Probabilistic Metric Temporal Graph Logic

Sven Schneider, Maria Maximova, Holger Giese

Technische Berichte Nr. 140

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Universität Potsdam

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

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

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 140

Sven Schneider | Maria Maximova | Holger Giese

Probabilistic Metric Temporal Graph Logic

Universitätsverlag Potsdam

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet via http://dnb.dnb.de/.

Universitätsverlag Potsdam 2022
http://verlag.ub.uni-potsdam.de/

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

The series **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** is edited by the professors of the Hasso Plattner Institute for Digital Engineering at the University of Potsdam.

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

Layout: Tobias Pape
Print: docupoint GmbH Magdeburg

ISBN 978-3-86956-517-0

Also published online on the publication server of the University of Potsdam:
https://doi.org/10.25932/publishup-51506
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-515066

Cyber-physical systems often encompass complex concurrent behavior with timing constraints and probabilistic failures on demand. The analysis whether such systems with probabilistic timed behavior adhere to a given specification is essential. When the states of the system can be represented by graphs, the rule-based formalism of Probabilistic Timed Graph Transformation Systems (PTGTSs) can be used to suitably capture structure dynamics as well as probabilistic and timed behavior of the system. The model checking support for PTGTSs w.r.t. properties specified using Probabilistic Timed Computation Tree Logic (PTCTL) has been already presented. Moreover, for timed graph-based runtime monitoring, Metric Temporal Graph Logic (MTGL) has been developed for stating metric temporal properties on identified subgraphs and their structural changes over time.

In this paper, we (a) extend MTGL to the Probabilistic Metric Temporal Graph Logic (PMTGL) by allowing for the specification of probabilistic properties, (b) adapt our MTGL satisfaction checking approach to PTGTSs, and (c) combine the approaches for PTCTL model checking and MTGL satisfaction checking to obtain a Bounded Model Checking (BMC) approach for PMTGL. In our evaluation, we apply an implementation of our BMC approach in AutoGraph to a running example.

Contents

1 Introduction

Cyber-physical systems often encompass complex concurrent behavior with timing constraints and probabilistic failures on demand [16, 17]. Such behavior can then be captured in terms of probabilistic timed state sequences (or spaces) where time may elapse between successive states and where each step in such a sequence has a designated probability. The analysis whether such systems adhere to a given specification describing admissible or desired system behavior is essential in a model-driven development process.

Graph Transformation Systems (GTSs) [4] can be used for the modeling of systems when each system state can be represented by a graph and when the changes of such states can be captured by rule-based graph transformation. Moreover, timing constraints based on clocks, guards, invariants, and clock resets as in Probabilistic Timed Automata (PTA) [12] have been combined with graph transformation in Timed Graph Transformation Systems (TGTSs) [3] and probabilistic aspects have been added to graph transformation in Probabilistic Graph Transformation Systems (PGTSs) [10]. Finally, the formalism of PTGTSs [13] integrates both extensions and offers model checking support w.r.t. PTCTL [11, 12] properties employing the Prism model checker [11]. The usage of PTCTL allows for stating probabilistic real-time properties on the induced PTGT state space where each graph in the state space is labeled with a set of Atomic Propositions (APs) obtained by evaluating that graph w.r.t. e.g. some property specified using Graph Logic (GL) [6, 17].

However, structural changes over time in the state space cannot always be directly specified using APs that are *locally* evaluated for each graph. To express such structural changes over time, we introduced MTGL [5, 17] based on GL. Using MTGL conditions, an unbounded number of subgraphs can be tracked over timed graph transformation steps in a considered state sequence once bindings have been established for them via graph matching. Moreover, MTGL conditions allow to identify graphs where certain elements have just been added to (removed from) the current graph. Similarly to MTGL, for runtime monitoring, Metric First-Order Temporal Logic (MFOTL) [2] (with limited support by the tool Monpoly) and the non-metric timed logic Eagle [1, 7] (with full tool support) have been introduced operating, instead of graphs, on sets of relations and Java objects as state descriptions, respectively.

Obviously, both logics PTCTL and MTGL have distinguishing key strengths but also lack bindings on the part of PTCTL and an operator for expressing probabilistic requirements on the part of MTGL.[1] Furthermore, specifications using both, PTCTL

[1] PTCTL model checkers such as Prism do not support the branching capabilities of PTCTL as of now due to the complexity of the corresponding algorithms.

and MTGL conditions, are insufficient as they cannot capture phenomena based on probabilistic effects and the tracking of subgraphs at once. Hence, a more complex combination of both logics is required. Moreover, realistic systems often induce infinite or intractably large state spaces prohibiting the usage of standard model checking techniques. Bounded Model Checking (BMC) has been proposed in [8] for such cases implementing an on-the-fly analysis. Similarly, reachability analysis w.r.t. a bounded number of steps or a bounded duration have been discussed in [9].

To combine the strengths of PTCTL and MTGL, we introduce PMTGL by enriching MTGL with an operator for expressing probabilistic requirements as in PTCTL. Moreover, we present a BMC approach for PTGTSs w.r.t. PMTGL properties by combining the PTCTL model checking approach for PTGTSs from [13] (which is based on a translation of PTGTSs into PTA) with the satisfaction checking approach for MTGL from [5, 17]. In our approach, we just support *bounded* model checking since the binding capabilities of PMTGL conditions require non-local satisfaction checking taking possibly the entire history of a (finite) path into account as for MTGL conditions. However, we obtain even *full* model checking support for the case of finite loop-free state spaces and for the case where the given PMTGL condition does not need to be evaluated beyond a maximal time bound.

As a running example, we consider a system in which a sender decides to send messages at nondeterministically chosen time points, which have then to be transmitted to a receiver via a network of routers within a given time bound. For this scenario, we employ MTGL allowing to identify messages that have just been sent, to track them over time, and to check whether their individual deadlines are met.

This paper is structured as follows. In chapter 2, we recall the formalism of PTA. In chapter 3, we discuss further preliminaries including graph transformation, graph conditions, and the formalism of PTGTSs. In chapter 4, we recall MTGL and present the extension of MTGL to PMTGL in terms of syntax and semantics. In chapter 5, we present our BMC approach for PTGTSs w.r.t. PMTGL properties. In chapter 6, we evaluate our BMC approach by applying its implementation in the tool AutoGraph to our running example. Finally, in chapter 7, we close the paper with a conclusion and an outlook on future work.

2 Probabilistic Timed Automata

In this section, we introduce the syntax and semantics of PTA [12] and probabilistic timed reachability problems to be solved for PTA using Prism [11].

