb{R}_{\geq 0}$  the sets of non-negative integers, integers, non-negative rationals and non-negative reals, respectively. Let  $\bowtie \in \{<, \leq, =, \geq, >\}$ .

*Clocks* are real-valued variables that all evolve over time at the same rate. Throughout this paper, we assume a finite set  $\mathbb{X} = \{x_1, \dots, x_H\}$  of *clocks*. A *clock valuation* is a function  $\mu : \mathbb{X} \rightarrow \mathbb{R}_{\geq 0}$ , assigning a non-negative value to each clock. We write  $\mathbf{0}$  for the clock valuation assigning 0 to all clocks. Given a constant  $d \in \mathbb{R}_{\geq 0}$ ,  $\mu + d$  denotes the valuation s.t.  $(\mu + d)(x) = \mu(x) + d$ , for all  $x \in \mathbb{X}$ .

A (*timing*) *parameter* is an unknown rational-valued timing constant of a model. Throughout this paper, we assume a finite set  $\mathbb{P} = \{p_1, \dots, p_M\}$  of *parameters*. A *parameter valuation*  $v$  is a function  $v : \mathbb{P} \rightarrow \mathbb{Q}_{\geq 0}$ .

A *parametric clock constraint*  $C$  is a conjunction of inequalities over  $\mathbb{X} \cup \mathbb{P}$  of the form  $x \bowtie \sum_{1 \leq i \leq |\mathbb{P}|} \alpha_i p_i + d$ , with  $p_i \in \mathbb{P}$ , and  $\alpha_i, d \in \mathbb{Z}$ . Given  $C$ , we write  $\mu \models v(C)$  if the expression obtained by replacing each  $x$  with  $\mu(x)$  and each  $p$  with  $v(p)$  in  $C$  evaluates to true.

### 3.1 Parametric timed automata

Parametric timed automata (PTAs) extend TAs with a finite set of timing parameters allowing to model unknown constants.

**Definition 1 (Parametric timed automaton [AHV93]).** *A PTA  $\mathcal{A}$  is a tuple  $\mathcal{A} = (\Sigma, L, \ell_0, \ell_f, \mathbb{X}, \mathbb{P}, I, E)$ , where:*

1.  $\Sigma$  is a finite set of actions;
2.  $L$  is a finite set of locations;
3.  $\ell_0 \in L$  is the initial location;
4.  $\ell_f \in L$  is the final location;
5.  $\mathbb{X}$  is a finite set of clocks;
6.  $\mathbb{P}$  is a finite set of parameters;
7.  $I$  is the invariant, assigning to every  $\ell \in L$  a parametric clock constraint  $I(\ell)$  (called invariant);
8.  $E$  is a finite set of edges  $e = (\ell, g, a, R, \ell')$  where  $\ell, \ell' \in L$  are the source and target locations,  $a \in \Sigma$ ,  $R \subseteq \mathbb{X}$  is a set of clocks to be reset, and  $g$  is a parametric clock constraint (called guard).

As often, we assume PTAs extended with discrete global variables such as integer- or Boolean-valued variables. We also assume standard parallel composition of PTAs, synchronized on actions. The parallel composition of  $n$  PTAs is a PTA.

**Definition 2 (Valuation of a PTA).** *Given a parameter valuation  $v$ , we denote by  $v(\mathcal{A})$  the non-parametric structure where all occurrences of a parameter  $p_i$  have been replaced by  $v(p_i)$ .*

*Remark 1.* We have a direct correspondence between the valuation of a PTA and the definition of a TA. TAs were originally defined with integer constants in [AD94], while our definition of PTAs allows *rational*-valued constants. By assuming a rescaling of the constants (i.e., by multiplying all constants in a TA

by the least common multiple of their denominators), we obtain an equivalent (integer-valued) TA, as defined in [AD94]. So we assume in the following that  $v(\mathcal{A})$  is a TA.

**Definition 3 (Semantics of a TA).** *Given a PTA  $\mathcal{A} = (\Sigma, L, \ell_0, \ell_f, \mathbb{X}, \mathbb{P}, I, E)$  and a parameter valuation  $v$  the semantics of the TA  $v(\mathcal{A})$  is given by the TTS  $\mathfrak{T}_{v(\mathcal{A})} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \mathbb{R}_{\geq 0}, \rightarrow)$ , with*

1.  $\mathfrak{S} = \{(\ell, \mu) \in L \times \mathbb{R}_{\geq 0}^H \mid \mu \models I(\ell)v\}$ ,  $\mathfrak{s}_0 = (\ell_0, \mathbf{0})$ ,
2.  $\rightarrow$  consists of the discrete and (continuous) delay transition relations:
  - (a) discrete transitions:  $(\ell, \mu) \xrightarrow{e} (\ell', \mu')$ , if  $(\ell, \mu), (\ell', \mu') \in \mathfrak{S}$ , and there exists  $e = (\ell, g, a, R, \ell') \in E$ , such that  $\mu' = [\mu]_R$ , and  $\mu \models v(g)$ .
  - (b) delay transitions:  $(\ell, \mu) \xrightarrow{d} (\ell, \mu + d)$ , with  $d \in \mathbb{R}_{\geq 0}$ , if  $\forall d' \in [0, d], (\ell, \mu + d') \in \mathfrak{S}$ .

