Probabilistic Timed Graph Transformation Systems

Maria Maximova, Holger Giese, Christian Krause

Technische Berichte Nr. 118

des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 118

Maria Maximova | Holger Giese | Christian Krause

Probabilistic Timed Graph Transformation Systems

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2017
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URN urn:nbn:de:kobv:517-opus4-397055
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-397055

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-405-0

Today, software has become an intrinsic part of complex distributed embedded real-time systems. The next generation of embedded real-time systems will interconnect the today unconnected systems via complex software parts and the service-oriented paradigm. Therefore besides timed behavior and probabilistic behavior also structure dynamics, where the architecture can be subject to changes at run-time, e.g. when dynamic binding of service end-points is employed or complex collaborations are established dynamically, is required. However, a modeling and analysis approach that combines all these necessary aspects does not exist so far.

To fill the identified gap, we propose Probabilistic Timed Graph Transformation Systems (PTGTSs) as a high-level description language that supports all the necessary aspects of structure dynamics, timed behavior, and probabilistic behavior. We introduce the formal model of PTGTSs in this paper and present a mapping of models with finite state spaces to probabilistic timed automata (PTA) that allows to use the PRISM model checker to analyze PTGTS models with respect to PTCTL properties.

Contents

1 Introduction

Today, software has become an intrinsic part of complex distributed embedded real-time systems, which need to realize more advanced functionality. The next generation of embedded real-time systems will interconnect the today unconnected systems via complex software parts and the service-oriented paradigm. It is envisioned that such networked systems will be able to behave much more intelligently by building communities of autonomous agents that exploit local and global networking to adapt and optimize their functionality [6].

In contrast to today's real-time systems, their behavior will in addition be characterized by *structure dynamics* that results from their complex coordination behavior. This structure dynamics requires execution in real-time and reconfiguration at run-time to adjust the systems behavior to its changing context and goals, leading to self-adaptation and self-optimization [24]. For these systems, also the structure resp. architecture is subject to changes at run-time, e.g. when dynamic binding of service end-points is employed or complex collaborations are established dynamically. In the latter case, often the structural context in the form of local topology and distribution information is particularly important.

As a concrete example for such an advanced embedded real-time system, the RailCab research project [23] aims at combining a passive track system with intelligent shuttles that operate autonomously, act individually, and make independent and decentralized operational decisions. For the RailCab application example it holds that some functionality may be safety-critical such as the convoy coordination, or mission-critical for economic reasons such as the negotiation of the transport contracts. Furthermore, the required properties are not merely qualitative ones but also quantitative ones involving time as well as probabilities. For instance, convoy coordination protocols have to be established between shuttles nearby in the topology, usually involving hard real-time constraints, and the sent protocol message may be lost with a non-zero probability. Consequently, we need methods and tools to guarantee critical quantitative properties when developing such systems, which include *structure dynamics*, *timed behavior*, and *probabilistic behavior*.

Combinations of different modeling approaches have led to a number of new interesting applications in the last couple of years. In the following, we briefly describe related modeling and analysis approaches, which combine some of the aspects of *structure dynamics*, *timed behavior*, and *probabilistic behavior*.

Timed graph transformation systems (TGTSs) [4, 9, 21] facilitate the modeling of timed behavior in graph transformation systems using timed automata concepts.[1] Specifically, nodes can be annotated with real-valued clocks which can be dynamically added and removed from the systems. Rules can include clock constraints as additional application conditions, and clocks can be reset. Using symbolic, *zone*-based representations [7, 19] and an implementation in an extension [21] of the GROOVE tool [15], the state spaces of TGTSs can be explored and analyzed, e.g. for time-bounded reachability checks. Moreover, inductive invariant checking [4] for TGTSs provides a means to deal with infinite-state systems. Thus, TGTSs enable the analysis of combined models with structure dynamics and real-time behavior. However, probabilistic behavior is not supported.

A combination of structure dynamics and probabilistic behavior is supported by probabilistic graph transformation systems (PGTSs) [16], which are an extension of the graph transformation theory with discrete probabilistic behavior. In PGTSs, transformation rules are allowed to have multiple right-hand sides, where each of them is annotated with a probability. The choice for a rule match is nondeterministic, whereas the effect of a rule is probabilistic. This approach can be used to model randomized behavior and on-demand probabilistic failures, such as message loss in unreliable communication channels and supports modeling and analysis by an extension of the HENSHIN [13] tool and a mapping to the PRISM [17] model checker.[2]

Real-time rewrite theories as supported by the executable specification language of Real-time MAUDE [22] facilitate combined modeling of structure dynamics and real-time behavior. Analysis goals include reachability checks for failures of safety properties and model checking of time-bounded temporal logic properties. Such properties are in general not decidable and therefore the provided tool support is incomplete.

Probabilistic rewrite theories implemented in PMAUDE [1] provide a combination of structure dynamics, probabilistic behavior for discrete branching, and stochastic timed behavior. Properties for PRTs are specified using probabilistic temporal logic and checked using discrete event simulation, e.g. using the VESTA tool [26]. However, in order to simulate and analyze models in PMAUDE, *all* nondeterminism has to be resolved, i.e., neither discrete nondeterministic choice nor timed nondeterminism as required for real-time behavior, are allowed.

Probabilistic Timed Automata (PTA) [18] combine the modeling features of Markov decision processes (MDPs) [5] and timed automata (TA) [2, 3] and thereby

[1] An alternative approach for graph transformation systems with time was developed in [10]. However, this approach is not suitable in our context since symbolic state space representations and quantitative analysis methods are not considered in [10].

[2] Also stochastic graph transformation systems (SGTSs) [12] that incorporate stochastic timed behavior into GTSs by including continuous-time probability distributions that describe the average delay of firing of rules, once they are enabled, have been proposed. However, note that they do neither support probabilistic behavior nor real-time behavior as they assume a different model of time.

allow to analyze systems exhibiting both timed and probabilistic phenomena. Analysis goals for PTA include the checking of probabilistic time-bounded reachability, computation of rewards, as well as PTCTL model checking [18]. Such properties can be analyzed for PTA, e.g., using the PRISM tool.

The timed and probabilistic extensions of rewrite systems, specifically rewrite theories in MAUDE variants and GTSs, provide the best coverage for the required modeling features. However, none of the existing models facilitates the modeling and analysis of *all* identified requirements.

To fill the identified gap, we propose to combine and extend the existing models to the formalism of Probabilistic Timed Graph Transformation Systems (PTGTSs) that supports modeling and analysis of *structure dynamics*, *timed behavior*, and *probabilistic behavior*. We introduce the formal model of PTGTSs in this paper and present a mapping of models with finite state spaces to probabilistic timed automata (PTA) that allows to use the PRISM model checker to analyze PTGTS models with respect to PTCTL properties.

This technical report is structured as follows. First, the necessary prerequisites in form of probabilistic timed automata (PTA) are recapitulated in Chapter 2. Then, we introduce Probabilistic Timed Graph Transformation Systems (PTGTSs) in Chapter 3. Subsequently in Chapter 4, we present the tool support for our approach using the graph transformation tool HENSHIN and apply it to model our running example handling a shuttle scenario. Finally in Chapter 5, we consider the analysis of PTGTS models by combining the state space generation of HENSHIN and the PTA model checking of the PRISM tool via a mapping. The technical report is closed with some final conclusions and an outlook on planned future work.

2 Probabilistic Timed Automata

In this chapter we first informally introduce the formalisms of probabilistic [25] resp. timed automata [2, 3] and then combine them to the notion of probabilistic timed automata [18] used for modeling of real-time systems with probability.

Probabilistic automata (PA) were introduced in [25] to add probabilistic choice to finite automata by assigning to each edge a probability. The notion of *discrete probability distribution* plays a central role in the context of PA.

Definition 1 (Discrete Probability Distribution)
Consider a denumerable set A.

- *A function $\mu : A \to [0,1]$ is a discrete probability distribution if $\sum_{a \in A} \mu(a) = 1$.*
- *$Dist(A)$ denotes the set of all discrete probability distributions $\mu : A \to [0,1]$.*

Timed automata (TA) [2, 3] have proven to be a very successful modeling and analysis formalism for real-time systems such as embedded software. TA extend finite automata by making use of clocks, which restrict the behavior of the TA based on invariants, guards, and clock resets making use of constraints over clocks.

Definition 2 (Clock Constraints)
For a set X of clocks $\Phi(X)$ denotes the set of all clock constraints ϕ generated by:

$$\phi ::= x_i \sim c \mid x_i - x_j \sim c \mid \phi \wedge \phi,$$

where $\sim \in \{ <, >, \leq, \geq \}$, $c \in \mathbb{N} \cup \{ \infty \}$ are constants, and $x_i, x_j \in X$ are clocks.

The configurations of TA consist of the current location of the automaton and an assignment of each clock to a current clock value given as a real number, called clock valuation. This clock valuation is used to evaluate clock constraints introduced before to restrict the behavior of an automaton.

Definition 3 (Clock Valuation)
For a set X of clocks $\mathcal{V}(X)$ denotes the set of all functions $v : X \to \mathbb{R}$ called clock valuations, *which are also used in the context of the following notions:*

- ***Clock Reset:*** *Let $v : X \to \mathbb{R}$ and $X' \subseteq X$. Then $v[X' := 0] : X \to \mathbb{R}$ is a* clock reset *such that for any $x \in X$ holds if $x \in X'$ then $v[X' := 0](x) = 0$ else $v[X' := 0](x) = v(x)$.*
- ***Clock Increment:*** *Let $v : X \to \mathbb{R}$ and $\delta \in \mathbb{R}$. Then $v + \delta : X \to \mathbb{R}$ is a* clock increment *such that for any $x \in X$ holds $(v + \delta)(x) = v(x) + \delta$.*
- ***Clock Constraint Satisfaction:*** *Let $v : X \to \mathbb{R}$ and ϕ be some constraint over X. Then $v \models \phi$ denotes that v satisfies the constraint ϕ.*
- ***Initial Clock Valuation:*** *$v_0 : X \to \mathbb{R}$ is the* initial clock valuation *if $v_0(x) = 0$ for every $x \in X$. $\mathcal{V}_0(X)$ is the singleton set containing the (unique) initial clock valuation.*

The formalism of probabilistic timed automata (PTA) is an extension of TA. PTA allow for nondeterministic system behavior and, in addition, a probabilistic choice between follower states using discrete probability distributions over edges. An important feature of PTA are invariants given by clock constraints. Invariants enable the specification of upper time bounds for steps to be executed and, hence, restrict the set of admissible reachable states of a system. In the following we consider the formal definition of PTA in the sense as introduced in [18].