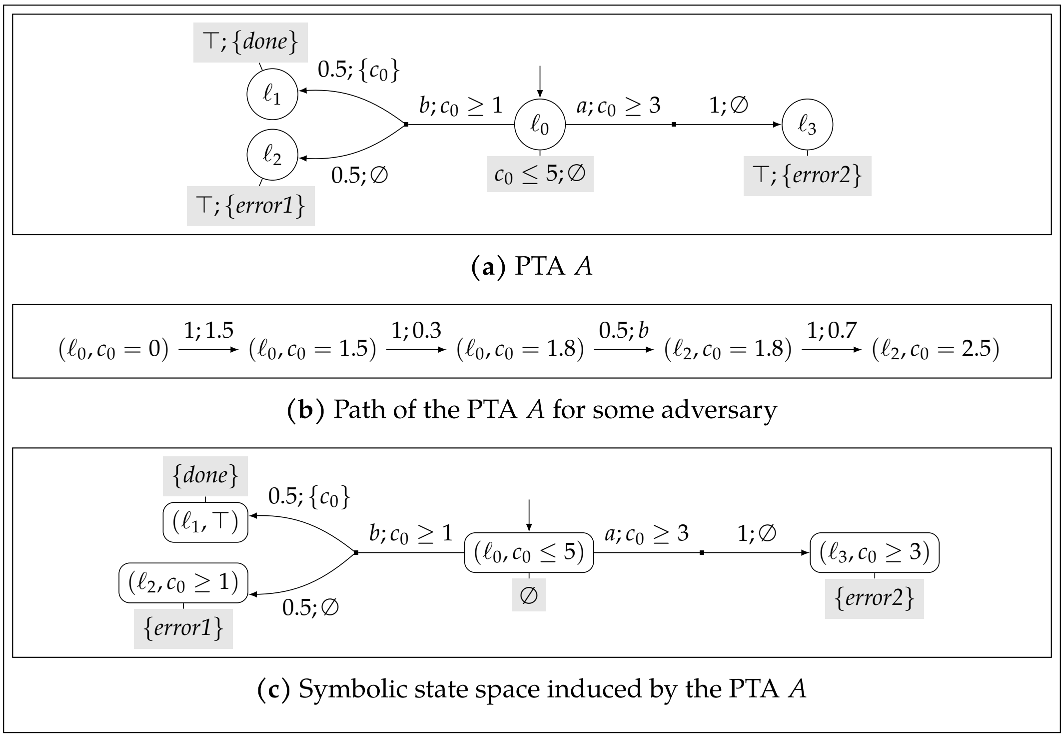
For a set of clock variables X, clock constraints $\psi \in \mathsf{CC}(X)$ are finite conjunctions of clock comparisons of the form $c_1 \sim n$ and $c_1 - c_2 \sim n$ where $c_1, c_2 \in X$, $\sim \in \{<, >, \leq, \geq\}$, and $n \in \mathbf{N} \cup \{\infty\}$. A clock valuation $v \in \mathsf{CV}(X)$ of type $v : X \longrightarrow \mathbf{R}_0^+$ satisfies a clock constraint ψ, written $v \models \psi$, as expected. The initial clock valuation $\mathsf{ICV}(X)$ maps all clocks to 0. For a clock valuation v and a set of clocks X', $v[X' := 0]$ is the clock valuation mapping the clocks from X' to 0 and all other clocks according to v. For a clock valuation v and a duration $\delta \in \mathbf{R}_0^+$, $v + \delta$ is the clock valuation mapping each clock x to $v(x) + \delta$.

For a countable set A, $\mu : A \longrightarrow [0, 1]$ is a Discrete Probability Distribution (DPD) over A, written $\mu \in \mathsf{DPD}(A)$, if the probabilities assigned to elements add up to 1, i.e., $\sum \{\!|\mu(a) \mid a \in A|\!\} = 1$ using summation over multisets. Moreover, the *support* of μ, written $\mathsf{supp}(\mu)$, contains all $a \in A$ for which the probability $\mu(a)$ is non-zero.

PTA combine the use of clocks to capture real-time phenomena and probabilism to approximate/describe the likelihood of outcomes of certain steps. A PTA (such as A from Figure 2.1a) consists of (a) a set of locations with a distinguished initial location (such as ℓ_0), (b) a set of clocks (such as c_0) which are initially set to 0, (c) an assignment of a set of APs (such as $\{done\}$) to each location (for subsequent analysis of e.g. reachability properties), (d) an assignment of constraints over clocks to each location as invariants such as $(c_0 \leq 5)$, and (e) a set of probabilistic timed edges. Each probabilistic timed edge consists thereby of (i) a single source location, (ii) at least one target location, (iii) an action (such as a or b), (iv) a clock constraint (such as $c_0 \geq 3$) specifying as a guard when the edge is enabled based on the current values of the clocks, and (v) a DPD assigning a probability to each pair consisting of a set of clocks to be reset (such as $\{c_0\}$) and a target location to be reached.

Definition 1 (PTA)**.** A *probabilistic timed automaton* (*PTA*) A is a tuple with the following components:

- $\mathsf{locs}(A)$ is a finite set of locations,
- $\mathsf{iloc}(A)$ is the unique initial location from $\mathsf{locs}(A)$,
- $\mathsf{acts}(A)$ is a finite set of actions disjoint from $\mathbf{R}_0^+$,
- $\mathsf{clocks}(A)$ is a finite set of clocks,
- $\mathsf{invs}(A) : \mathsf{locs}(A) \longrightarrow \mathsf{CC}(\mathsf{clocks}(A))$ maps each location to an invariant for that location such that the initial clock valuation satisfies the invariant of the initial location (i.e., $\mathsf{ICV}(\mathsf{clocks}(A)) \models \mathsf{invs}(A)(\mathsf{iloc}(A))$),
- $\mathsf{edges}(A) \subseteq \mathsf{locs}(A) \times \mathsf{acts}(A) \times \mathsf{CC}(\mathsf{clocks}(A)) \times \mathsf{DPD}(2^{\mathsf{clocks}(A)} \times \mathsf{locs}(A))$ is a finite set of PTA edges of the form (ℓ_1, a, ψ, μ) where ℓ_1 is the source location, a



(a) PTA A

(b) Path of the PTA A for some adversary

(c) Symbolic state space induced by the PTA A

Figure 2.1: PTA A, one of its paths, and its symbolic state space

is an action, ψ is a guard, and μ is a DPD mapping pairs (Res, ℓ_2) of clocks to be reset and target locations to probabilities,

- $\mathsf{aps}(A)$ is a finite set of APs, and
- $\mathsf{lab}(A) : \mathsf{locs}(A) \longrightarrow 2^{\mathsf{aps}(A)}$ maps each location to a set of APs.

The semantics of a PTA is given in terms of the induced Probabilistic Timed System (PTS). The states of the induced PTS are pairs of locations and clock valuations. The sequences of steps between such states define timed probabilistic paths. Each successive step in a path (such as the one in Figure 2.1b) is determined by an adversary which resolves the nondeterminism of the PTA by selecting either a duration by which all clocks are advanced in a timed step or a PTA edge that is used in a discrete step.

Definition 2 (PTS Induced by PTA)**.** Every PTA A induces a unique *probabilistic timed system (PTS)* $\mathsf{PTAtoPTS}(A) = P$ consisting of the following components:

- $\mathsf{states}(P) = \{(\ell, v) \in \mathsf{locs}(A) \times \mathsf{CV}(\mathsf{clocks}(A)) \mid v \models \mathsf{invs}(A)(\ell)\}$ contains as PTS states pairs of locations and clock valuations satisfying the location's invariant,
- $\mathsf{istate}(P) = (\mathsf{iloc}(A), \mathsf{ICV}(\mathsf{clocks}(A)))$ is the unique initial state from $\mathsf{states}(P)$,
- $\mathsf{acts}(P) = \mathsf{acts}(A)$ is the same set of actions,
- $\mathsf{steps}(P) \subseteq \mathsf{states}(P) \times (\mathsf{acts}(P) \cup \mathbf{R}_0^+) \times \mathsf{DPD}(\mathsf{states}(P))$ is the set of PTS steps.[1] A PTS step $((\ell, v), a, \mu) \in \mathsf{steps}(P)$ contains a source state (ℓ, v), an action from

[1] See [12] for a full definition of induced timed and discrete steps.

$\mathsf{acts}(P)$ for a discrete step or a duration from $\mathbf{R}_0^+$ for a timed step, and a DPD μ assigning a probability to each possible target state,

- $\mathsf{aps}(P) = \mathsf{aps}(A)$ is the same set of APs, and
- $\mathsf{lab}(P)(\ell, v) = \mathsf{lab}(A)(\ell)$ labels states in P according to the location labeling of A.

For model checking PTA [12], Prism does not compute the induced PTS according to Definition 2 instead it computes a *symbolic* state space (as in Figure 2.1c). In this symbolic state space, states are given by pairs of locations and clock constraints (called *zones*) where one state (ℓ, ψ) represents all pairs of states (ℓ, v) such that $v \models \psi$. To allow for such a symbolic state space representation, the syntax of clock constraints has been carefully chosen.

In chapter 5, we will use Prism to solve the following analysis problems defined for induced PTSs.

Definition 3 (Min/Max Probabilistic Timed Reachability Problems)**.** Evaluate the expression $\mathcal{P}_{op=?}(\mathsf{F}\ ap)$ for a PTS P with $op \in \{\mathsf{min}, \mathsf{max}\}$ and $ap \in \mathsf{aps}(P)$ to obtain the infimal/supremal probability (depending on op) over all adversaries to reach some state in P labeled with ap.