Moreover we write  $(\ell, \mu) \xrightarrow{(d,e)} (\ell', \mu')$  for a combination of a delay and discrete transition if  $\exists \mu'' : (\ell, \mu) \xrightarrow{d} (\ell, \mu'') \xrightarrow{e} (\ell', \mu')$ .

Given a TA  $\mathcal{A}$  with concrete semantics  $\mathfrak{T}_{\mathcal{A}}$ , we refer to the states of  $\mathfrak{S}$  as the *concrete states* of  $\mathcal{A}$ . A *run* of  $\mathcal{A}$  is an alternating sequence of concrete states of  $\mathcal{A}$  and pairs of edges and delays starting from the initial state  $\mathfrak{s}_0$  of the form  $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots$  with  $i = 0, 1, \dots$ ,  $e_i \in E$ ,  $d_i \in \mathbb{R}_{\geq 0}$  and  $(\ell_i, \mu_i) \xrightarrow{(d_i, e_i)} (\ell_{i+1}, \mu_{i+1})$ .

## 4 Railway system model

We formalize here our railway system model. Our railway model is inspired by that of [KR23], with some differences that will be highlighted. A key difference is the ability of our model to define *parametric* durations. We also propose a more formal definition of the system.

### 4.1 Rail network graph

The infrastructure is modeled using a double-vertex graph [Mon92], with nodes and segments. Nodes can be normal nodes (not allowing stopping) or stations (where trains may choose to stop or not). Segments have a length and a speed limit, encoded here using a segment traversal time (which can be exceeded for slower trains that have a speed limit smaller than the segment maximal speed). Boundary nodes are start or end nodes for the trains. As in [KR23], we do not model slope, angle or tunnels. However, cycles can be encoded, as opposed to [Lut+21] where this is not immediate.

We assume that segments are bidirectional, that at most one train is allowed in a segment, and that each segment is longer than any train; as a consequence, a train can occupy at most two segments at once. As in [KR23], “to support modeling of railway junctions, nodes of the graph have two sides (illustrated by black and blue or red colors in Fig. 1). In order to avoid physically impossible

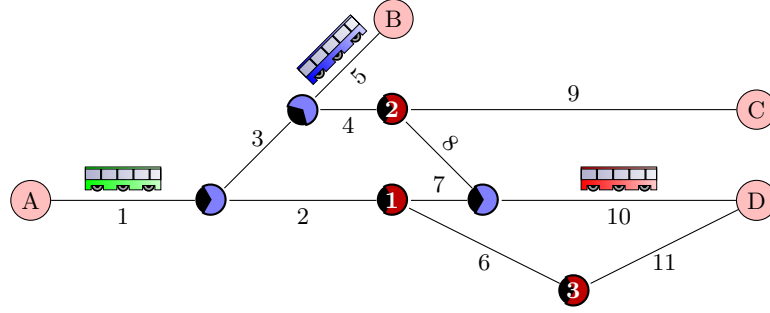


Fig. 1: An example of a rail network graph with 3 trains [KR23]

(e.g., too sharp) turns, a train has to visit both sides when transferring via such a double-sided node.” Different from [KR23], we model segment length and speed using a *traversal time*; similarly, since trains can occupy two segments at the same time, we model the time needed to completely move from one segment to the next one using another traversal time. These times are minimum, as slower trains can potentially define longer times for each segment and pairs of segments (see Definition 5). Formally:

**Definition 4 (rail network graph).** A rail network graph is a tuple  $\mathcal{G} = (N, B, St, Seg, SegDur, SegsDur, T)$  where

- $N$  is the set of nodes,
- $B \subseteq N$  is the set of boundary nodes,
- $St \subseteq N$  is the set of stations,
- $Seg$  is the set of segments,
- $SegDur : Seg \rightarrow \mathbb{Q}_{\geq 0} \cup \mathbb{P}$  assigns a (possibly parametric) duration to each segment,
- $SegsDur : (Seg \times Seg) \rightarrow \mathbb{Q}_{\geq 0} \cup \mathbb{P}$  assigns a (possibly parametric) duration to each pair of consecutive segments, and
- $T \in 2^{Seg} \times N \times 2^{Seg}$  is the set of transitions.

Transitions encode the way trains can move via nodes. For example, given a transition  $(l, n, r) \in T$ , a train can move from any segment  $seg \in l$  to any segment  $seg' \in r$  via  $n$  (or the opposite way).

*Example 1.* Consider the rail network graph in Fig. 1. The graph contains 4 boundary nodes (A, B, C, D) and 3 stations (depicted in red, and labeled with a number). Other nodes are normal nodes. Segments are labeled with a number identifying them. We can assume for example that the minimum time to traverse segment 1 is set to 8, the time to move from segment 1 to 2 is 2, while the time to move from segment 1 to 3 is 1 (values not depicted in Fig. 1).