Definition 4 (Probabilistic Timed Automata)
A tuple $A = (S, L_{AP}, s_0, X, I, P, \tau)$ is a probabilistic timed automaton (PTA) *if*

- *S is a finite set of locations,*
- *$L_{AP} : S \to 2^{AP}$ is a labeling function assigning to each location the set of atomic propositions that are true in that location,*
- *s_0 is an initial location with $s_0 \in S$,*
- *X is a finite set of clocks,*
- *$I : S \to \Phi(X)$ is a function assigning to each location a clock constraint (also called an invariant),*
- *$P : S \to 2_{\text{fn}}^{Dist(S \times 2^X)}$ is a function assigning to each location a finite nonempty set*[3] *of discrete probability distributions containing follower locations and corresponding clock resets,*
- *$\tau = (\tau_s)_{s \in S}$ is a family of functions where, for any $s \in S$, $\tau_s : P(s) \to \Phi(X)$ assigns to each $p \in P(s)$ a clock constraint (also called a guard).*

The single step relation describes the behavior of PTA by defining two kinds of steps: timed steps where all clock values are increased by the time elapsed and transition steps where a PTA switches states when allowed by the current clock values according to the used probability distributions without elapsing of time.

Definition 5 (Single Step Relation)
Let $A = (S, L_{AP}, S_0, X, I, P, \tau)$ be a PTA and states of A elements of $S \times \mathcal{V}(X)$. Then the single step relation *is given as follows:*

- ***Timed Step:** $(s, v) \xrightarrow{\delta}{}_{A}^{\text{PTA}} (s, v + \delta)$ if $\delta > 0$ and for each δ' it holds that $0 \leq \delta' \leq \delta$ implies that $v + \delta' \models I(s)$,*
- ***Transition Step:** $(s, v) \xrightarrow{\mu}{}_{A}^{\text{PTA}} (s', v[X' := 0])$ if $X' \subseteq X$, $\mu \in P(s)$, $\mu(s', X') > 0$, $v \models \tau_s(\mu)$, and $v[X' := 0] \models I(s')$.*

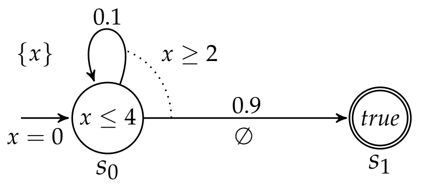
In the following we provide an example coming from the context of the RailCab scenario [23] to demonstrate the behavior of PTA informally.

Example 1 (Probabilistic Timed Automata)
To avoid collisions, and to reduce energy consumption shuttles can communicate and form convoys. We consider a small example for a PTA A, which models the sending of a communication request by a shuttle (see the picture below). The system has a global clock x, which is initially set to 0. Starting in location s_0, a shuttle sends a communication request,

[3]For an arbitrary set M, 2_{fn}^{M} denotes the set of finite nonempty subsets of M.

which can be received by another shuttle in location s_1 with the probability 0.9 or can be lost during the communication process with the probability 0.1. The sending procedure can start after at least two minutes (given by the guard $x \geq 2$) and should be finished after at most four minutes (given by the invariant $x \leq 4$ on the location s_0). If the communication request is lost, the shuttle returns back to the starting position in location s_0, the global clock is reset to 0 (see the label $\{x\}$ at the loop), and the shuttle can repeat the whole procedure again. If the communication request is received successfully, the system terminates in the location s_1, for which only the trivial invariant true *has to be satisfied, without additional resetting of the global clock (see the label $\emptyset$ at the edge between s_0 and s_1). The described probability distribution for the probabilistic step leaving s_0 is called μ in the following exemplary step sequence of the PTA A consisting of timed steps and transition steps.*

$$\begin{aligned}(s_0, \{x \mapsto 0\}) &\xrightarrow{3.4}{}^{\text{PTA}}_{A} (s_0, \{x \mapsto 3.4\}) \xrightarrow{\mu}{}^{\text{PTA}}_{A} (s_0, \{x \mapsto 0\}) \\ &\xrightarrow{2.3}{}^{\text{PTA}}_{A} (s_0, \{x \mapsto 2.3\}) \xrightarrow{\mu}{}^{\text{PTA}}_{A} (s_1, \{x \mapsto 2.3\})\end{aligned}$$



The iterated execution of the sending until location s_1 is reached can be analyzed w.r.t. its timing. In fact, when the communication happens as soon as possible the probability to reach the location s_1 in $2n$ minutes is $1 - 0.1^n$.

According to [18], the underlying model for PTA is given by so-called probabilistic timed structures (PTSs). A PTS is a variant of a Markov decision process (MDP) [5], which is obtained by extension of a timed structure [14] with the probabilistic choice over transitions, i.e., the transition function *Steps* of a PTS results in a choice over pairs consisting of a duration of a transition and a discrete probability distribution over the follower states.

Definition 6 (Probabilistic Timed Structure)
A probabilistic timed structure (PTS) $\mathcal{M} = (Q, Steps, L_{AP})$ *is a labeled MDP where*

- *Q is a set of states,*
- *$Steps : Q \to 2^{\mathbb{R} \times Dist(Q)}$ is a transition function assigning to each state $q \in Q$ a set $Steps(q)$ of pairs (t, p), where $t \in \mathbb{R}$ is a duration of a transition and $p \in Dist(Q)$ is a discrete probability distribution over the follower states,*
- *$L_{AP} : Q \to 2^{AP}$ is a state labeling function.*

Besides the definition of PTA behavior in the form of the single step relation, we also define in the following how PTA give rise to PTSs to enable the comparison of PTA and PTGTS models later on and to be able to make use of the PTCTL logic [18], which has a semantics defined on PTSs as well.

Definition 7 (Induced PTS for a PTA)
Let $A = (S, L_{AP}, s_0, X, I, P, \tau)$ be a PTA. Then $\mathcal{M}_A = (Q_A, Steps_A, L_A)$ is the induced PTS *if*

- $Q_A = \{(s,v) \mid (s_0, v_0) \xrightarrow{*}{}^{\text{PTA}}_{A} (s,v)\}$,
- $L_A(s,v) = L_{AP}(s)$,
- $Steps_A(s,v) = \{\ (\delta, \nu)\ \mid\ (s,v) \xrightarrow{\delta}{}^{\text{PTA}}_{A} (s, v+\delta)\ \wedge\ \nu(s, v+\delta) = 1\ \}$
  $\cup\ \{\ (0,\nu)\ \mid\ (s,v) \xrightarrow{\mu}{}^{\text{PTA}}_{A} (s', v[X' := 0])$
  $\wedge\ \nu(\bar{s}, \bar{v}) = \sum_{\mu \in P(s):\ \mu(s'',X'')>0\ \wedge\ (s,v) \xrightarrow{\mu}{}^{\text{PTA}}_{A} (s'', v[X'':=0]) = (\bar{s},\bar{v})} \mu(s'', X'')\ \}$.

For the case of a transition step we define the probability distribution ν on the possible follower states such that if alternative PTA steps result in identical PTA configurations, the probabilities for their occurrence are added, as required to obtain a well-defined probability distribution.

We use the PTCTL logic defined on PTSs to state various relevant properties on probabilistic real-time systems.

Definition 8 (Syntax of PTCTL)
Let AP be a set of atomic propositions. The set of PTCTL formulas *ψ over AP is given as follows:*

$$\psi ::= true \mid a \mid \phi \mid \psi_1 \wedge \psi_2 \mid \neg\psi_1 \mid Z.\psi_1 \mid \mathcal{P}_{\sqsupseteq\lambda}(\psi_1\ \exists\,\mathcal{U}\ \psi_2) \mid \mathcal{P}_{\sqsupseteq\lambda}(\psi_1\ \forall\,\mathcal{U}\ \psi_2)$$

where $a \in AP$, ϕ is a clock constraint, $Z \subseteq \mathcal{Z}$ is a set of formula clocks disjoint to the clocks in X, ψ_1, ψ_2 are PTCTL formulas, $\lambda \in [0,1]$, $\sqsupseteq \in \{>, \geq\}$, $\mathcal{P}$ is a probabilistic operator from PCTL, and $\mathcal{U}$ is a temporal until-operator from CTL.

The semantics for PTCTL is defined using a satisfaction relation. Let $\mathcal{M} = (Q, Steps, L_{AP})$ be a PTS and Adv_S the set of all its adversaries[4]. Then for an arbitrary state $q \in Q$, clock valuation $v : X \to \mathbb{R}$ with the set X of clocks, and PTCTL formula ψ, the satisfaction relation $(q,v) \models_{Adv_S} \psi$ is defined, according to [18], inductively as follows:

- $(q,v) \models_{Adv_S} true$ is satisfied for all states q and clock valuations v.
- $(q,v) \models_{Adv_S} a$ if a is an atomic proposition associated to the state q by the labeling function L_{AP}.
- $(q,v) \models_{Adv_S} \phi$ if a clock constraint ϕ is *true* when valuated using the clock valuation v.
- $(q,v) \models_{Adv_S} \psi_1 \wedge \psi_2$ if (q,v) satisfies both ψ_1 and ψ_2.
- $(q,v) \models_{Adv_S} \neg\psi_1$ if (q,v) does not satisfy ψ_1.
- $(q,v) \models_{Adv_S} Z.\psi_1$ if ψ_1 is satisfied by q and the valuation obtained by the clock reset $v[Z' := 0]$.
- $(q,v) \models_{Adv_S} \mathcal{P}_{\sqsupseteq\lambda}(\psi_1\ \exists\,\mathcal{U}\ \psi_2)$ if for some adversary $A \in Adv_S$ it holds that the added probability of the paths starting in q and satisfying ψ_1 until satisfying ψ_2 is greater (resp. greater equal) than λ.