For example, for the PTS $P = \mathsf{PTAtoPTS}(A)$ induced by the PTA A from Figure 2.1a, (a) $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ done)$ is evaluated to probability 0.5 since a probability maximizing adversary would enable the discrete step using action b at time point 1 to reach ℓ_1 with probability 0.5 and (b) $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ done)$ is evaluated to probability 0 since a probability minimizing adversary would enable the discrete step using action a at time point 3 to reach ℓ_3 from which then no location labeled with *done* can be reached.

3 Probabilistic Timed Graph Transformation Systems

In this section, we briefly recall graphs, graph transformation, graph conditions, and the formalism of PTGTSs in our notation.

Using the variation of symbolic graphs [15] from [17], we consider typed attributed graphs (short graphs) (such as G_0 in Figure 3.1b), which are typed over a type graph TG (such as TG in Figure 3.1a). In such graphs, attributes are connected to *local variables* and an Attribute Condition (AC) over a many sorted first-order attribute logic is used to specify the values for these variables. Morphisms $m : G_1 \rightarrow G_2$ between graphs must ensure that the AC of G_2 is more restrictive compared to the AC of G_1 (w.r.t. the mapping of variables by m). Hence, the AC $\perp$ (false) in TG means that TG does not restrict attribute values. Lastly, we denote monomorphisms (short monos) by $m : G_1 \hookrightarrow G_2$.

Graph Conditions (GCs) [6, 17] of GL are used to state properties on graphs requiring the presence or absence of certain subgraphs in a host graph.

Definition 4 (GCs)**.** For a graph H, $\phi_H \in \mathsf{GC}(H)$ is a *graph condition* (*GC*) over H defined as follows:

$$\phi_H ::= \top \mid \neg\phi_H \mid \phi_H \wedge \phi_H \mid \exists(f, \phi_{H'}) \mid \nu(g, \phi_{H''})$$

where $f : H \hookrightarrow H'$ and $g : H'' \hookrightarrow H$ are monos and where additional operators such as $\perp$, $\vee$, and $\forall$ are derived as usual.

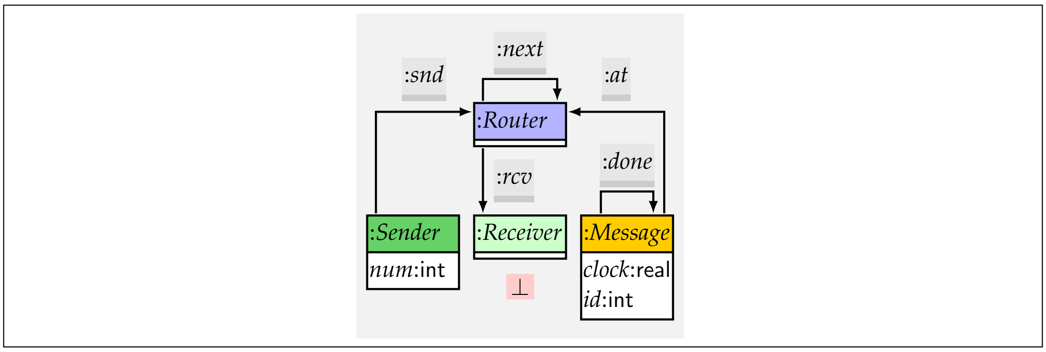
The satisfaction relation [6, 17] for GL defines when a mono satisfies a GC. Intuitively, for a graph H, the operator $\exists$ (called *exists*) is used to extend a current match of H to a supergraph H' and the operator ν (called *restrict*) is used to restrict a current match of H to a subgraph H''.

Definition 5 (Satisfaction of GCs)**.** A mono $m : H \hookrightarrow G$ *satisfies* a GC ϕ over H, written $m \models \phi$, if an item applies:

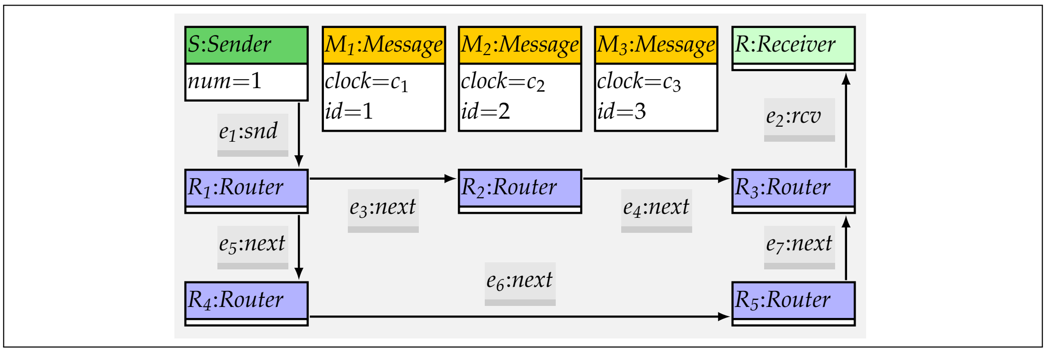
- $\phi = \top$.
- $\phi = \neg\phi'$ and $m \not\models \phi'$.
- $\phi = \phi_1 \wedge \phi_2$, $m \models \phi_1$, and $m \models \phi_2$.
- $\phi = \exists(f : H \hookrightarrow H', \phi')$ and $\exists m' : H' \hookrightarrow G.\ m' \circ f = m \wedge m' \models \phi'$.
- $\phi = \nu(g : H'' \hookrightarrow H, \phi')$ and $m \circ g \models \phi'$.

Moreover, if $\phi \in \mathsf{GC}(\emptyset)$ is a GC over the empty graph, $\mathrm{i}(G) : \emptyset \hookrightarrow G$ is an initial morphism, and $\mathrm{i}(G) \models \phi$, then the host graph G satisfies ϕ, written $G \models \phi$.

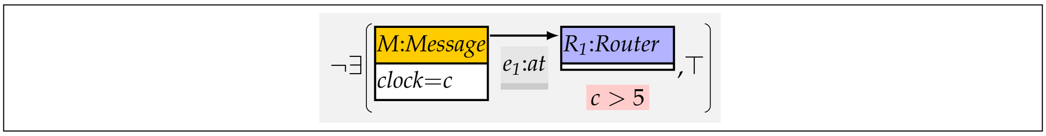
A Graph Transformation (GT) step is performed by applying a GT rule $\rho = (\ell : K \hookrightarrow L, r : K \hookrightarrow R, \gamma)$ for a match $m : L \hookrightarrow G$ on the graph to be transformed (see [17] for technical details). A GT rule specifies that (a) the graph elements in $L - \ell(K)$