## 4.2 Trains

A train is characterized by its velocity limit and its connection. Different from [KR23], weight, acceleration and deceleration are not encoded; we assume they can be incorporated in segment durations. Trains always drive at their maximum possible speed; they can however stop arbitrarily long in stations.

**Connection constraints** As in [KR23], a *connection* is a mapping of a train to a non-empty list of nodes that must be visited in the given order. Only the first and the last element can be boundary nodes. The list of nodes must start with the boundary node denoting the starting point of the train. The list may then contain nodes that must be visited; if a node is a station, then the train can stop at this station. Trains can only stop at stations part of the connection. If the last node of the list is a boundary node, then the train must end in this node. If the last node of the connection is not a boundary node, then the train can end in any boundary node.

Each train has exactly one connection.

*Example 2 (train connections [KR23]).* Consider the green train from Fig. 1. Assume its connection is  $[A, 3]$ . This connection denotes that the green train must start at node A, must stop at train station 3, but cannot stop at train station 1 because it is not part of the connection. The train can end in any boundary node (even though, considering the graph topology, only D can be an end node considering the connection).

Consider the red train from Fig. 1. Assume its connection is  $[D, A]$ . This connection denotes that the train must depart from D, and reach A without stopping at any intermediate station; note that there are three paths allowing this connection.

**Definition 5 (train).** *Given a rail network graph  $\mathcal{G} = (N, B, St, Seg, SegDur, SegsDur, T)$ , a train over  $\mathcal{G}$  is a triple  $t = (TSegDur, TSegsDur, C)$  where*

- $TSegDur : Seg \rightarrow \mathbb{Q}_{\geq 0} \cup \mathbb{P}$  assigns a possibly parametric duration to each segment,
- $TSegsDur : (Seg \times Seg) \rightarrow \mathbb{Q}_{\geq 0} \cup \mathbb{P}$  assigns a possibly parametric duration to each pair of consecutive segments, and
- $C \in N^*$  is the train connection.

Given a segment, a train drives at its maximum speed depending on the network conditions: that is, the segment duration for this train is the *maximum* between the segment duration specified by the network ( $SegDur$ ) and the segment duration specified by the train ( $TSegDur$ )—and similarly for pairs of consecutive segments.

### 4.3 Schedule constraints

We formalize the schedule constraints from [KR23], allowing to compare the time when a train arrives or departs from a node:  $arrival(t, n)$  (resp.  $departure(t, n)$ ) denotes the time when train  $t$  arrives at (resp. leaves) node  $n$ . We use as generic notation  $visit(t, n)$  to denote arrival or departure. We define three forms of constraints, detailed in the following. An originality of our approach is that we also allow for *parametric* constraints.

**Ordering constraints** Ordering constraints constrain the order in which visits should be made. They are of the form

$$visit_1(t_1, n_1) \bowtie visit_2(t_2, n_2).$$

**Absolute timing constraints** Absolute timing constraints constrain the visit of a node at an absolute time. They are of the form

$$visit(t, n) \bowtie d, \text{ with } d \in \mathbb{Q}_{\geq 0} \cup \mathbb{P}.$$

**Relative timing constraints** Relative timing constraints constrain the time between two visits. Let  $transfer(visit_1(t_1, n_1), visit_2(t_2, n_2)) := visit_2(t_2, n_2) - visit_1(t_1, n_1)$ . Then relative timing constraints are of the form

$$transfer(visit_1(t_1, n_1), visit_2(t_2, n_2)) \bowtie d, \text{ with } d \in \mathbb{Q}_{\geq 0} \cup \mathbb{P}.$$

Finally, let us define  $wait(t, n) := transfer(arrival(t, n), departure(t, n))$ .

*Example 3 (schedule constraints).* We formalize in the following some of the informal examples from [KR23]. The fact that the blue train must start before the green train can be encoded using  $departure(t_{blue}, A) \leq departure(t_{green}, A)$ . The fact that the red train starts before the green train approaches node 1 can be encoded using  $departure(t_{red}, D) \leq arrival(t_{green}, 1)$ . The fact that the red train must reach A within 10 time units after entering the network can be encoded using  $transfer(departure(t_{red}, D), arrival(t_{red}, A)) \leq 10$ . The fact that the green train must wait at node 3 for at least  $p$  time units can be encoded using  $wait(t_{green}, 3) \geq p$ .

### 4.4 Constrained railway system

**Definition 6 (constrained railway system).** A constrained railway system is a tuple  $S = (\mathcal{G}, \mathcal{T}, SC)$  where

- $\mathcal{G}$  is a rail network graph,
- $\mathcal{T}$  is a set of trains over  $\mathcal{G}$ , and
- $SC$  is a set of schedule constraints.



## 4.5 Objective

### Train trajectory problem under uncertain speeds:

INPUT: a constrained railway system

PROBLEM: Synthesize segment durations and schedule constraints parameters such that all train connections and schedule constraints are met.

## 5 Translation to parametric timed automata

### 5.1 Overview of the translation

Our translation is modular, in the sense that each train and each schedule constraint is translated into a different PTA. The system is made of the parallel composition of these PTAs, synchronized using the actions modeling the arrival of a train into a node, and the departure of a train from a node.