[4]An adversary resolves the nondeterminism occurring in an execution trace by choosing the step to be executed in each case. Hence, the choice of the adversary has often a great impact on the satisfaction of a PTCTL formula.

- $(q, v) \models_{Adv_S} \mathcal{P}_{\sqsupseteq\lambda}(\psi_1 \ \forall\, \mathcal{U}\ \psi_2)$ if for all adversaries $A \in Adv_S$ it holds that the added probability of the paths starting in q and satisfying ψ_1 until satisfying ψ_2 is greater (resp. greater equal) than λ.

In our running example, introduced in Chapter 4, we state the desired probabilistic reachability properties and verify them using the PRISM model checker, which is able to check a subset of PTCTL using multiple back-end engines.

3 Probabilistic Timed Graph Transformation Systems

In this chapter we recall the framework of graph transformation systems (GTSs) and introduce a new formalism of Probabilistic Timed Graph Transformation Systems (PTGTSs) allowing for modeling and analysis of structure dynamics, timed behavior as well as probabilistic behavior of systems.

In context of our approach we focus on the formalism of typed graphs. A *graph* $G = (G_V, G_E, s_G, t_G)$ consists of a set G_V of nodes, a set G_E of edges, and source and target functions $s_G, t_G : G_E \to G_V$. For two given graphs $G = (G_V, G_E, s_G, t_G)$ and $H = (H_V, H_E, s_H, t_H)$, a *graph morphism* $f : G \to H$ is a pair of mappings $f_V : G_V \to H_V$, $f_E : G_E \to H_E$ compatible with the source and target functions, i.e., $f_V \circ s_G = s_H \circ f_E$ and $f_V \circ t_G = t_H \circ f_E$.

Let TG be a distinguished graph, called a type graph. Then a typed graph is given by a tuple (G, type) consisting of a graph G together with a graph morphism $\mathit{type} : G \to TG$. For two given typed graphs $G'_1 = (G_1, \mathit{type}_1)$ and $G'_2 = (G_2, \mathit{type}_2)$, a *typed graph morphism* $f : G'_1 \to G'_2$ is a graph morphism $f : G_1 \to G_2$ compatible with the typing functions, i.e., $\mathit{type}_2 \circ f = \mathit{type}_1$.

The adaptation of graphs can be realized using graph transformation rules, which are to be understood as local rule-based modifications defining additions and removals of substructures. A rule $\rho = L \xleftarrow{l} K \xrightarrow{r} R$ is given by a span of injective typed graph morphisms with the graphs L and R called the left-hand resp. the right-hand side of the rule. The transformation procedure defining a graph transformation step is formally introduced by the *DPO approach* [8], where a rule application is defined in terms of category theory by a diagram consisting of two pushouts with total morphisms. For a formal definition of DPO transformation steps see [8].

In the following we introduce the new formalism of PTGTSs. We assume here that all graphs considered in the context of PTGTSs are typed over some type graph TG containing at least a type node *Clock*. Furthermore, for every graph G we use the function $CN(G) = \{n \mid n \in G_V \wedge \mathit{type}_V(n) = \mathit{Clock}\}$ returning all nodes of the type *Clock* contained in G to identify in every graph the nodes used for time measurement only. In the following we call such identified nodes simply *clocks*.

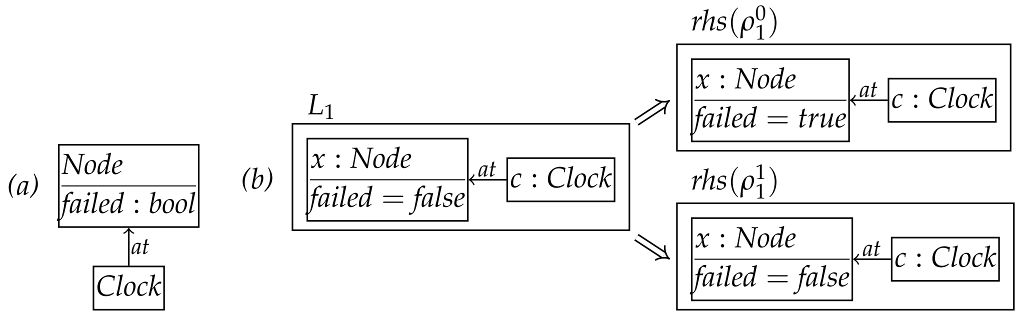
The formalism of PTGTSs is a combination of Probabilistic Graph Transformation Systems (PGTSs) [16] and Timed Graph Transformation Systems (TGTSs) [4, 9, 21]. Similarly to PGTSs, transformation rules in PTGTSs can have multiple right-hand sides, where each of them is annotated with a probability. The choice for a rule match is nondeterministic, whereas the effect of a rule is probabilistic. Similarly to TGTSs, each probabilistic timed graph transformation rule has a guard formulated over clocks contained in the left-hand side of the rule, which is used to control the rule application. Moreover, each rule contains the information about clocks that have to be reset during the rule application.

Definition 9 (Probabilistic Timed Graph Transformation Rule)
$R = (L, P, \mu, \phi, r_C)$ *is a* probabilistic timed graph transformation rule *if*

- *L is a common left-hand side graph,*
- *P is a finite set of graph transformation rules ρ with $lhs(\rho) = L$, where $lhs(\rho)$ provides the left-hand side of the rule ρ,*
- *$\mu \in Dist(P)$ is a probability distribution,*
- *$\phi \in \Phi(CN(L))$ is a guard over nodes of the type Clock contained in the left-hand side graph L,*
- *$r_C \subseteq CN(L)$ is a set of nodes of the type Clock contained in the left-hand side graph L to be reset.*

In the following we give a short example for a probabilistic timed rule.

Example 2 (Probabilistic Timed Graph Transformation Rule)
As an example we model the failure of a hardware node using a probabilistic timed rule fail*, which has two right-hand sides for the case where the node fails with the probability of 10% or not with the probability of 90%. The adjacent clock is used to ensure that the probabilistic timed rule is executed not more than every two time units by resetting the clock during each application and by using a guard that requires the clock to have a value greater than or equal two to capture that the hardware node can fail for not modeled external requests with a minimal arrival time greater than or equal two time units. The underlying type graph for this rule is depicted in the picture (a) below. Formally, the rule is given by fail $= (L_1, P_1, \mu_1, \phi_1, r_{C_1})$ with L_1 as given in the picture (b) to the left, $P_1 = \{\rho_1^0, \rho_1^1\}$ with $rhs(\rho_1^0)$[5] and $rhs(\rho_1^1)$ as given in the picture (b) to the right, $\mu_1 = \{(\rho_1^0, 0.1), (\rho_1^1, 0.9)\}$, $\phi_1 = (c \geq 2)$, and $r_{C_1} = \{c\}$.*



Invariants, as a central concept of PTA, are given for PTGTSs in the form of conditions over clocks that are checked to be satisfied for a given configuration.

Definition 10 (Probabilistic Timed Graph Transformation Invariant)
$\Theta = (L, \phi)$ *is a* probabilistic timed graph transformation invariant *if*

[5] $rhs(\rho)$ denotes the right-hand side of the rule ρ.

- *L is a graph,*
- *$\phi \in \Phi(CN(L))$ is an invariant formula over nodes of the type Clock contained in the graph L.*

For atomic propositions, which are used in the context of the PTCTL logic, we make use of the same kind of conditions as for invariants. For this reason we formally denote an atomic proposition also by $\Theta = (L, \phi)$. The atomic propositions are checked to set the appropriate labels to the PTGTS configurations.

In the following we define PTGTSs comprising the notions introduced above. For a concrete example of a PTGTS see Chapter 4.

Definition 11 (Probabilistic Timed Graph Transformation System)
$S = (TG, G_0, v_0, \Pi, I, AP, prio)$ is a probabilistic timed graph transformation system (PTGTS) *if*

- *TG is a finite type graph including the type node Clock,*
- *G_0 is a finite initial graph over TG,*
- *$v_0 : CN(G_0) \to \mathbb{R}$ is the initial clock valuation assigning the clock value 0 to every node of the type Clock in G_0,*
- *Π is a finite set of probabilistic timed rules,*
- *I is a finite set of probabilistic timed invariants,*
- *AP is a finite set of probabilistic timed atomic propositions,*
- *$prio : \Pi \to \mathbb{N}$ is a priority function assigning a priority to each rule.*[6]

As a next step we define when a PTGTS configuration consisting of a graph and a current clock valuation satisfies some invariant.

Definition 12 (Probabilistic Timed Invariant Satisfaction)
Let $S = (TG, G_0, v_0, \Pi, I, AP, prio)$ be a PTGTS, $\Theta = (L, \phi) \in I$, G be a graph typed over TG, and $v \in \mathcal{V}(CN(G))$ be a clock valuation for nodes of the type Clock in G. Then $(G, v) \models \Theta$ if for every injective match $m : L \to G$ with $m = (m_V, m_E)$ it holds that $v \circ m_V \models \phi$.

Not every configuration of a PTGTS reached by rule based structure adaptation is valid since the invariants of the PTGTS need to be considered, too.

Definition 13 (Probabilistic Timed Graph Transformation State)
Let $S = (TG, G_0, v_0, \Pi, I, AP, prio)$ be a PTGTS. Then $q = (G, v)$ is a probabilistic timed graph transformation state *(also called configuration) of S (written $q \in states(S)$) if*

- *G is a graph typed over TG,*
- *$v \in \mathcal{V}(CN(G))$ is a clock valuation for nodes of the type Clock in G,*
- *$(G, v) \models \Theta$ for each $\Theta \in I$.*

[6]For the priority function it holds that the higher the number assigned to a rule the higher is the priority of the rule.