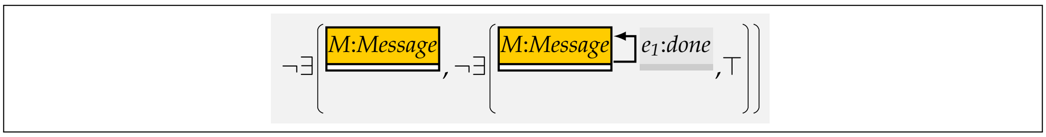
(a) Type graph TG



(b) Initial graph G_0

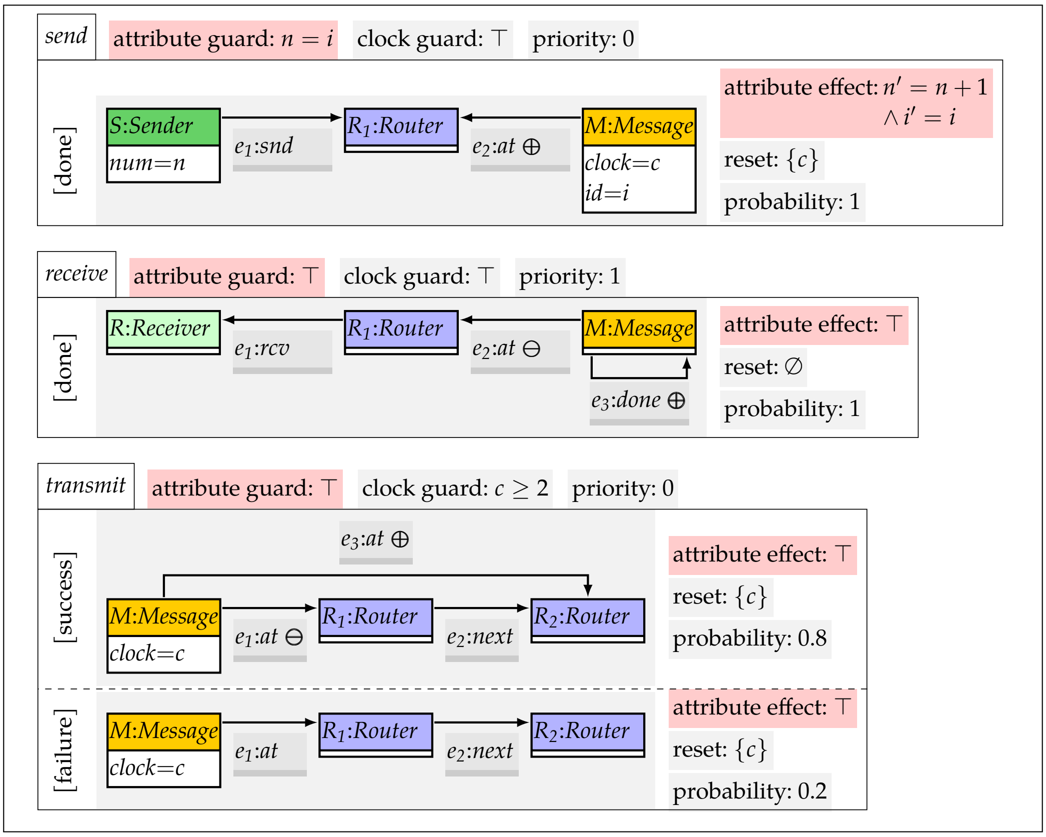


(c) PTGT invariant ϕ_{inv}

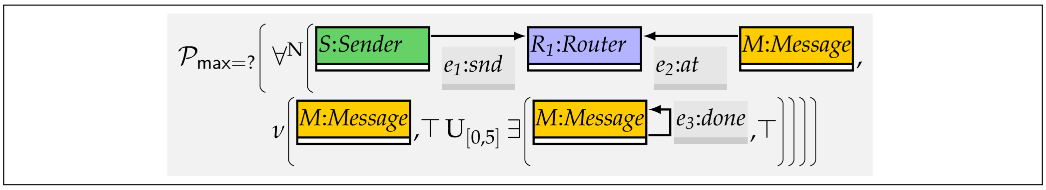


(d) PTGT AP ϕ_{fin}

Figure 3.1: Elements of the PTGTS and PMTGC χ_{max} for the running example (1/2)



(a) PTGT rules σ_{send}, $\sigma_{\mathsf{receive}}$, and $\sigma_{\mathsf{transmit}}$



(b) PMTGC χ_{max} where the additional MTGL operator *forall-new* (written $\forall^{\mathrm{N}}$) is derived from the operator *exists-new* by $\forall^{\mathrm{N}}(f, \theta') = \neg\exists^{\mathrm{N}}(f, \neg\theta')$

Figure 3.2: Elements of the PTGTS and PMTGC χ_{max} for the running example (2/2)

are to be deleted and the graph elements in $R - r(K)$ are to be added using the monos ℓ and r, respectively, according to a Double Pushout (DPO) diagram and (b) the values of variables of R are derived from those of L using the AC γ (e.g. $x' = x + 2$) in which the variables from L and R are used in unprimed and primed form, respectively. Nested application conditions given by GCs are straightforwardly supported by our approach but, to improve readability, not used in the running example and omitted subsequently.

PTGTSs introduced in [13] are a probabilistic real-time extension of Graph Transformation Systems (GTSs) [4]. We have shown in [13] that PTGTSs can be translated into equivalent PTA and, hence, PTGTSs can be understood as a high-level language for PTA.

Similarly to PTA, a PTGT state is given by a pair (G, v) of a graph and a clock valuation. The initial state is given by a distinguished initial graph and a valuation mapping all clocks to 0. For our running example, the initial graph (given in Figure 3.1b) captures a sender, which is connected via a network of routers to a receiver, and three messages to be sent. The type graph of a PTGTS also identifies attributes representing clocks.[1] For our running example, the type graph *TG* is given in Figure 3.1a where each *clock* attribute of a message represents such a clock. PTGT invariants are specified using GCs. Their evaluation for reachable graphs then results in clock constraints representing invariants as for PTA. For our running example, the PTGT invariant ϕ_{inv} from Figure 3.1c prevents that time elapses once a message was at one router for 5 time units. PTGT APs are also specified using GCs but a state (G, v) is labeled by such a PTGT AP if the evaluation of the GC for G results in a satisfiable clock constraint (i.e., the labeling of (G, v) is independent from v). For our running example, the AP ϕ_{fin} from Figure 3.1d labels states where *each* message has been successfully delivered to the receiver as indicated by the *done* loop.

PTGT rules of a PTGTS then correspond to edges of a PTA and contain (a) a left-hand side graph L, (b) an AC specifying as an *attribute guard* non-clock attributes of L, (c) an AC specifying as a *clock guard* clock attributes of L, (d) a natural number describing a *priority* where higher numbers denote higher priorities, and (e) a nonempty set of tuples of the form $(\ell : K \hookrightarrow L, r : K \hookrightarrow R, \gamma, C, p)$ where (ℓ, r, γ) is an underlying GT rule, C is a set of clocks contained in R to be reset, and p is a real-valued probability from $[0, 1]$ where the probabilities of all such tuples must add up to 1. See Figure 3.2a for the three PTGT rules σ_{send}, $\sigma_{\mathsf{receive}}$, and $\sigma_{\mathsf{transmit}}$ from our running example where the first two PTGT rules have each a unique underlying GT rule $\rho_{\mathsf{send,done}}$ and $\rho_{\mathsf{receive,done}}$, respectively, and where the last PTGT rule has two underlying GT rules $\rho_{\mathsf{transmit,success}}$ and $\rho_{\mathsf{transmit,failure}}$. For each of these underlying GT rules, we depict the graphs L, K, and R in a single graph where graph elements to be removed and to be added are annotated with $\ominus$ and $\oplus$, respectively. Further information about the PTGT rule (i.e., the attribute guard, clock guard, and priority) and each of its underlying GT rules (i.e., the *attribute effect* γ, set of clocks to be reset called *reset*, and *probability*) is given in red (for ACs) and gray boxes (for the rest).

[1] For a PTGT state (G, v), the values of clocks of G are stored in v and not in G.

The PTGT rule σ_{send} is used to push the next message into the network by connecting it to the router that is adjacent to the sender. Thereby, the attribute *num* of the sender is used to push the messages in the order of their *id* attributes. The PTGT rule $\sigma_{\mathsf{receive}}$ has the higher priority 1 and is used to pull a message from the router that is adjacent to the receiver by marking the message with a *done* loop. Lastly, the PTGT rule $\sigma_{\mathsf{transmit}}$ is used to transmit a message from one router to the next one. This transmission is successful with probability 0.8 and fails with probability 0.2. The clock guard of $\sigma_{\mathsf{transmit}}$ (together with the fact that the clock of the message is reset to 0 whenever $\sigma_{\mathsf{transmit}}$ is applied or when the message was pushed into the network using σ_{send}) ensures that transmission attempts may happen not faster than every 2 time units.

The semantics of a PTGTS is given by its induced PTS as in [13] using here concrete PTGT states instead of their equivalence classes for brevity.