### 5.2 Railway model and trains

Due to the *concurrent* and *real-time* nature of the system, a simple discrete graph with, e.g., a list of trains currently at each node, is not a suitable approach. Instead, we choose a fully distributed approach, where each train evolves in its own representation of the rail network graph, in a continuous manner. That is, we define for each train  $k$  a PTA (with a single clock  $x^k$ ), with the set of locations being made of the segments and the node of the rail network graph.

The mutual exclusion in segments and nodes is ensured using global Boolean variables, carefully tested and updated when attempting to enter, and when exiting a segment or node. More precisely, the occupancy of each segment  $seg_i$  is encoded by a Boolean variable  $segfree_i$  (denoting that the segment is free).

We give in Fig. 2 the encoding of two consecutive segments  $seg_i$  and  $seg_j$  via a node  $n_{ij}$  for a given train  $t_k = (TSegDur, TSegsDur, C)$ . As expected, the train can remain in a segment exactly  $\max(SegDur(seg_i), TSegDur(seg_i))$  time units, and similarly in a location modeling the move of a segment to the next one (here location “ $n_{ij}$ ”). A train can move to the node between two segments only if the next segment ( $seg_j$ ) is free (“ $segfree_j = \top$ ”), and the segment then becomes occupied (“ $segfree_j \leftarrow \perp$ ”). The actions “ $arrival_{ij}^k$ ” and “ $departure_{ij}^k$ ” are used to (potentially) synchronize with PTAs modeling schedule constraints.

If the node between  $seg_i$  and  $seg_j$  is a station in which the train may stop (because it is part of its connection), then it is possible to stay longer in this node: in that case, the “=” sign in the guard between locations “ $n_{ij}$ ” and “ $seg_j$ ” in Fig. 2 becomes “ $\geq$ ”, and the invariant of location “ $n_{ij}$ ” is removed.

*Connections* A connection is easily encoded using a discrete integer-valued variable: a node occurring at the  $n$ th position of the connection can only be visited if  $n - 1$  nodes were visited by this train in the past, which is easily modeled by incrementing a local discrete integer. The initial location of the train PTA is the node in which the train starts, and the final location is the node the train is supposed to reach at the end of its connection.

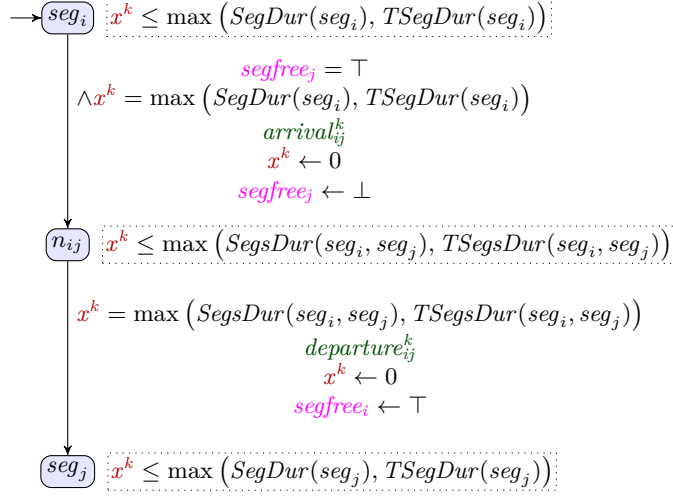
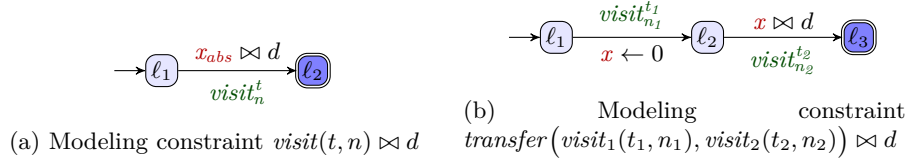

 Fig. 2: Modeling two consecutive segments  $\text{seg}_i$  and  $\text{seg}_j$  via node  $n_{ij}$  for train  $t_k$ 


Fig. 3: Modeling schedule constraints

### 5.3 Schedule constraints

Ordering constraints are modeled exactly like connections, using discrete variables making sure the visits are performed in the specified order.

Each absolute timing constraint is modeled using a dedicated PTA, using a (single, global) clock  $x_{abs}$  measuring the absolute time, i.e., never reset throughout the PTAs. We give in Fig. 3a the PTA modeling constraint  $\text{visit}(t, n) \bowtie d$ . The PTA simply constrains action  $\text{visit}_n^t$  to occur only when guard “ $x_{abs} \bowtie d$ ” is satisfied (recall that “ $\text{visit}$ ” stands for either “ $\text{arrival}$ ” or “ $\text{departure}$ ”).