The behavior of PTGTSs is defined by the probabilistic timed graph transformation steps. We distinguish here similarly to PTA two kinds of steps: timed steps increasing the clock values by the time elapsed and transition steps allowing to switch a configuration under certain conditions as defined below. For the transition steps we ensure that the guard of the rule is satisfied by the current clock valuation and match, that no transition step with higher priority can be executed, and that all steps in the selected probability distribution μ are enabled. Furthermore, considering a rule ρ with non-zero probability $\mu(\rho)$, we define the single rule single step relation based on the expected DPO transformation step and ensure that the clock valuations of the source and target states are compatible also enforcing the clock resets of the rule.

Definition 14 (Probabilistic Timed Graph Transformation Step)
Let $S = (TG, G_0, v_0, \Pi, I, AP, prio)$ be a PTGTS. Then the single step relation *is given as follows:*

- ***Timed Step:*** $(G, v) \xrightarrow{\delta}{}_S^{\mathrm{PTGTS}} (G, v + \delta)$ *if* $\delta > 0$ *and for each* δ' *it holds that* $0 \leq \delta' \leq \delta$ *implies that* $(G, v + \delta') \in states(S)$.
- ***Transition Step:*** $(G_1, v) \xrightarrow{R,\rho,m}{}_S^{\mathrm{PTGTS}} (G_2, v')$ *if*
  - $R = (L, P, \mu, \phi, r_C) \in \Pi$ *is a probabilistic timed rule,*
  - $m : L \rightarrow G_1$ *with* $m = (m_V, m_E)$ *is an injective match,*
  - $v \circ m_V \models \phi$,
  - $\rho \in P$ *is a transformation rule with non-zero probability* $\mu(\rho) > 0$,
  - $\nexists G_2', v'', R', \rho', m'$ *such that* $(G_1, v) \xrightarrow{R',\rho',m'}{}_S^{\mathrm{PTGTS}} (G_2', v'')$ *and* $prio(R') > prio(R)$,
  - $(G_1, v) \overset{R,\rho,m}{\twoheadrightarrow}{}_S^{\mathrm{PTGTS}} (G_2, v')$,
  - $\forall \rho' \in P \setminus \{\rho\}$ *such that* $\mu(\rho') > 0$ *there is a graph* G_2' *such that* $(G_1, v) \overset{R,\rho',m}{\twoheadrightarrow}{}_S^{\mathrm{PTGTS}} (G_2', v')$,

  where $(G_1, v) \overset{R,\rho,m}{\twoheadrightarrow}{}_S^{\mathrm{PTGTS}} (G_2, v')$ *is the single rule single step relation if*
  - $(G_1, v), (G_2, v') \in states(S)$,
  - $\rho = (L \xleftarrow{l} K \xrightarrow{r} R)$ *is a graph transformation rule,*
  - $(1) + (2)$ *is a DPO diagram for the transformation step* $G_1 \overset{\rho,m}{\Longrightarrow} G_2$,
  - *clock valuation functions* $v : CN(G_1) \rightarrow \mathbb{R}$ *and* $v' : CN(G_2) \rightarrow \mathbb{R}$ *are compatible, i.e.,* $\forall X \in CN(G_1).\ (\forall Y \in CN(D).\ (l_V'(Y) = X) \Rightarrow (v'(r_V'(Y)) = v[m_V(r_C) := 0](X)))$[7]
  - *the clock value* 0 *is assigned to all created nodes of the type Clock, i.e.,* $\forall Z \in CN(G_2) \setminus r_V'(CN(D)).\ v'(Z) = 0$.

[7]For morphisms between clocks we omit the restricted notation $f_V|_{CN(G_1)} : CN(G_1) \rightarrow CN(G_2)$ and use the unrestricted notation $f_V : CN(G_1) \rightarrow CN(G_2)$ to simplify the representation.

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d, "k"] \\
G_1 & D \arrow[l, "l'"'] \arrow[r, "r'"] & G_2
\end{tikzcd}
\qquad
\begin{tikzcd}
CN(L) \arrow[d, "m_V"'] \arrow[dr, phantom, "="] & CN(K) \arrow[l, "l_V"'] \arrow[r, "r_V"] \arrow[d] \arrow[dr, phantom, "="] & CN(R) \arrow[d, "k_V"] \\
CN(G_1) \arrow[dr, "v"'] & CN(D) \arrow[l, "l'_V"'] \arrow[r, "r'_V"] & CN(G_2) \arrow[dl, "v'"] \\
& \mathbb{R} &
\end{tikzcd}$$

In our subsequent translation of PTGTSs into the corresponding PTSs we identify configurations of PTGTSs up to isomorphism. For this purpose we now introduce such isomorphisms.

Definition 15 (Isomorphisms on States of PTGTSs)
Let S be a PTGTS and (G_1, v_1), $(G_2, v_2) \in states(S)$. Then $(G_1, v_1) \cong (G_2, v_2)$ if

- *$i : G_1 \rightarrow G_2$ with $i = (i_V, i_E)$ is an isomorphism,*
- *$\forall X \in CN(G_1)$ it holds that $v_1(X) = v_2(i_V(X))$.*

$$\begin{tikzcd}
CN(G_1) \arrow[rr, "i_V"] \arrow[dr, "v_1"'] & {=} & CN(G_2) \arrow[dl, "v_2"] \\
& \mathbb{R} &
\end{tikzcd}$$

Analogously to PTA we provide a PTS for every PTGTS and, hence, allow for a comparison of PTA and PTGTSs by comparing their semantics in the sense of the corresponding PTSs.

Definition 16 (Induced PTS for a PTGTS)
Let $S = (TG, G_0, v_0, \Pi, I, AP, prio)$ be a PTGTS. Then $\mathcal{M}_S = (Q_S, Steps_S, L_S)$ is the induced PTS if

- $Q_S = \{[(G,v)]_{\cong} \mid (G_0, v_0) \xrightarrow{*}{}_S^{\text{PTGTS}} (G,v)\}$,
- $L_S([(G,v)]_{\cong}) = \{\Theta \in AP \mid (G,v) \models \Theta\}$,
- $Steps_S([(G,v)]_{\cong}) = \{\ (\delta, \nu)\ \mid\ (G,v) \xrightarrow{\delta}{}_S^{\text{PTGTS}} (G', v')\ \wedge\ \nu([(G', v')]_{\cong}) = 1\ \}$
$\cup\ \{\ (0, \nu)\ \mid\ R = (L, P, \mu, \phi, r_C) \in \Pi \wedge \rho \in P \wedge (G,v) \xrightarrow{R,\rho,m}{}_S^{\text{PTGTS}} (G', v')\ \}$
where $\nu([(\overline{G}, \overline{v})]_{\cong}) = \sum_{\rho' \in P : (G,v) \xrightarrow{R,\rho',m}{}_S^{\text{PTGTS}} (G'', v'') \cong (\overline{G}, \overline{v})} \mu(\rho')$.

In the induced PTS we consider configurations up to isomorphism using their equivalence classes and derive the labeling of configurations by evaluating the atomic propositions of the PTGTS. For the step relation we need to collate PTGTS steps with common target when constructing the probability distribution ν to ensure well-definedness.

We furthermore employ negative application conditions (*NAC*s) [11] and attributes for PTGTSs. They allow to increase the descriptive expressiveness of the rules and can be added straightforwardly to the presented formalization.

4 Modeling

To support modeling and, subsequently, analysis of PTGTS models with their probabilistic and timed behavior, we extended the existing support of the HENSHIN tool [13] for PGTSs [16]. Analogously to PGTSs, the elements and links between the elements are captured by an EMF model represented as a class diagram (the type graph of the PTGTS as given in Figure 4.1a). In addition, we require a *Clock* element to be present in the EMF model to enable the modeling of timed behavior. The probabilistic choices are modeled as for PGTSs with multiple HENSHIN transformation rules with the same name and the same left-hand side (e.g., depicted in Figures 4.2d and 4.2e for the rule *connect*). To support the modeling of real-time behavior, we associate clock guards (CG), clock invariants (CI), and clock resets (CR) to the rules via corresponding annotations added to the property list of the GTS in HENSHIN. Consequently, the HENSHIN model includes all details of a PTGTS model. Since HENSHIN does not include rules' annotations in their visual representation we label the rules in this paper (e.g., in Figure 4.2) with (CG); (CI); (CR) where void elements are represented by —.

Syntactically, a rule $L \xleftarrow{l} K \xrightarrow{r} R$ in HENSHIN is given by a single graph annotated with specific stereotypes. The stereotypes *«preserve»*, *«delete»*, and *«create»* correspond to the elements of K, L without K, and R without K, respectively. The stereotype *«forbid»* is used to specify $\mathcal{NAC}$s and can be parametrized as in *«forbid#n»* for $n \in \mathbb{N}$ to distinguish between multiple $\mathcal{NAC}$s.