Definition 6 (PTS Induced by PTGTS)**.** Every PTGTS S induces, using the operation $\mathsf{PTGTStoPTS}$, a unique *PTS* $\mathsf{PTGTStoPTS}(S) = P$ consisting of the following components:

- $\mathsf{states}(P)$ contains as PTS states pairs (G, v) where G is a graph and v is a valuation of the clocks of G satisfying the PTGT invariants of S,
- $\mathsf{istate}(P)$ is the unique initial state from $\mathsf{states}(P)$ consisting of the initial graph of S and the initial clock valuation of its clocks,
- $\mathsf{acts}(P)$ contains tuples of the form (σ, m, sp) consisting of the used PTGT rule σ, the used match m, and a mapping sp of each GT rule ρ in $\mathsf{rules}(\sigma)$ to the GT span $(k_1 : D \hookrightarrow G, k_2 : D \hookrightarrow H)$ constructed for a GT step from G to H using ρ,
- $\mathsf{steps}(P) \subseteq \mathsf{states}(P) \times (\mathsf{acts}(P) \cup \mathbf{R}_0^+) \times \mathsf{DPD}(\mathsf{states}(P))$ is the set of PTS steps.[2] A PTS step $((G, v), a, \mu) \in \mathsf{steps}(P)$ contains a source state (G, v), an action from $\mathsf{acts}(P)$ for a discrete step or a duration from $\mathbf{R}_0^+$ for a timed step, and a DPD μ assigning a probability to each possible target state,
- $\mathsf{aps}(P) = \mathsf{aps}(S)$ is the same set of PTGT APs, and
- $\mathsf{lab}(P)(G, v) = \{\phi \in \mathsf{aps}(S) \mid G \models \phi\}$ labels states in P with PTGT APs based only on the satisfaction of GCs for graphs.

[2] See [13] for a full definition of induced timed and discrete steps.

4 Probabilistic Metric Temporal Graph Logic

Before introducing PMTGL, we recall MTGL [5, 17] and adapt it to PTGTSs. To simplify our presentation, we focus on a restricted set of MTGL operators and conjecture that the presented adaptations of MTGL are compatible with full MTGL from [17] as well as with the orthogonal MTGL developments in [18].

The Metric Temporal Graph Conditions (MTGCs) of MTGL are specified using (a) the GC operators to express properties on a single graph in a path and (b) metric temporal operators to navigate through the path. For the latter, the operator $\exists^{\mathrm{N}}$ (called *exists-new*) is used to extend a current match of a graph H to a supergraph H' in the future such that some additionally matched graph element could not have been matched earlier. Moreover, the operator U (called *until*) is used to check whether an MTGC θ_2 is eventually satisfied in the future within a given time interval while another MTGC θ_1 is satisfied until then.

Definition 7 (MTGCs)**.** For a graph H, $\theta_H \in \mathsf{MTGC}(H)$ is a *metric temporal graph condition* (*MTGC*) over H defined as follows:

$$\theta_H ::= \top \mid \neg\theta_H \mid \theta_H \wedge \theta_H \mid \exists(f, \theta_{H'}) \mid \nu(g, \theta_{H''}) \mid \exists^{\mathrm{N}}(f, \theta_{H'}) \mid \theta_H \, \mathrm{U}_I \, \theta_H$$

where $f : H \hookrightarrow H'$ and $g : H'' \hookrightarrow H$ are monos and where I is an interval over $\mathbf{R}_0^+$.

For our running example, consider the MTGC given in Figure 3.2b inside the operator $\mathcal{P}_{\mathsf{max=?}}(\cdot)$. Intuitively, this MTGC states that (*forall-new*) whenever a message has just been sent from the sender to the first router, (*restrict*) when only tracking this message (since at least the edge e_2 can be assumed to be removed in between), (*until*) eventually within 5 time units, (*exists*) this message is delivered to the receiver as indicated by the *done* loop.

In [5, 17], MTGL was defined for timed graph sequences in which only discrete steps are allowed each having a duration $\delta > 0$. We now adapt MTGL to PTGTSs in which discrete steps and timed steps are interleaved and where zero time may elapse between two discrete steps.

To be able to track subgraphs in a PTS path π over time using matches, we first identify the graph $\pi(\tau)$ in π at a position $\tau = (t, s) \in \mathbf{R}_0^+ \times \mathbf{N}$ where t is a total time point and s is a step index.[1]

Definition 8 (Graph at Position)**.** A graph G is at position $\tau = (t, s)$ in a path π of PTS P, written $\pi(\tau) = G$, if $\mathsf{pos}(\pi, t, s, i) = G$ for some index i is defined as follows:

- If $\pi_0 = ((G, v), a, \mu, (G', v'))$, then $\mathsf{pos}(\pi, 0, 0, 0) = G$.

[1]To compare positions, we define $(t, s) < (t', s')$ if either $t < t'$ or $t = t'$ and $s < s'$.

- If $\pi_i = ((G,v), a, \mu, (G',v'))$, $\mathsf{pos}(\pi,t,s,i) = G$, and $a \in \mathbf{R}^+$, then $\mathsf{pos}(\pi, t+\delta, s, i) = G$ for each $\delta \in [0,a)$ and $\mathsf{pos}(\pi, t+a, 0, i+1) = G'$.
- If $\pi_i = ((G,v), a, \mu, (G',v'))$, $\mathsf{pos}(\pi,t,s,i) = G$, and $a \notin \mathbf{R}^+$, then $\mathsf{pos}(\pi, t, s+1, i+1) = G'$.

A given match $m : H \hookrightarrow \pi(\tau)$ into the graph $\pi(\tau)$ at position τ can be propagated forwards/backwards over the PTS steps in a path to the graph $\pi(\tau')$. Such a propagated match $m' : H \hookrightarrow \pi(\tau')$, written $m' \in \mathrm{PM}(\pi, m, \tau, \tau')$, can be obtained uniquely if *all* matched graph elements $m(H)$ are preserved by the considered PTS steps, which is trivially the case for timed steps. When some graph element is not preserved, $\mathrm{PM}(\pi, m, \tau, \tau')$ is empty.

We now present the semantics of MTGL by providing a satisfaction relation, which is defined as for GL for the operators inherited from GL and as explained above for the operators *exists-new* and *until*.

Definition 9 (Satisfaction of MTGCs)**.** An MTGC $\theta \in \mathsf{MTGC}(H)$ over a graph H is *satisfied* by a path π of the PTS P, a position $\tau \in \mathbf{R}_0^+ \times \mathbf{N}$, and a mono $m : H \hookrightarrow \pi(\tau)$, written $(\pi, \tau, m) \models \psi$, if an item applies:
- $\theta = \top$.
- $\theta = \neg\theta'$ and $(\pi, \tau, m) \not\models \theta'$.
- $\theta = \theta_1 \wedge \theta_2$, $(\pi, \tau, m) \models \theta_1$, and $(\pi, \tau, m) \models \theta_2$.
- $\theta = \exists(f : H \hookrightarrow H', \theta')$ and $\exists m' : H' \hookrightarrow \pi(\tau)$. $m' \circ f = m \wedge (\pi, \tau, m') \models \theta$.
- $\theta = \nu(g : H'' \hookrightarrow H, \theta')$ and $(\pi, \tau, m \circ g) \models \theta'$.
- $\theta = \exists^{\mathrm{N}}(f : H \hookrightarrow H', \theta')$ and there are $\tau' \geq \tau$, $m' \in \mathrm{PM}(\pi, m, \tau, \tau')$, and $m'' : H' \hookrightarrow \pi(\tau')$ s.t. $m'' \circ f = m'$, $(\pi, \tau', m'') \models \theta$, and for each $\tau'' < \tau'$ it holds that $\mathrm{PM}(\pi, m'', \tau', \tau'') = \emptyset$.
- $\theta = \theta_1 \mathrm{U}_I \theta_2$ and there is $\tau' \in I \times \mathbf{N}$ s.t.
  - there is $m' \in \mathrm{PM}(\pi, m, \tau, \tau')$ s.t. $(\pi, \tau', m') \models \theta_2$ and
  - for every $\tau \leq \tau'' < \tau'$ there is $m'' \in \mathrm{PM}(\pi, m, \tau, \tau'')$ s.t. $(\pi, \tau'', m'') \models \theta_1$.

Moreover, if $\theta \in \mathsf{MTGC}(\emptyset)$, $\tau = (0,0)$, and $(\pi, \tau, \mathrm{i}(\pi(\tau))) \models \theta$, then $\pi \models \theta$.

We now introduce the Probabilistic Metric Temporal Graph Conditions (PMTGCs) of PMTGL, which are defined based on MTGCs.