Each relative timing constraint is modeled using a dedicated PTA, using a local clock  $x$  measuring the relative time between different events. We give in Fig. 3b the PTA modeling the relative timing constraint  $\text{transfer}(\text{visit}_1(t_1, n_1), \text{visit}_2(t_2, n_2)) \bowtie d$ . This PTA constrains the time difference between  $\text{visit}_{n_1}^{t_1}$  and  $\text{visit}_{n_2}^{t_2}$  to be as specified by the constraint, using guard “ $x \bowtie d$ ”.

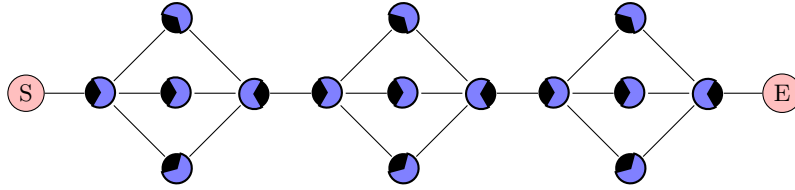


Fig. 4: An example of a serial-parallel infrastructure, with  $N_S = N_P = 3$  [KR23]

#### 5.4 Solving the train trajectory problem

Given  $\mathcal{A}$  the PTA resulting of the parallel composition of the aforementioned PTAs, the trajectory problem without timing parameters is satisfied if the final location of all PTAs is reachable. That is, each train reached its final destination, while all schedule constraints are satisfied.

In the presence of timing parameters, the set of parametric durations (modeling uncertain segment durations) for which all train connections and schedule constraints are met corresponds to the set of parameter valuations for which the final location of all PTAs is reachable. This can be solved using reachability-synthesis, i.e., the synthesis of timing parameters for which some PTA location is reachable [AHV93]. While reachability-emptiness is in general undecidable [AHV93; And19], reachability-synthesis can be effectively solved under the assumption that there is no loop in the rail network graph.

## 6 Experiments

As a proof of concept, we verify two constrained railway systems using the IMITATOR parametric timed model checker [And21]. IMITATOR takes as input networks of parametric timed automata extended with a number of features, such as synchronization and discrete variables (used here). Experiments were conducted on a Dell Precision 5570 with an Intel® Core™ i7-12700H with 32 GiB memory running Linux Mint 21 Vanessa. We used IMITATOR 3.4-alpha “Cheese Durian”, build `feat/forall_actions/788f551`. Models (including durations for Fig. 1) and results can be found at <https://www.imitator.fr/static/ICFEM24> and <https://doi.org/10.5281/zenodo.13789618>.

*Running example* We first consider the running example in Fig. 1, without timing parameters. IMITATOR derives in 1.24s that the train trajectory problem is satisfied, i.e., there exists a schedule such that all trains meet their connections and schedule constraints. Second, we add a parametric schedule constraint  $visit(t_{red}, A) \leq p_R$ , where  $p_R \in \mathbb{P}$ . That is,  $p_R$  denotes the upper bound such that the red train reaches its destination. IMITATOR derives in 1.91s the set of parameter valuations  $p_R \geq 36$ , i.e., the red train cannot be faster than 36 time units. Third, we parametrize the minimum duration for segments 2 and 7, i.e.,  $SegDur(seg_2) = p_2$  and  $SegDur(seg_7) = p_7$ ,

with  $p_2, p_7 \in \mathbb{P}$ . IMITATOR derives in 9.83s the set of parameter valuations  $(p_R \geq p_2 + 28) \vee (p_R \geq p_2 + p_7 + 20) \vee (p_R \geq 45)$ . This gives the correct (sound and complete) condition over the segment durations and schedule constraints parameters such that all train connections and schedule constraints are met. The fact that IMITATOR derives symbolic (continuous) sets of timing parameters is a major advantage over, e.g., SMT solvers, that would typically derive (non-necessarily complete) discrete sets of specific valuations. The symbolic and dense nature of our results comes from the underlying symbolic techniques for parameter synthesis using PTAs.

*Scalability* To evaluate the scalability of our approach, we consider a serial-parallel infrastructure, i.e., a network with  $N_S$  serially connected groups of  $N_P$  identical parallel tracks with a station. Connections only include  $[S, E]$ , i.e., trains cannot stop at any station, and are free to use any path. This variety of choices for each train obviously leads to an exponential blowup. We consider various values for the number  $N_T$  of trains in the infrastructure, and for the number of groups  $N_S$  and parallel tracks  $N_P$ ; as in [KR23], we fix  $N_S = N_P$ . An example with  $N_S = N_P = 3$  is given in Fig. 4:  $N_S = 3$  denotes the three groups from  $S$  to  $E$  (from left to right), while  $N_P = 3$  denotes the three options (drawn vertically) to traverse one group.

We reuse two scenarios from [KR23]: the “nop” scenario does not contain any parametric duration, nor any schedule constraint; however, contrarily to [KR23], we add one parametric absolute timing constraint:  $arrival(t_{N_T}, E) \leq J$ , with  $J \in \mathbb{P}$  (in [KR23],  $J$  is a constant manually tuned). That is, we measure the end-to-end time from the first train leaving  $S$  to the last train reaching  $E$ .