As a running example, we model a scenario inspired by the RailCab project [23], where a service choreography coordinates the movement of shuttles on tracks, as a PTGTS using HENSHIN. The type graph and the initial track topology are given in Figure 4.1. In the context of our scenario, tracks are connected to the adjacent tracks by *next* edges. Shuttles are located on tracks, which is represented by *at* edges. Shuttles can move forward on tracks being in the *DRIVE* mode or can initiate emergency brakes changing to the *BRAKE* mode to avoid collisions. To avoid collisions shuttles can also communicate and establish connections. A connection is associated with the leading shuttle via a *target* edge and with the following shuttle via a *source* edge. The connection attempt between two shuttles may fail, but it can be repeated after both involved shuttles moved one track forward. This aspect is expressed by the shuttle's attribute *canConnect*. Two connected shuttles are allowed to be at the same track without being involved in a collision. In the initial topology parallel tracks leading to the same track after one, two or more successor tracks are marked by *conflict* nodes. Two shuttles can try to establish a connection if they are on tracks connected by a *conflict* node. Another reason for communication of the shuttles is the reduction of energy consumption. For this reason shuttles can form convoys also establishing a connection. We also equip shuttles and tracks

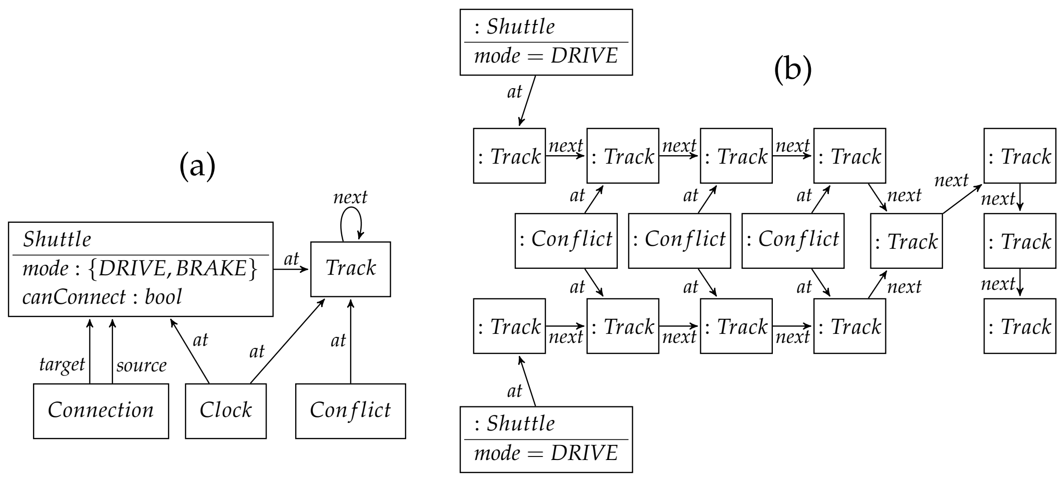


Figure 4.1: Type graph of the shuttle scenario (a) and topology with 3 conflict nodes (b)

with *clocks* needed for time measurement only to be able to control the time for rule applications. Note that we do not depict the nodes of the type *Clock* in the rules explicitly to keep the rule representation concise but use the annotation *e.c* to refer to the clock *c* linked to some element *e*.

The behavior of the shuttle scenario is modeled in HENSHIN using the following probabilistic timed rules. Shuttles can drive alone or can build convoys to reduce the energy consumption. The rule *driveAlone* (see Figure 4.2a) allows a shuttle that is leading a convoy or a shuttle driving without a convoy to move forward if there are no shuttles located at the track after the subsequent track and any of that track's predecessor tracks. The four *driveConvoy* rules (see Figure 4.3) allow a shuttle to follow a leading shuttle depending on the layout of single tracks in the current situation. Following the aim to reflect the real-time behavior properly, we require that moving on a single track can take between 3 and 4 minutes, which we express using the corresponding guards and invariants, respectively, formulated over the track clocks for the driving rules. For the rule *driveAlone*, the corresponding guard is given by the annotation $t1.c \geq 3$ and the corresponding invariant is not depicted in Figure 4.2a but is given by $\Theta_{driveAlone} = (lhs(driveAlone), t1.c \leq 4)$ for a track $t1$ with its clock c. To be able to measure the time spent on a track properly, we reset the clock of a track to which a shuttle is moving after applying one of the driving rules. Considering again the rule *driveAlone*, the corresponding clock reset is given by the annotation $t2.c' = 0$ for a track $t2$ with its clock c'.

Shuttles may connect with each other to create convoys and to prevent collisions. Figures 4.2d and 4.2e depict the probabilistic timed rule *connect* allowing two driving shuttles located at parallel critical tracks to communicate and to create a convoy. The *NAC*s of the rule express intuitively that there must not already exist a connection between the two considered shuttles, the shuttle chosen as the leader must not be leading another convoy, and the shuttle chosen as the following shuttle must not be following another shuttle. Since a connection request can be lost after

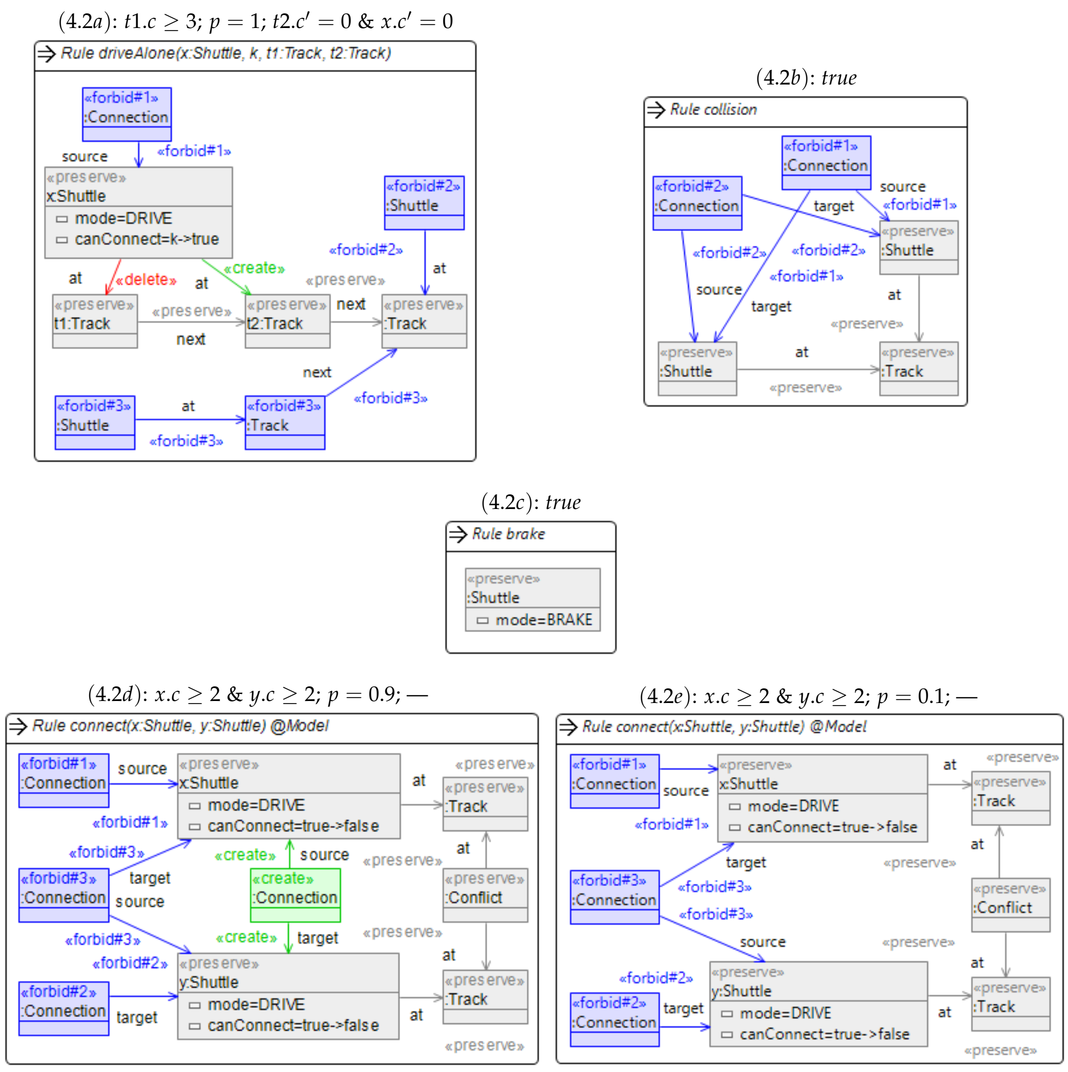


Figure 4.2: Rules *driveAlone* and *connect* as well as atomic propositions *collision* and *brake*

its sending, the rule *connect* has two different right-hand sides representing on the one hand, the case that the connection is established successfully as depicted in Figure 4.2d (which happens with the probability of 90%) and on the other hand the situation that the connection request has been lost as depicted in Figure 4.2e (which occurs with the probability of 10%). We assume furthermore that each communication attempt takes at least two minutes for each shuttle and model this behavior by using the corresponding guard $x.c \geq 2$ & $y.c \geq 2$ (for two shuttles x, y and their respective clocks c) for communicating shuttles in both basic rules *connect* (see Figures 4.2d and 4.2e). Moreover, since a communication attempt can be repeated only after each of the shuttles has moved one track forward, we reset a clock of a shuttle each time it has used the rule *driveAlone* after a communication attempt, which is formulated by the corresponding clock reset $x.c' = 0$ of the rule *driveAlone* for a shuttle x with its clock c'.

After establishing a connection, which leads to the creation of a convoy, the follower shuttle moves forward using one of the rules *driveConvoy* given in Figure 4.3. The rule *driveConvoy1* (see Figure 4.3a) allows a shuttle following another shuttle to move to the track on which the leading shuttle is currently located. The rule *driveConvoy2* (see Figure 4.3b) allows a shuttle in a convoy to move to the next track if its leading shuttle is two tracks ahead. The rule *driveConvoy3* (see Figure 4.3c) allows a shuttle in a convoy to move to the subsequent track if that track is parallel (connected by a *conflict* node) to its leading shuttle's current position. The rule *driveConvoy4* (see Figure 4.3d) allows a shuttle in a convoy to move to the subsequent track if that track's successor track is parallel to the leading shuttle's current position. To be able to express that moving on a single track can take between 3 and 4 minutes, we use similar to the rule *driveAlone* the corresponding guards and invariants formulated over the track clocks. For each of the rules *driveConvoyI* for $I \in \{1, 2, 3, 4\}$ the corresponding guard is given by the annotation $t1.c \geq 3$ and the corresponding invariant is not depicted in Figure 4.3 but is given by $\Theta_{driveConvoyI} = (lhs(driveConvoyI), t1.c \leq 4)$ for a track $t1$ with its clock c. To be able to measure the time spent on a track properly, we reset the clock of a track to which a shuttle is moving after applying one of the *driveConvoyI* rules. The corresponding clock reset for each of the *driveConvoyI* rules is given by the annotation $t2.c' = 0$ for a track $t2$ with its clock c'.