Definition 10 (PMTGCs)**.** Each *probabilistic metric temporal graph condition* (*PMTGC*) is of the form $\chi = \mathcal{P}_{\sim c}(\theta)$ where $\sim \in \{\leq, <, >, \geq\}$, $c \in [0,1]$ is a probability, and $\theta \in \mathsf{MTGC}(\emptyset)$ is an MTGC over the empty graph. Moreover, we also call expressions of the form $\mathcal{P}_{\mathsf{min}=?}(\theta)$ and $\mathcal{P}_{\mathsf{max}=?}(\theta)$ PMTGCs.

The satisfaction relation for PMTGL defines when a PTS satisfies a PMTGC.

Definition 11 (Satisfaction of PMTGCs)**.** A PTS P *satisfies* the PMTGC $\chi = \mathcal{P}_{\sim c}(\theta)$, written $P \models \chi$, if, for any adversary *Adv*, the probability over all paths of *Adv* that satisfy θ is $\sim c$. Moreover, $\mathcal{P}_{\mathsf{min}=?}(\theta)$ and $\mathcal{P}_{\mathsf{max}=?}(\theta)$ denote the infimal and supremal expected probabilities over all adversaries to satisfy θ (cf. Definition 3).

For our running example, the evaluation of the PMTGC χ_{max} from Figure 3.2b for the PTS induced by the PTGTS from Figure 3.1 and Figure 3.2 results in the probability of $0.8^6 = 0.262144$ using a probability maximizing adversary *Adv* as follows. Whenever the first graph of the PMTGC can be matched, this is the result of an application of the PTGT rule σ_{send}. The adversary *Adv* ensures then that each message is transmitted as fast as possible to the destination router R_3 by (a) letting time pass only when this is unavoidable to satisfy some guard and (b) never allowing to match the router R_4 by the PTGT rule $\sigma_{\mathsf{transmit}}$ as this leads to a transmission with 3 hops. For each message, the only transmission requiring at most 5 time units transmits the message via the router R_2 to router R_3 using 2 hops in $2 + 2$ time units. The urgently (i.e., without prior delay) applied PTGT rule $\sigma_{\mathsf{receive}}$ then attaches a *done* loop to the message as required by χ_{max}. Since the transmissions of the messages do not affect each other and messages are successfully transmitted only when both transmission attempts succeeded, the maximal probability to satisfy the inner MTGC is $(0.8 \times 0.8)^3$.

5 Bounded Model Checking Approach

We now present our approach for reducing the BMC problem for a fixed PTGTS S, a fixed PMTGC $\chi = \mathcal{P}_{\sim c}(\theta)$, and an optional time bound $T \in \mathbf{R}_0^+ \cup \{\infty\}$ to a model checking problem for a PTA and an analysis problem from Definition 3. Using this approach, we can analyze whether S satisfies χ when restricting the discrete behavior of S to the time interval $[0, T)$. In fact, we only consider PMTGCs of the form $\mathcal{P}_{\mathsf{min}=?}(\theta)$ or $\mathcal{P}_{\mathsf{max}=?}(\theta)$ for computing expected probabilities since they are sufficient to analyze the PMTGC $\mathcal{P}_{\sim c}(\theta)$.[1] See Table 5.1 for an overview of the subsequently discussed steps of our approach.

Step 1: Encoding the Time Bound into the PTGTS

For the given PTGTS S and time bound T, we construct an adapted PTGTS S' into which the time bound T is encoded (for $T = \infty$, we use $S' = S$). In S', we ensure that all discrete PTGT steps and all PTGT invariants are disabled when time bound T is reached. For this purpose, we (a) add an additional node b of a fresh node type *Bound* with a clock x to the initial graph of S and to the graphs L, K, and R of each underlying GT rule $\rho = (\ell : K \hookrightarrow L, r : K \hookrightarrow R, \gamma)$ of each PTGT rule σ of S, (b) add a PTGT rule with a priority higher than all other used priorities deleting the node b urgently with a guard $x = T$, and (c) extend each PTGT invariant ϕ to $\phi \vee \neg\exists(b{:}Bound, \top)$ disabling it for states where the b node has been removed. For the resulting PTGTS S', we then solve the model checking problem for the given PMTGC χ.

Step 2: Construction of an Equivalent PTA

For the PTGTS S' from step 1, we now construct an equivalent PTA A using the operation $\mathsf{PTGTStoPTA}$, which is based on a similar operation from [13].

As a first step, we obtain the underlying GTS (G_0, P) of S' where G_0 is the initial graph of S' and P contains all underlying GT rules ρ of all PTGT rules σ of S' as in [13]. As a second step, we construct for this GTS its GT state space (Q, E) consisting of states Q and edges E as in [13] but deviate by not identifying isomorphic states, which results in a tree-shaped GT state space with root G_0.[2] Note that the paths through (Q, E) symbolically describe all timed probabilistic paths through S'. As a third step, we again deviate from [13] and modify (Q, E) into (Q', E') by adding *time point clocks* throughout the paths of (Q, E) as follows: If N is the maximal number of graphs in any path π of (Q, E), we (a) create additional time point clocks tpc_1 to tpc_N, (b) add the i time point clocks tpc_1 to tpc_i to the ith graph in any path of the state space, and (c) add the clock tpc_i to the reset set of the step leading to the

[1] For example, $\mathcal{P}_{\mathsf{min}=?}(\theta) = c$ implies satisfaction of $\mathcal{P}_{\geq c'}(\theta)$ for any $c' \geq c$.

[2] Our BMC approach cannot be used if the PTGTS S' results in an infinite (Q, E).

Table 5.1: Overview of the steps of our BMC approach

Step	Inputs				Outputs
1	*PTGTS S*	*Time Bound T*			*PTGTS S′*
2	*PTGTS S′*				*PTA A*
3	*PTA A*				*GH-Map M_{GH}*
4	*PMTGC χ*				*GC ϕ*
5	*GC ϕ*	*GH-Map M_{GH}*			*AC-Map M_{AC}*
6	*PTA A*				*Zone-Map M_{Zone}*
7	*PMTGC χ*	*GH-Map M_{GH}*	*AC-Map M_{AC}*	*Zone-Map M_{Zone}*	*AP-Map M_{AP}*
8	*PTA A*	*AP-Map M_{AP}*			*Probability Interval I*

graph G_i in any path of the state space. Consequently, the AC $tpc_i - tpc_j$ for $j \geq i$ expresses the time expired between the graphs G_i and G_j. Finally, as in [13], we construct the resulting PTA A from the given PTGTS S' and the state space (Q', E') by (a) aggregating GT steps with a common source state and a match belonging to one PTGT rule, (b) annotating such aggregated GT steps with the clock-based timing constraints given by the guards and resets of the used PTGT rule, and (c) adding the clock-based timing constraints given by the PTGT invariants to the resulting PTA. This PTA construction ensures that the resulting PTA A is equivalent to the given PTGTS S'.

Lemma 1 (Soundness of PTA Construction)**.** If the given PTGTS S' has a finite tree-shaped state space (Q, E), then the two PTSs $\mathsf{PTAtoPTS}(\mathsf{PTGTStoPTA}(S'))$ and $\mathsf{PTGTStoPTS}(S')$ return the same results for the analysis problems from Definition 3. See appendix for a proof sketch.

In step 8, we will apply the Prism model checker [11] to the obtained PTA A and an analysis problem from Definition 3 corresponding to the given PMTGC χ. For this purpose, we obtain in steps 3–7 the set of leaf-locations of the PTA, in which the MTGC θ used inside the PMTGC χ is not violated, and then label precisely those locations from that set with an additional AP *success*. Employing this AP, the analysis problems from Definition 3 can be used to express the minimal/maximal probability to reach no violation.

Step 3: Folding of Paths into Graphs with History

For the given PTA A, we consider its structural paths π, which are the paths through the GT state space (Q', E') from which A was constructed. Such paths π may have timed realizations π' in which timed steps and discrete steps using the PTA edges of π are interleaved. Following the satisfaction checking approach for MTGL from [5, 17], we translate the MTGC satisfaction problem into an equivalent GC satisfaction problem using an operation fold (introduced subsequently) and an operation encode (introduced in step 4). Both operations together ensure for each timed realization π' of a structural path π of the given PTA A that $\pi' \models \theta$ iff $G'_H \models \phi$ when $\mathsf{fold}(\pi) = G_H$ is a Graph with History (GH), the graph G'_H is obtained from G_H by adding the durations of steps in π' as ACs over the time point variables contained in G_H, and $\mathsf{encode}(\theta) = \phi$.