The “last” scenario in [KR23] additionally ensures that the last train takes less than  $bnd$  time units between its departure and arrival. Again, we parametrize this value instead of manually enumerating it, by adding the following relative timing constraint:  $transfer(departure(t_{N_T}, S), arrival(t_{N_T}, E)) \leq bnd$ , with  $bnd \in \mathbb{P}$ .

Note that a direct comparison with the experiments of [KR23] would probably not make sense since *i*) the model is not the same (on the one hand, a more involved dynamics is considered in [KR23] and, on the other hand, we allow for more flexible durations and schedule constraints), *ii*) the segment durations are not given in [KR23] and, most importantly, *iii*) we *synthesize* correct valuations while [KR23] only *verifies* the system for constant values.

We give in Table 1a the results for the “nop” scenario: we give from left to right the numbers of groups (and parallel tracks), of trains, of PTAs in the translated model, of clocks, of parameters, of generated states during the analysis; we finally give the synthesized value for  $J$  and the computation time. “**T.O.**” denotes timeout after 1 800s. Similarly, we give in Table 1b the results for the “last” scenario with, as additional column, the synthesized bound “ $bnd$ ” for the relative timing constraint.

While the computation time is clearly exponential, which is not a surprise considering the way we designed this scalability test, a positive outcome is that we get interesting results for up to 4 trains or up to 4 groups of 4 parallel tracks,

Table 1: Experiments

(a) Scenario “nop”								(b) Scenario “last”								
$N_S$	$N_T$	$ \mathcal{A} $	$ \mathbb{X} $	$ \mathbb{P} $	$ S $	$J$	$t(s)$	$N_S$	$N_T$	$ \mathcal{A} $	$ \mathbb{X} $	$ \mathbb{P} $	$ S $	$J$	$bnd$	$t(s)$
2	1	2	2	1	25	63	0.01	2	1	2	3	2	25	63	63	0.01
2	2	3	3	1	238	75	0.08	2	2	3	4	2	238	75	63	0.12
2	3	4	4	1	2323	87	2.68	2	3	4	5	2	2323	87	63	3.40
2	4	5	5	1	22450	99	96.02	2	4	5	6	2	22450	99	63	113.41
3	1	2	2	1	51	92	0.01	3	1	2	3	2	51	92	92	0.02
3	2	3	3	1	1237	104	0.68	3	2	3	4	2	1052	104	92	0.91
3	3	4	4	1	27385	116	137.61	3	3	4	5	2	27385	116	92	157.78
3	4	-	-	-	-	-	<b>T.O.</b>	3	4	-	-	-	-	-	-	<b>T.O.</b>
4	1	2	2	1	87	121	0.03	4	1	2	3	2	87	121	121	0.04
4	2	3	3	1	3195	133	3.28	4	2	3	4	2	3195	133	121	4.64
4	3	-	-	-	-	-	<b>T.O.</b>	4	3	-	-	-	-	-	-	<b>T.O.</b>

a rather elaborate situation—especially in a parametric setting with unknown timing bounds in the schedule constraints.

A difference with [KR23; Lut+21] is that we can automatically synthesize the bound between the first train departure and the last train arrival, while they are manually iterated in [KR23; Lut+21]. The second parameter (“ $bnd$ ”) is simply tested in [KR23] against 3 values ( $10$ ,  $10^2$ ,  $10^3$ ) without attempting to synthesize a tight valuation.

## 7 Conclusion and perspectives

We presented a formal model for verifying constrained railway systems in the presence of unknown durations, not only to model segment traversal times, but also to be used in relative and absolute schedule constraints. Our translation to PTAs allowed us to verify benchmarks using IMITATOR, and to derive internal segment durations and optimal values for schedule constraints.

We believe our framework, although simple, can serve as a preliminary basis for more involved settings. Notably, modeling acceleration and deceleration would be an interesting enhancement, possibly using piecewise discretization to keep the linear nature of our framework. Taking energy consumption into consideration would be another interesting future work, e.g., with an optimality criterion, perhaps integrating our setting with other approaches such as [Bac+21; LRS21]. In addition, tackling the exponential blowup could be partially achieved using partial order or symmetry reductions, since these models are heavily symmetric. Finally, we used here an *ad hoc* modeling language; integrating this framework into standard domain-specific languages will be an interesting extension.

*Acknowledgments* I would like to thank Tomáš Kolárik and Stefan Ratschan for introducing me to their work, and an anonymous reviewer for helpful comments.

The colored trains drawn using  $\text{\LaTeX}$  TikZ in Fig. 1 are designed by `cfr` from `stackexchange`.