In the case if no connection attempt was successful, a shuttle, which is driving behind another one, has to brake to avoid a collision, if both shuttles come too close to each other. The rules for the execution of emergency brakes are given in the Figure 4.4. If a shuttle without a leader is two tracks behind another shuttle, it may move to the adjacent track and switch then from the driving into the braking mode, which is ensured by the rule *brake1* (see Figure 4.4a). Furthermore, if an unconnected shuttle is on a track whose successor is parallel to a track occupied by another shuttle with both parallel tracks having the same successor track, it may move to the subsequent track and switch from the driving into the braking mode according to the rule *brake2* given in Figure 4.4b. The *brake* rules do not depend on the current clock valuation and are applied solely based on the current graph structure, hence, both rules have only the non-restricting guard *true* as well as no invariants and clock resets.

After a shuttle has changed into braking mode, the system detects the safety-critical situation and terminates using the rule *cleanupError* (see Figure 4.5a). The rule *cleanupError* deletes both shuttles from the considered topology to establish a terminal state representing an unsafe run of the system and adds two isolated tracks to the system to distinguish terminal states. Otherwise, if one of both shuttles has reached the next-to-last track of the system and both shuttles are still in the driving mode, the system terminates using the rule *cleanupOk* (see Figure 4.5b), which establishes the terminal state that represents a safe run of the system by deleting both shuttles from the considered topology. As before, to distinguish terminal states, one isolated track is added to the system. Similar to the *brake* rules,

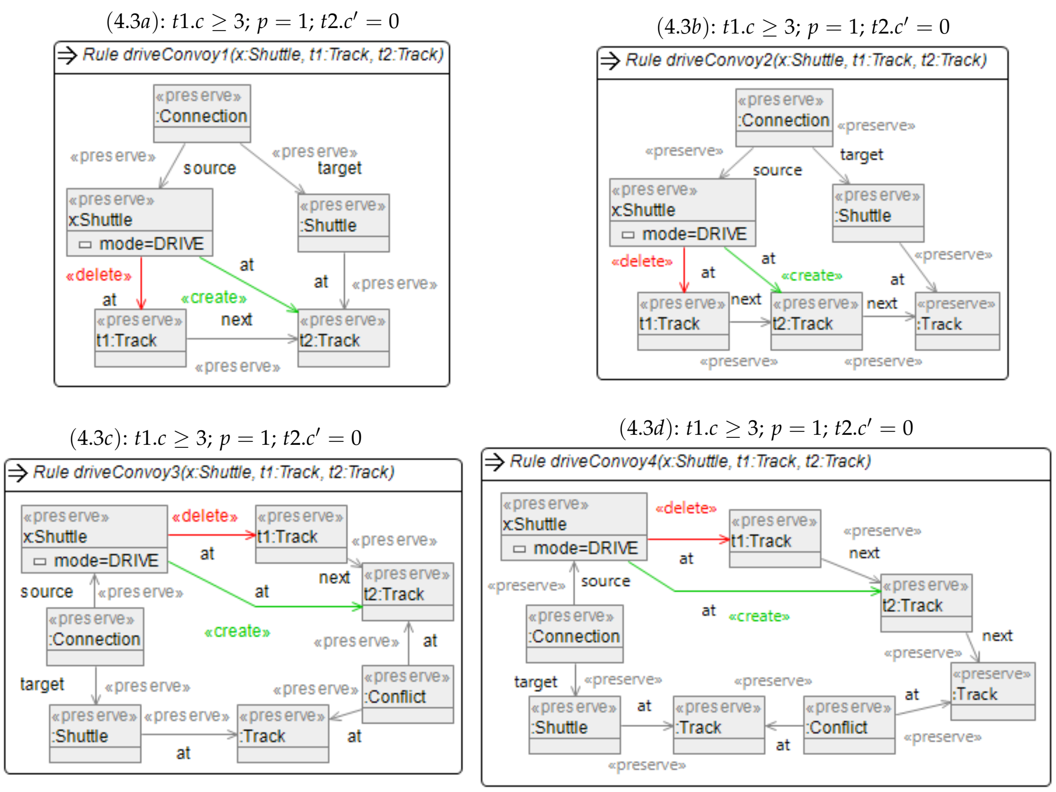


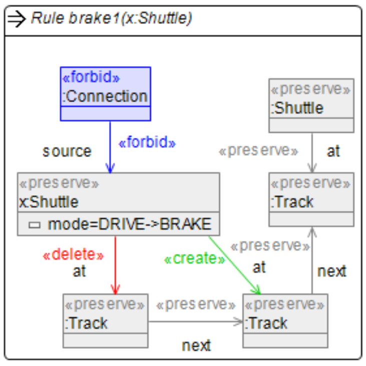
Figure 4.3: Rules *driveConvoy*

the *cleanup* rules do not depend on the current clock valuation and are applied solely based on the current graph structure, hence, both rules have only the non-restricting guard *true* as well as no invariants and clock resets.

In the context of our shuttle scenario, we consider two atomic propositions *collision* and *brake* modeled as non-changing rules in HENSHIN as depicted in Figures 4.2b and 4.2c, respectively. The atomic proposition *collision* depicts the situation when a collision occurs, which means that two shuttles are located at the same track without having a connection, while the atomic proposition *brake* shows a shuttle in braking mode. Using these atomic propositions we can mark the configurations in the state space that satisfy these atomic propositions and subsequently use these marked configurations for the analysis of the system. Since atomic propositions are used in the context of our example for marking purposes only, we equip them always with the clock constraint *true*.

In our running example we also make use of priorities (also for the non-changing rules representing atomic propositions) to ensure (a) that collisions are detected properly ($prio(collision)$=2 as a highest priority for the atomic proposition *collision*), (b) that the rule *cleanupError* and the atomic proposition *brake* detect emergency brakes immediately ($prio(brake)$=$prio(cleanupError)$=2), and (c) that the rule *connect* is applied whenever possible to guarantee that the shuttles do not continue to drive

(4.4*a*): *true*; $p = 1$; —



(4.4*b*): *true*; $p = 1$; —

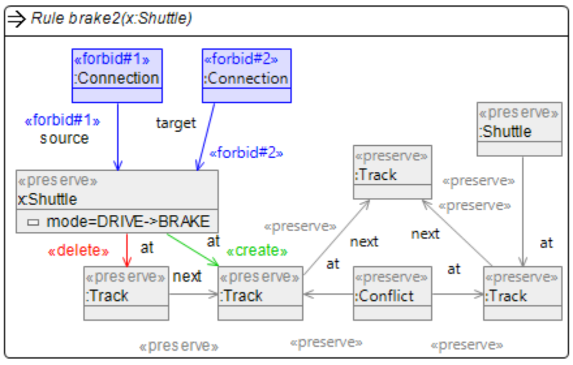
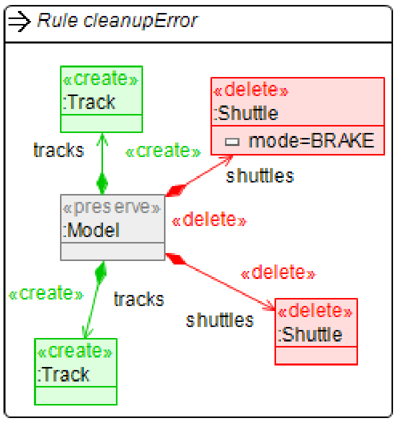


Figure 4.4: Rules *brake*

(4.5*a*): *true*; $p = 1$; —



(4.5*b*): *true*; $p = 1$; —

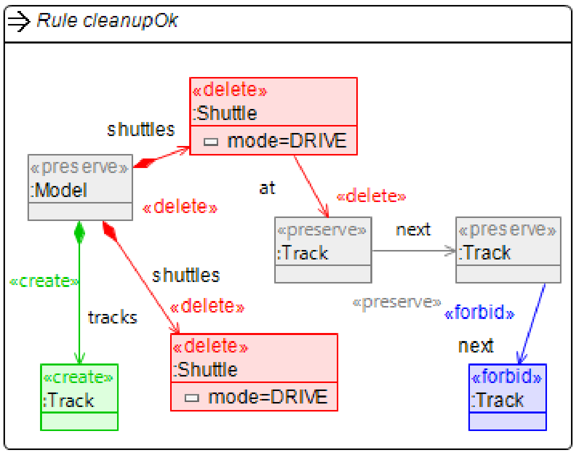


Figure 4.5: Rules *cleanup*

alone without executing connection attempts (*prio*(*connect*)=1). All other rules have the default priority 0.

The extension of HENSHIN as well as the full details of the shuttle scenario modeled as a PTGTS are available at `http://mde-lab.de/ptgts-checking`.

5 Analysis

As outlined in the previous chapter, we can model PTGTSs using the HENSHIN tool. To analyze a PTGTS model, we can chain together the capabilities of the HENSHIN tool for GTSs and the PRISM model checker for PTA.

In *Step 1* we use the capability of HENSHIN to generate the state space of a GTS by considering the probabilistic choices between the different right-hand sides of probabilistic timed rules as if they were nondeterministic and ignoring clock guards, clock resets as well as clock invariants. In *Step 2* we extend our mapping from PGTSs to PA [16] to be able to convert the state space generated for the PTGTS into the corresponding PTA. In this step we replace the nondeterministic choice between the different right-hand sides of probabilistic timed rules by probabilistic transitions, including clock guards and clock resets as well as adding atomic propositions and the set of clock invariants, which must hold for all valid states of the system. The PTA generated in this way has an input format of PRISM allowing to verify PTCTL properties by computing the corresponding minimum and maximum probabilities. Finally, in *Step 3* we can model check the resulting PTA with PRISM according to the properties of interest for the PTGTS. Note that this tool chain can only be used in practice, if the state space generation using HENSHIN terminates in *Step 1* and results in a finite state space of moderate size, which PRISM is capable to analyze.

For the shuttle PTGTS described in the previous chapter, we executed several experiments by using this outlined tool chain.