The operation fold is applied to each structural path π of the given PTA A aggregating the information about the nature and timing of all GT steps into a single resulting GH. As a first step, we construct the colimit G_H for the diagram of the GT spans of π (given by the *sp* components of step actions according to Definition 6), which contains all graph elements that existed at some time point in π. As a second step, each node and edge in G_H is equipped with additional *creation/deletion time stamp attributes cts/dts* and *creation/deletion index attributes cidx/didx*. As a third step, the ACs $cts = tpc_0 - tpc_j$ and $cidx = j$ are added for each node/edge that appeared first in the graph G_j in the path π. As a fourth step, the ACs $dts = tpc_0 - tpc_j$ and $didx = j$ are added for each node/edge that is removed in the step reaching G_j in the path π. Finally, the ACs $dts = -1$ and $didx = -1$ are added for nodes/edges that are never removed in π.[3]

As output, we obtain the so-called *GH-restrictions GH-Map M_{GH}* mapping all leaf-locations ℓ of the PTA A to the GH constructed for the path ending in ℓ.

Step 4: Encoding of an MTGC as a GC

We now discuss the operation encode for translating the MTGC θ contained in the given PMTGC χ into a corresponding GC ϕ. Intuitively, this operation recursively encodes the requirements (see the items of Definition 9) expressed using MTGL operators on a timed realization π' (of a structural path π of the PTA A folded in step 3) using GL operators on the GH G_H (obtained by folding π) with additional ACs. In particular, quantification over positions $\tau = (t, s)$, as for the operators *exists-new* and *until*, is encoded by quantifying over additional variables x_t and x_s representing t and s, respectively. Moreover, matching of graphs, as for the operators *exists* and *exists-new*, is encoded by an additional AC alive. This AC requires that each matched graph element in the GH G_H has *cts*, *dts*, *cidx*, and *didx* attributes implying that this graph element exists for the position (x_t, x_s) in π'. Lastly, matching of new graph elements in the *exists-new* operator is encoded by an additional AC earliest. This AC requires, in addition to alive, that one of the matched graph elements has *cts* and *cidx* attributes equal to x_t and x_s, respectively.[3]

As output, we obtain the GC ϕ, which expresses the MTGC θ based on the graph G'_H obtained from the timed realization π' in step 3.

Step 5: Construction of AC-Restrictions of Violations

For each GH G_H (from the given *GH-Map M_{GH}*) obtained in step 3 for some path π, we evaluate the *negation* of the given GC ϕ obtained in step 4 for this G_H. The result of this evaluation is an AC γ, which describes valuations of the variables contained

[3]The presented operations fold and encode are adaptations of the corresponding operations from [5, 17] to the modified MTGL satisfaction relation defined for PTSs (see Definition 9). The adapted operation fold uses ACs to express clock differences instead of concrete assignments and employs additional index attributes *cidx/didx*. The adapted operation encode uses the additional *step index variable x_s* in the alive and earliest ACs to take not only the time stamp but also the step index into account.

in G_H. Each such valuation describes a timed realization π' of π not satisfying the MTGC θ (i.e., a violation) by providing real-valued time points for the additional time point clocks contained in G_H. In the sense of the equivalence discussed in step 3, such a valuation represents the durations of timed steps in π', which can be added in the form of an AC to G_H resulting in the graph G'_H such that $\pi' \not\models \theta$ and $G'_H \not\models \phi$.

For our running example, any path π ends with all messages being received. The obtained AC γ describes then that a violation has occurred when, for one of the messages, the sum of the timed steps between sending and receiving exceeds 5 time units. Certainly, due to possible interleavings of discrete steps and different routes from R_1 to R_3, there are various structural paths of A ending in different GHs each resulting in a different AC γ.

As output, we obtain the so-called *AC-restrictions AC-Map M_{AC}* mapping all leaf-locations ℓ of the PTA A to the AC γ constructed for the GH G_H (which is obtained for the path π ending in ℓ).

Step 6: Construction of Zone-Restrictions of Violations

We adapt the given PTA A from step 2 to a resulting PTA A' by adding an additional AP *terminal* and by labeling all leaf-locations with this AP. We then construct the symbolic zone-based state space for the PTA A' by evaluating $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ \mathit{terminal})$ (see Definition 3) using a minor adaptation of the Prism model checker that outputs the states $s = (\ell, \psi)$ labeled with the AP *terminal* containing a location ℓ and a clock constraint ψ as a zone (which is unique due to the tree-shaped form of the PTA A). For each structural path π of the PTA A ending in the location ℓ, the zone ψ symbolically represents all timed realizations π' of π, which respect the timing constraints of the PTA A, in terms of differences between the additional time point clocks added in step 2.

For our running example, the zone ψ obtained for some leaf-location then contains the clock constraints capturing for each message that (a) 2 to 5 time units elapsed before each transmission attempt and (b) no time elapsed between the arrival of that message at router R_3 and its reception by the receiver.

As output, we obtain the so-called *zone-restrictions Zone-Map M_{Zone}* mapping all leaf-locations ℓ of the PTA A to the zone ψ obtained for ℓ.

Step 7: Construction of Violations

We now combine the restrictions captured by the given mappings *GH-Map M_{GH}*, *AC-Map M_{AC}*, and *Zone-Map M_{Zone}* to determine the leaf-locations of the PTA A representing violations. A leaf-location ℓ represents a violation when it is reached by a structural path π of A that is realizable in terms of a timed realization π' such that the interleaving of timed and discrete steps in π' (which depends on the considered adversary) results in a violation when reaching ℓ. For this purpose, we compare the AC-restrictions with the zone-restrictions in a way that depends on whether the given PMTGC χ is of the form $\mathcal{P}_{\mathsf{max}=?}(\theta)$ or $\mathcal{P}_{\mathsf{min}=?}(\theta)$. In the following, we consider the case for max (and the case for min in brackets). We define the AC γ_{check} as $M_{Zone}(\ell) \wedge \neg(M_{AC}(\ell) \wedge M_{GH}(\ell).\mathrm{ac})$ (for min: $M_{Zone}(\ell) \wedge M_{AC}(\ell) \wedge M_{GH}(\ell).\mathrm{ac}$) where $M_{GH}(\ell).\mathrm{ac}$ denotes the AC of the GH $M_{GH}(\ell)$. This AC is satisfiable (for min: unsatisfiable) iff a violation is avoidable (for min: unreachable) for any probability maximizing (for min: probability minimizing) adversary based on interleavings of timed steps. We use the SMT solver Z3 [14] to decide whether the obtained AC γ_{check} is satisfiable (for min: unsatisfiable).

As output, we obtain the so-called *AP-Map* M_{AP}, which maps all leaf-locations ℓ of the PTA A to a set of APs. The set of APs $M_{AP}(\ell)$ contains (a) the APs *success* and *maybe*, if Z3 returns that the checked AC γ_{check} is satisfiable (for min: unsatisfiable) and (b) the AP *maybe*, if Z3 does not return a result. Hence, structural paths of the PTA A ending in locations labeled with the AP *success* represent PTGTS paths definitely (for min: possibly) satisfying the considered MTGC whereas PTGTS paths ending in locations labeled with the AP *maybe* may or may not represent such paths.

Step 8: Computation of Resulting Probabilities

In steps 1–7, we reduced the considered BMC problem to one of the analysis problems from Definition 3 for which Prism can be applied. For this last step, we adapt the given PTA A from step 2 to a PTA A' by adding the labeling captured by the given *AP-Map* M_{AP} from step 7. We compute and output the probability intervals $I = [\mathcal{P}_{\mathsf{min}=?}(\mathit{success}), \mathcal{P}_{\mathsf{min}=?}(\mathit{maybe})]$ and $I = [\mathcal{P}_{\mathsf{max}=?}(\mathit{success}), \mathcal{P}_{\mathsf{max}=?}(\mathit{maybe})]$ of possible expected probability values for $\mathcal{P}_{\mathsf{min}=?}(\theta)$ and $\mathcal{P}_{\mathsf{max}=?}(\theta)$, respectively. If Z3 always succeeded in step 7, this probability interval I will be a singleton. Lastly, we state that the presented BMC approach is sound (up to the imprecision possibly induced by Z3).