## References

- [AAM06] Yasmina Abdeddaïm, Eugene Asarin, and Oded Maler. “Scheduling with timed automata”. In: *TCS* 354.2 (Mar. 2006), pp. 272–300. DOI: [10.1016/j.tcs.2005.11.018](https://doi.org/10.1016/j.tcs.2005.11.018) (cit. on p. 2).
- [AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: *TCS* 126.2 (Apr. 1994), pp. 183–235. DOI: [10.1016/0304-3975\(94\)90010-8](https://doi.org/10.1016/0304-3975(94)90010-8) (cit. on pp. 2–4).
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, USA: ACM, 1993, pp. 592–601. DOI: [10.1145/167088.167242](https://doi.org/10.1145/167088.167242) (cit. on pp. 2, 3, 10).
- [AM12] Yasmina Abdeddaïm and Damien Masson. “Real-Time Scheduling of Energy Harvesting Embedded Systems with Timed Automata”. In: *RTCSA* (Aug. 19–22, 2012). Seoul, South Korea: IEEE Computer Society, 2012, pp. 31–40. DOI: [10.1109/RTCSA.2012.21](https://doi.org/10.1109/RTCSA.2012.21) (cit. on p. 2).
- [And+21] Étienne André, Emmanuel Coquard, Laurent Fribourg, Jawher Jerray, and David Lesens. “Parametric schedulability analysis of a launcher flight control system under reactivity constraints”. In: *FI* 182.1 (Sept. 2021), pp. 31–67. DOI: [10.3233/FI-2021-2065](https://doi.org/10.3233/FI-2021-2065) (cit. on p. 2).
- [And17] Étienne André. “A unified formalism for monoprocessor schedulability analysis under uncertainty”. In: *FMICS-AVoCS* (Sept. 18–20, 2017). Ed. by Ana Cavalcanti, Laure Petrucci, and Cristina Seceleanu. Vol. 10471. LNCS. Torino, Italy: Springer, 2017, pp. 100–115. DOI: [10.1007/978-3-319-67113-0\\_7](https://doi.org/10.1007/978-3-319-67113-0_7) (cit. on p. 2).
- [And19] Étienne André. “What’s decidable about parametric timed automata?” In: *STTT* 21.2 (Apr. 2019), pp. 203–219. DOI: [10.1007/s10009-017-0467-0](https://doi.org/10.1007/s10009-017-0467-0) (cit. on p. 10).
- [And21] Étienne André. “IMITATOR 3: Synthesis of timing parameters beyond decidability”. In: *CAV* (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. LNCS. virtual: Springer, 2021, pp. 1–14. DOI: [10.1007/978-3-030-81685-8\\_26](https://doi.org/10.1007/978-3-030-81685-8_26) (cit. on p. 10).
- [Avr+19] Camelia Avram, Karolina Bezerra, Dan Radu, Jose Machado, and Adina Astilean. “A Formal Approach for Railroad Traffic Modelling Using Timed Automata”. In: *Innovation, Engineering and Entrepreneurship*. Ed. by José Machado, Filomena Soares, and Germano Veiga. Springer International Publishing, 2019, pp. 307–314. ISBN: 978-3-319-91334-6. DOI: [10.1007/978-3-319-91334-6\\_42](https://doi.org/10.1007/978-3-319-91334-6_42) (cit. on p. 2).
- [Bac+21] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. “Optimal and robust controller synthesis using energy timed automata with uncertainty”. In: *FAC* 33.1 (2021), pp. 3–25. DOI: [10.1007/s00165-020-00521-4](https://doi.org/10.1007/s00165-020-00521-4) (cit. on p. 12).