In our first experiment we determined using PRISM the states generated by HENSHIN that remain reachable with non-zero probability when considering the timing behavior. Thereby we exclude many of the calculated traces from the further analysis since they do not satisfy the corresponding time constraints. In Figure 5.2 we have visualized for the topology with 3 conflict nodes (given in Figure 4.1b) the reachable part of the GTS state space calculated by HENSHIN as depicted in Figure 5.1. Moreover, for the shuttle example PRISM detects for the topologies with 2–6 conflict nodes 53.1%, 58.0%, 62.0%, 64.3%, and 65.8% of the states generated by HENSHIN to be non-reachable as visualized in Figure 5.3.

In our second experiment we analyzed whether the considered shuttle scenario can under any circumstances exhibit a collision. In fact, collisions between shuttles cannot occur due to the nature of the contained transformation rules, which ensure that emergency brakes are applied if necessary. This can be verified already in HENSHIN after *Step 1* by using the atomic proposition *collision* (see Figure 4.2b) with the highest priority detecting a collision by marking configurations in the state space where the atomic proposition can be matched. In HENSHIN we then observe that this atomic proposition labels no state and, hence, no additional analysis using PRISM is required.

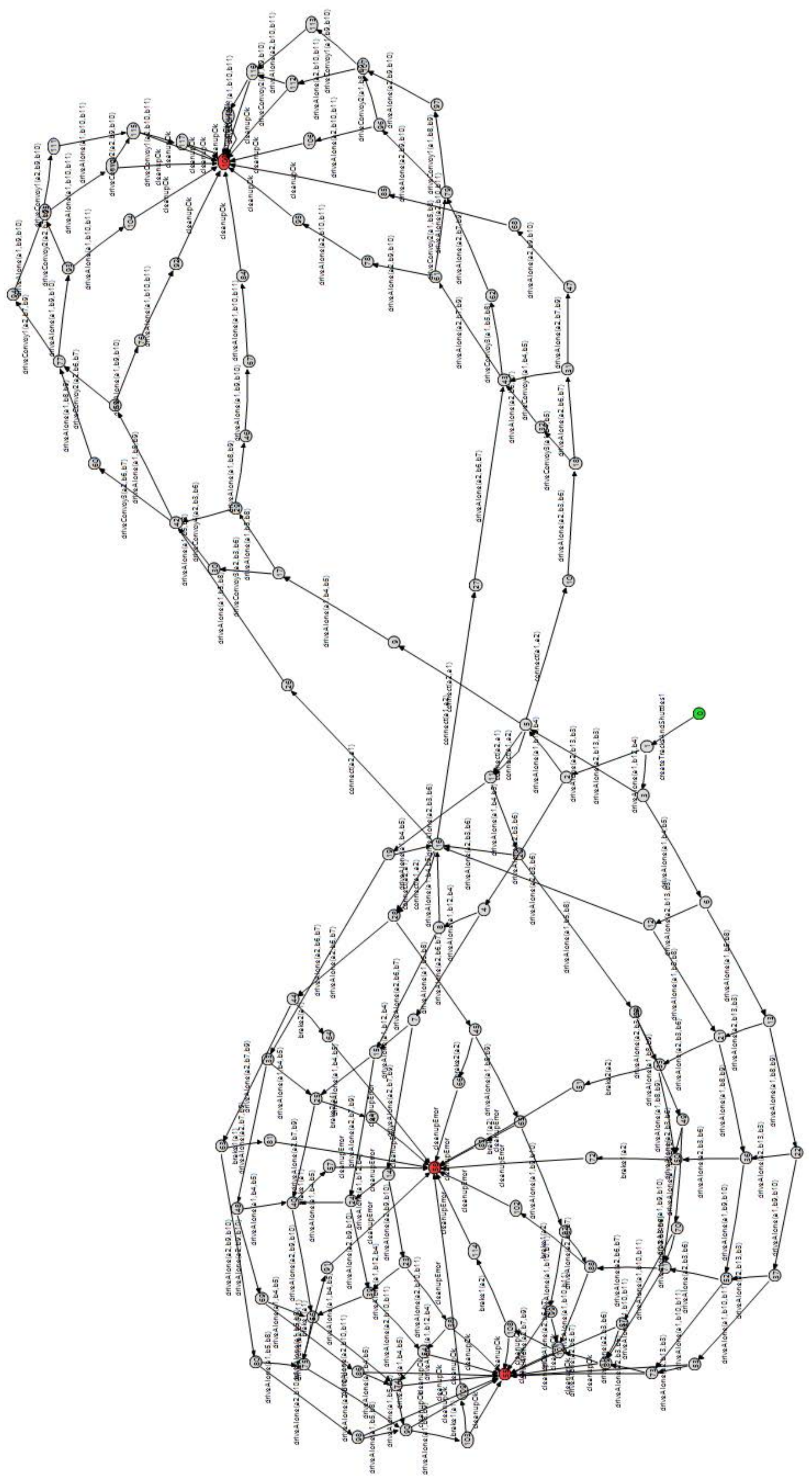


Figure 5.1: State space generated by HENSHIN for the topology containing 3 conflict nodes with the green start configuration and the red end configurations

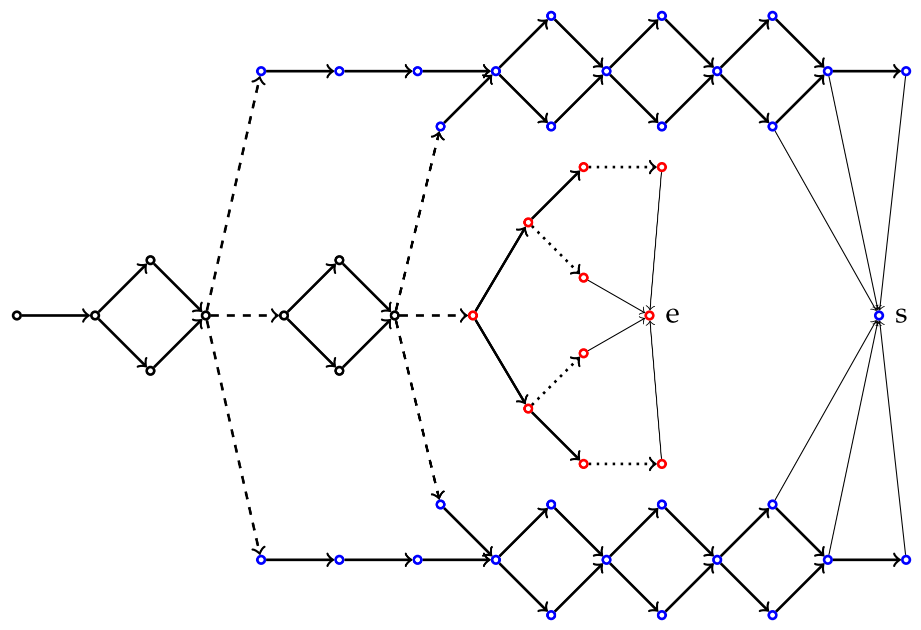


Figure 5.2: Reachable part (w.r.t. the timed behavior) of the GTS state space for the topology with 3 conflict nodes. Legend: drive rules (thick, continuous arrows), connect rule (dashed arrows), cleanup rules (thin, continuous arrows), brake rules (dotted arrows), will not make an emergency brake (blue nodes), will make an emergency brake (red nodes), will or will not make an emergency brake (black nodes), no emergency brake occurred (node s), and emergency brake occurred (node e)

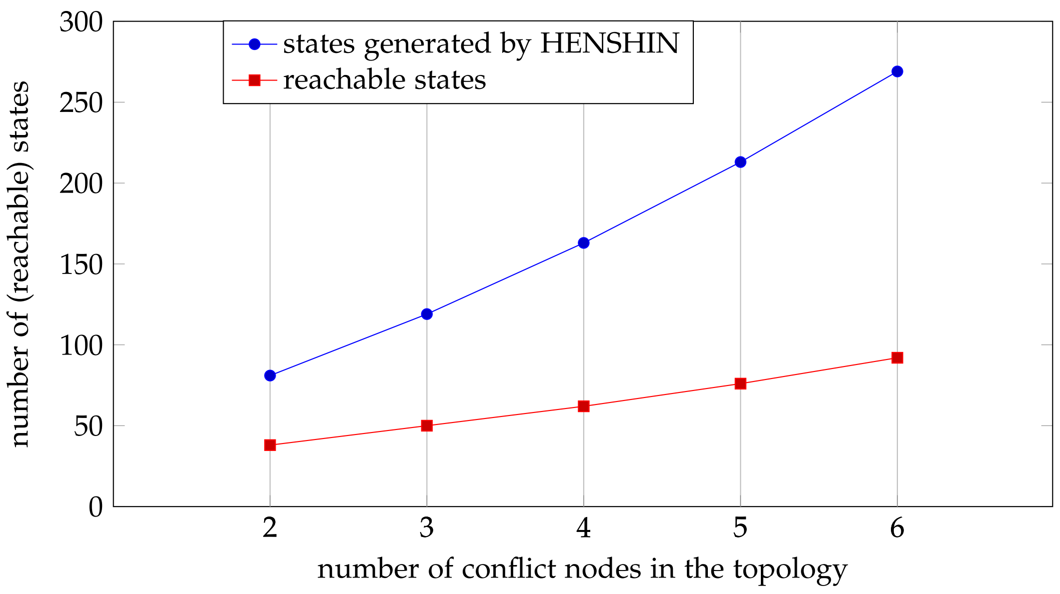


Figure 5.3: Fraction of states generated by HENSHIN in *Step 1* that satisfy the time constraints and thus are reachable in the PTA resulting from *Step 2*

Finally, in our third experiment we verified using PRISM the maximal probability with which the described shuttle system executes an emergency brake. For this reason we generated first using HENSHIN in 11.8–17.5 seconds state spaces with 81–269 states for the different topologies with 2–6 conflict nodes, respectively. In the PTCTL notation our property of interest can be given by $\mathcal{P}_{\geq\lambda}(\mathit{true}\ \exists\mathcal{U}\ \mathit{brake})$ for the atomic proposition *brake* given in Figure 4.2c where PRISM automatically returns the maximal probability value for λ by checking the equivalent property $\mathit{Pmax} =?\ [F\ \text{“}\mathit{brake}\text{”}]$ where F is the exists-eventually-operator. In Figure 5.4 we show for topologies with 2–6 conflict nodes, which also determine the initial distance of the shuttles to the critical track element, how the corresponding maximal probabilities depend on the likelihood of the successful connection establishment. As expected, the lower the probability for a non-successful connection attempt (x-axis) the lower the maximal probability for emergency brake execution (y-axis), which is the worst case scenario.[8] The computation of the probability values using PRISM required for topologies with 2–6 conflict nodes 0.3–181.7 seconds, respectively.