- [Ben+17] Massimo Benerecetti, Renato De Guglielmo, Ugo Gentile, Stefano Marone, Nicola Mazzocca, Roberto Nardone, Adriano Peron, Luigi Velardi, and Valeria Vittorini. “Dynamic state machines for modelling railway control systems”. In: *SCP* 133 (2017), pp. 116–153. DOI: [10.1016/J.SCICO.2016.09.002](https://doi.org/10.1016/J.SCICO.2016.09.002) (cit. on p. 2).
- [Cha+18] Ming Chai, Haifeng Wang, Jian Zhang, and Tao Tang. “Runtime Verification of Railway Interlocking Software with Parametric Timed Automata”. In: *ICIRT* (Dec. 12–14, 2018). Singapore, 2018, pp. 1–5. DOI: [10.1109/ICIRT.2018.8641559](https://doi.org/10.1109/ICIRT.2018.8641559) (cit. on p. 2).
- [Fri+12] Laurent Fribourg, David Lesens, Pierre Moro, and Romain Soulat. “Robustness Analysis for Scheduling Problems using the Inverse Method”. In: *TIME* (Sept. 12–14, 2012). Ed. by Mark Reynolds, Paolo Terenziani, and Ben Moszkowski. Leicester, UK: IEEE Computer Society Press, Sept. 2012, pp. 73–80. DOI: [10.1109/TIME.2012.10](https://doi.org/10.1109/TIME.2012.10) (cit. on p. 2).
- [Kar+19] Shyam Lal Karra, Kim Guldstrand Larsen, Florian Lorber, and Jiří Srba. “Safe and Time-Optimal Control for Railway Games”. In: *RSS-Rail* (June 4–6, 2019). Ed. by Simon Collart Dutilleul, Thierry Lecomte, and Alexander B. Romanovsky. Vol. 11495. LNCS. Lille, France: Springer, 2019, pp. 106–122. DOI: [10.1007/978-3-030-18744-6\\_7](https://doi.org/10.1007/978-3-030-18744-6_7) (cit. on p. 2).
- [Kha+16] Umar Khan, Jamil Ahmad, Tariq Saeed, and Sikandar Hayat Mirza. “On the real time modeling of interlocking system of passenger lines of Rawalpindi Cantt train station”. In: *Complex Adaptive Systems Modeling* 4 (2016), p. 17. DOI: [10.1186/S40294-016-0028-5](https://doi.org/10.1186/S40294-016-0028-5) (cit. on p. 2).
- [KMH01] Lina Khatib, Nicola Muscettola, and Klaus Havelund. “Mapping Temporal Planning Constraints into Timed Automata”. In: *TIME*. Cividale del Friuli, Italy: IEEE Computer Society, 2001, pp. 21–27. ISBN: 0-7695-1107-4. DOI: [10.1109/TIME.2001.930693](https://doi.org/10.1109/TIME.2001.930693) (cit. on p. 2).
- [KR23] Tomáš Kolárik and Stefan Ratschan. “Railway Scheduling Using Boolean Satisfiability Modulo Simulations”. In: *FM* (Mar. 6–10, 2023). Ed. by Marsha Chechik, Joost-Pieter Katoen, and Martin Leucker. Vol. 14000. LNCS. Lübeck, Germany: Springer, 2023, pp. 56–73. DOI: [10.1007/978-3-031-27481-7\\_5](https://doi.org/10.1007/978-3-031-27481-7_5) (cit. on pp. 1, 2, 4–7, 10–12).
- [LRS21] Didier Lime, Olivier H. Roux, and Charlotte Seidner. “Cost Problems for Parametric Time Petri Nets”. In: *FI* 183.1-2 (2021), pp. 97–123. DOI: [10.3233/FI-2021-2083](https://doi.org/10.3233/FI-2021-2083) (cit. on p. 12).
- [LTH20] Per Lange Laursen, Van Anh Thi Trinh, and Anne E. Haxthausen. “Formal Modelling and Verification of a Distributed Railway Interlocking System Using UPPAAL”. In: *ISoLA, Part III* (Oct. 20–30, 2020). Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 12478. LNCS. Rhodes, Greece: Springer, 2020, pp. 415–433. DOI: [10.1007/978-3-030-61467-6\\_27](https://doi.org/10.1007/978-3-030-61467-6_27) (cit. on p. 2).
- [Lut+21] Bjørnar Luteberget, Koen Claessen, Christian Johansen, and Martin Steffen. “SAT modulo discrete event simulation applied to railway design capacity analysis”. In: *FMSD* 57.2 (2021), pp. 211–245. DOI: [10.1007/S10703-021-00368-2](https://doi.org/10.1007/S10703-021-00368-2) (cit. on pp. 2, 4, 12).
- [Mon92] Markus Montigel. “Formal representation of track topologies by double vertex graphs”. In: *Railcomp*. Vol. 2. Computers in Railways 3. Washington DC, USA: Computational Mechanics Publications, 1992, pp. 359–370 (cit. on p. 4).

- [Nar+14] Roberto Nardone, Ugo Gentile, Adriano Peron, Massimo Benerecetti, Valeria Vittorini, Stefano Marrone, Renato De Guglielmo, Nicola Mazzocca, and Luigi Velardi. “Dynamic State Machines for Formalizing Railway Control System Specifications”. In: *FTSCS* (Nov. 6–7, 2014). Ed. by Cyrille Artho and Peter Csaba Ölveczky. Vol. 476. CCIS. Luxembourg: Springer, 2014, pp. 93–109. DOI: 10.1007/978-3-319-17581-2\_7 (cit. on p. 2).
- [Naz+19] Yul Y. Nazaruddin, Tua A. Tamba, K. Pradityo, B. Aristyo, and A. Widoyotriatmo. “Safety Verification of a Train Interlocking Timed Automaton Model”. In: *IFAC-PapersOnLine* 52.15 (2019). 8th IFAC Symposium on Mechatronic Systems MECHATRONICS 2019, pp. 331–335. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.11.696 (cit. on p. 2).
- [Van10] Somsak Vanit-Anunchai. “Modelling Railway Interlocking Tables Using Coloured Petri Nets”. In: *COORDINATION* (June 7–9, 2010). Ed. by Dave Clarke and Gul A. Agha. Vol. 6116. LNCS. Amsterdam, The Netherlands: Springer, 2010, pp. 137–151. DOI: 10.1007/978-3-642-13414-2\_10 (cit. on p. 2).
- [Van18] Somsak Vanit-Anunchai. “Modelling and simulating a Thai railway signalling system using Coloured Petri Nets”. In: *STTT* 20.3 (2018), pp. 243–262. DOI: 10.1007/S10009-018-0482-9 (cit. on p. 2).
- [YWK13] Lei Yuan, Junfeng Wang, and Renwei Kang. “The Verification of Temporary Speed Restriction of Train Control System Based on Timed Automata”. In: *ICCNCE*. Atlantis Press, July 2013, pp. 355–358. ISBN: 978-90-78677-67-3. DOI: 10.2991/iccnce.2013.89 (cit. on p. 2).