We can conclude that our running example modeled as a PTGTS behaves as desired because (a) collisions are avoided altogether and (b) the worst case probabilities for emergency brake can be controlled using the number of conflict nodes based on the likelihood of unsuccessful connection attempts. Our experiments demonstrate that we can analyze PTGTSs by employing the explained tool chain of HENSHIN operating on GTSs and PRISM analyzing PTA.

[8]The range of the considered probabilities for non-successful connection attempts has been taken from [20] where for close range communication and high data rates an error rate of at most 13% has been observed for wireless communication.

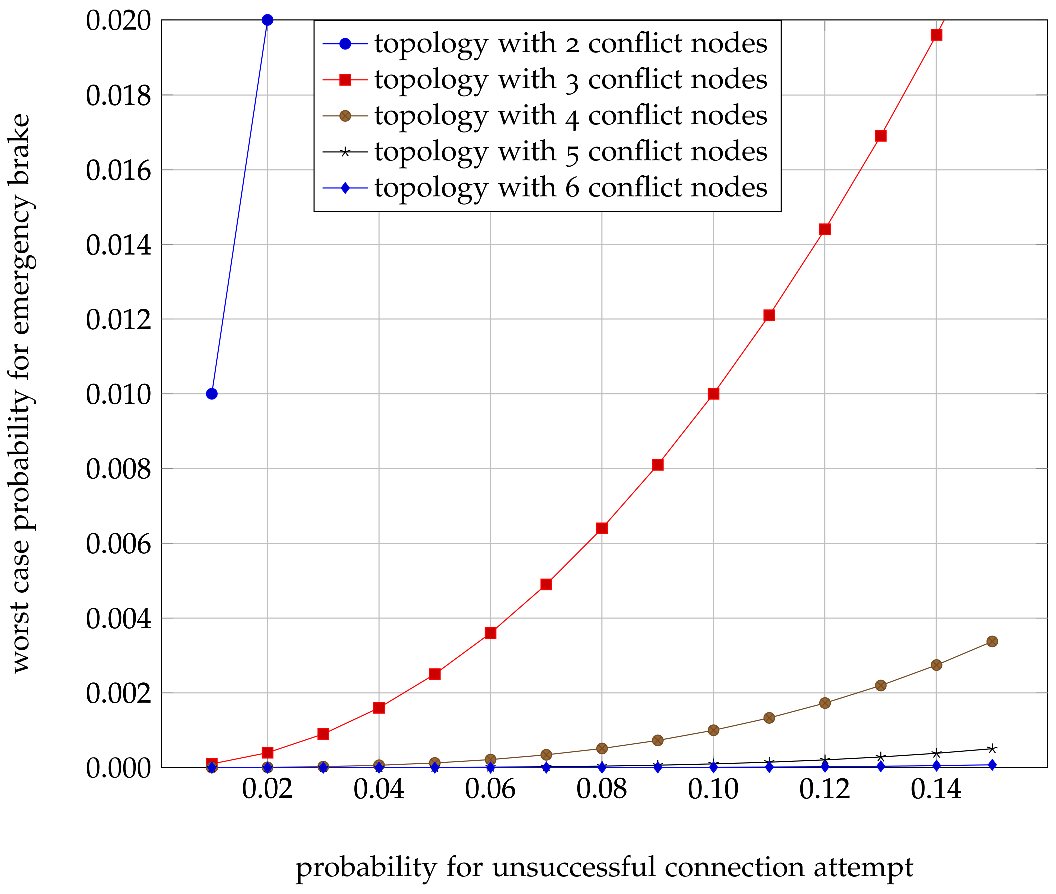


Figure 5.4: Visualization for the probability of an emergency brake

6 Conclusion and Future Work

In this paper we introduced Probabilistic Timed Graph Transformation Systems (PTGTSs) as a high-level description language supporting all the necessary aspects of structure dynamics, timed behavior, and probabilistic behavior that we identified as relevant for the next generation of embedded real-time systems employing the service-oriented paradigm. We presented the formal model of PTGTSs and outlined a mapping of PTGTS models employing the HENSHIN tool to probabilistic timed automata (PTA) such that the PRISM model checker can be used to analyze PTGTS models with respect to PTCTL properties.

As a future work we plan to provide a specification formalism operating on PTGTS to be able to state more complex properties on the structure dynamics, timed behavior, and probabilistic behavior of the given PTGTS model in a coherent way. Such an extension is then to be included in the mapping of PTGTS to PTA to allow for their automated verification using PRISM.

References

[1] G. Agha, J. Meseguer, and K. Sen. "PMaude: Rewrite-based Specification Language for Probabilistic Object Systems". In: *Electron. Notes Theor. Comput. Sci.* 153 (2 2006)., pages 213–239. ISSN: 1571-0661.

[2] R. Alur, C. Courcoubetis, and D. L. Dill. "Model-Checking in Dense Real-time". In: *Inf. Comput.* 104.1 (1993), pages 2–34. DOI: 10.1006/inco.1993.1024.

[3] R. Alur and D. L. Dill. "A Theory of Timed Automata". In: *Theor. Comput. Sci.* 126.2 (1994), pages 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[4] B. Becker and H. Giese. "On Safe Service-Oriented Real-Time Coordination for Autonomous Vehicles". In: *Proc. ISORC'08*. IEEE Computer Society Press, 2008, pages 203–210.

[5] R. Bellman. "A Markovian Decision Process". In: *Indiana Univ. Math. J.* 6 (4 1957), pages 679–684. ISSN: 0022-2518.

[6] B. Bouyssounouse and J. Sifakis, editors. *Embedded Systems Design: The ARTIST Roadmap for Research and Development*. LNCS 3436. Springer, 2005.

[7] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. "The tool KRONOS". In: *Proc. Hybrid Systems'95*. LNCS 1066. Springer, 1996, pages 208–219. ISBN: 3-540-61155-X.

[8] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, 2006.

[9] H. Giese. "Modeling and Verification of Cooperative Self-adaptive Mechatronic Systems". In: *Proc. Monterey Workshop 2005*. LNCS 4322. Springer, 2007, pages 258–280.

[10] S. Gyapay, D. Varró, and R. Heckel. "Graph transformation with time". In: *Fundamenta Informaticae* 58 (1 2003), pages 1–22. ISSN: 0169-2968.

[11] A. Habel, R. Heckel, and G. Taentzer. "GRAPH GRAMMARS WITH NEGATIVE APPLICATION CONDITIONS". In: *Fundam. Inf.* 26.3,4 (Dec. 1996), pages 287–313. ISSN: 0169-2968.

[12] R. Heckel, G. Lajios, and S. Menge. "Stochastic Graph Transformation Systems". In: *Fundamenta Informaticae* 74 (1 2006)., pages 63–84. ISSN: 0169-2968.

[13] *EMF Henshin*. The Eclipse Foundation. 2013.

[14] T. A. Henzinger and O. Kupferman. "From Quantity to Quality". In: *Hybrid and Real-Time Systems, International Workshop. HART'97, Grenoble, France, March 26-28, 1997, Proceedings*. Edited by O. Maler. Volume 1201. Lecture Notes in Computer Science. Springer, 1997, pages 48–62. ISBN: 3-540-62600-X. DOI: 10.1007/BFb0014712.

[15] H. Kastenberg and A. Rensink. "Model Checking Dynamic States in GROOVE". In: *Proc. SPIN'06*. LNCS 3925. Springer, 2006, pages 299–305.

[16] C. Krause and H. Giese. "Probabilistic Graph Transformation Systems". In: *Proc. ICGT'12*. LNCS 7562. Springer, 2012, pages 311–325.

[17] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. CAV'11*. LNCS 6806. Springer, 2011, pages 585–591.

[18] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. "Automatic verification of real-time systems with discrete probability distributions". In: *Theor. Comput. Sci.* 282.1 (2002), pages 101–150. DOI: 10.1016/S0304-3975(01)00046-9.

[19] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. "Symbolic model checking for probabilistic timed automata". In: *Inf. Comput.* 205 (7 2007)., pages 1027–1077. ISSN: 0890-5401.

[20] K. Lan, C. Chou, and D. Jin. "The effect of 802.11a on DSRC for ETC communication". In: *2012 IEEE Wireless Communications and Networking Conference, WCNC 2012, Paris, France, April 1-4, 2012*. IEEE, 2012, pages 2483–2487. ISBN: 978-1-4673-0436-8. DOI: 10.1109/WCNC.2012.6214215.

[21] S. Neumann. "Modellierung und Verifikation zeitbehafteter Graphtransformationssysteme mittels GROOVE". Master's thesis. University of Paderborn, 2007.

[22] P. C. Ölveczky and J. Meseguer. "Semantics and Pragmatics of Real-Time Maude". In: *Higher-Order and Symbolic Computation* 20 (1 2007)., pages 161–196.

[23] *RailCab homepage*. http://www.railcab.de.

[24] W. Schäfer and H. Wehrheim. "The Challenges of Building Advanced Mechatronic Systems". In: *Proc. FOSE'07*. IEEE Computer Society, 2007, pages 72–84.

[25] R. Segala. "Modeling and verification of randomized distributed real-time systems". PhD thesis. Massachusetts Institute of Technology, 1996.

[26] K. Sen, M. Viswanathan, and G. A. Agha. "VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems". In: *Second International Conference on the Quantitative Evaluaiton of Systems (QEST 2005), 19-22 September 2005, Torino, Italy*. IEEE Computer Society, 2005, pages 251–252. ISBN: 0-7695-2427-3. DOI: 10.1109/QEST.2005.42.

ISBN 978-3-86956-405-0
ISSN 1613-5652