Theorem 1 (Soundness of BMC Approach)**.** The presented BMC approach correctly analyzes (correctly approximates) satisfaction of PMTGCs when the returned probability interval I is (is not) a singleton. See appendix for a proof sketch.

6 Evaluation

To evaluate our BMC approach, we applied its implementation in the tool AutoGraph (where Prism and Z3 are used as explained before) to our running example given by the PMTGC χ_{max} from Figure 3.2b and the PTGTS from Figure 3.1 and Figure 3.2. In this application, we used the time bound $T = \infty$ for which the PTGTS was not adapted in step 1 because it already resulted in a *finite* tree-shaped GT state space (Q, E) in step 2.[1] The constraint solver Z3 was always able to decide all satisfaction problems in step 7, and the probability interval obtained in step 8 using Prism was $[0.262144, 0.262144]$, which is in accordance with our detailed explanations below Definition 11.

We also applied our BMC approach to the same PTGTS (again using the time bound $T = \infty$) and the PMTGC $\mathcal{P}_{\mathsf{min}=?}(\theta)$ where θ is the MTGC used in the PMTGC χ_{max} from Figure 3.2b. In this case, we obtained the probability interval $[0, 0]$ in step 8 since there is a probability minimizing adversary that sends the first message at time point 0 and then delays the first two transmission attempts of that message to time points 5 and 10 ensuring that the message is not received within 5 time units as required in the MTGC θ.

Both discussed applications of our BMC approach (where steps 1–7 can be reused for the second application) required negligible runtime and memory.

[1] In (Q, E), each of the three messages has either not yet been sent, is at one of the five routers, or has been received resulting in at most 7^3 states.

7 Conclusion and Future Work

In this paper, we introduced the Probabilistic Metric Temporal Graph Logic (PMTGL) for the specification of cyber-physical systems with probabilistic timed behavior modeled as PTGTSs. PMTGL combines (a) MTGL with its binding capabilities for the specification of timed graph sequences and (b) the probabilistic operator from PTCTL to express best-case/worst-case probabilistic timed reachability properties. Moreover, we presented a novel Bounded Model Checking (BMC) approach for PTGTSs w.r.t. PMTGL properties.

In the future, we will consider the case study [13, 16] of a cyber-physical system where, in accordance with real-time constraints, autonomous shuttles exhibiting probabilistic failures on demand navigate on a track topology. For this case study, we will evaluate the expressiveness and usability of PMTGL as well as the performance of our BMC approach. Also, we will integrate our MTGL-based approach from [18] for deriving so-called optimistic violations.

Bibliography

[1] H. Barringer, A. Goldberg, K. Havelund, and K. Sen. “Rule-Based Runtime Verification”. In: *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings*. Edited by B. Steffen and G. Levi. Volume 2937. Lecture Notes in Computer Science. Springer, 2004, pages 44–57. DOI: 10.1007/978-3-540-24622-0_5.

[2] D. A. Basin, F. Klaedtke, S. Müller, and E. Zalinescu. “Monitoring Metric First-Order Temporal Properties”. In: *J. ACM* 62.2 (2015), 15:1–15:45. DOI: 10.1145/2699444.

[3] B. Becker and H. Giese. “On Safe Service-Oriented Real-Time Coordination for Autonomous Vehicles”. In: *11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008), 5-7 May 2008, Orlando, Florida, USA*. IEEE Computer Society, 2008, pages 203–210. ISBN: 978-0-7695-3132-8. DOI: 10.1109/ISORC.2008.13.

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

[5] H. Giese, M. Maximova, L. Sakizloglou, and S. Schneider. “Metric Temporal Graph Logic over Typed Attributed Graphs”. In: *Fundamental Approaches to Software Engineering - 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*. Edited by R. Hähnle and W. M. P. van der Aalst. Volume 11424. Lecture Notes in Computer Science. Springer, 2019, pages 282–298. ISBN: 978-3-030-16721-9. DOI: 10.1007/978-3-030-16722-6_16.

[6] A. Habel and K. Pennemann. “Correctness of high-level transformation systems relative to nested conditions”. In: *Mathematical Structures in Computer Science* 19.2 (2009), pages 245–296. DOI: 10.1017/S0960129508007202.

[7] K. Havelund. “Rule-based runtime verification revisited”. In: *Int. J. Softw. Tools Technol. Transf.* 17.2 (2015), pages 143–170. DOI: 10.1007/s10009-014-0309-2.

[8] N. Jansen, C. Dehnert, B. L. Kaminski, J. Katoen, and L. Westhofen. “Bounded Model Checking for Probabilistic Programs”. In: *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*. Edited by C. Artho, A. Legay, and D. Peled. Volume 9938. Lecture Notes in Computer Science. 2016, pages 68–85. DOI: 10.1007/978-3-319-46520-3_5.

[9] J. Katoen. “The Probabilistic Model Checking Landscape”. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. Edited by M. Grohe, E. Koskinen, and N. Shankar. ACM, 2016, pages 31–45. DOI: 10.1145/2933575.2934574.

[10] C. Krause and H. Giese. “Probabilistic Graph Transformation Systems”. In: *Graph Transformations - 6th International Conference, ICGT 2012, Bremen, Germany, September 24-29, 2012. Proceedings*. Edited by H. Ehrig, G. Engels, H. Kreowski, and G. Rozenberg. Volume 7562. Lecture Notes in Computer Science. Springer, 2012, pages 311–325. ISBN: 978-3-642-33653-9. DOI: 10.1007/978-3-642-33654-6_21.

[11] M. Z. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-Time Systems". In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Edited by G. Gopalakrishnan and S. Qadeer. Volume 6806. Lecture Notes in Computer Science. Springer, 2011, pages 585–591. ISBN: 978-3-642-22109-5. DOI: 10.1007/978-3-642-22110-1_47.

[12] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. "Symbolic Model Checking for Probabilistic Timed Automata". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*. Edited by Y. Lakhnech and S. Yovine. Volume 3253. Lecture Notes in Computer Science. Springer, 2004, pages 293–308. ISBN: 3-540-23167-6. DOI: 10.1007/978-3-540-30206-3_21.

[13] M. Maximova, H. Giese, and C. Krause. "Probabilistic timed graph transformation systems". In: *J. Log. Algebr. Meth. Program.* 101 (2018), pages 110–131. DOI: 10.1016/j.jlamp.2018.09.003.

[14] Microsoft Corporation. *Z3*. URL: https://github.com/Z3Prover/z3.

[15] F. Orejas. "Symbolic graphs for attributed graph constraints". In: *J. Symb. Comput.* 46.3 (2011), pages 294–315. DOI: 10.1016/j.jsc.2010.09.009.

[16] RailCab Team. *RailCab Project*. https://www.hni.uni-paderborn.de/cim/projekte/railcab.

[17] S. Schneider, M. Maximova, L. Sakizloglou, and H. Giese. "Formal testing of timed graph transformation systems using metric temporal graph logic". In: *Int. J. Softw. Tools Technol. Transf.* 23.3 (2021), pages 411–488. DOI: 10.1007/s10009-020-00585-w.

[18] S. Schneider, L. Sakizloglou, M. Maximova, and H. Giese. "Optimistic and Pessimistic On-the-fly Analysis for Metric Temporal Graph Logic". In: *Graph Transformation - 13th International Conference, ICGT 2020, Held as Part of STAF 2020, Bergen, Norway, June 25-26, 2020, Proceedings*. Edited by F. Gadducci and T. Kehrer. Volume 12150. Lecture Notes in Computer Science. Springer, 2020, pages 276–294. DOI: 10.1007/978-3-030-51372-6_16.

Glossary

AC Attribute Condition.

AP Atomic Proposition.

BMC Bounded Model Checking.

DPD Discrete Probability Distribution.

GC Graph Condition.

GH Graph with History.

GL Graph Logic.

GTS Graph Transformation System.

MFOTL Metric First-Order Temporal Logic.

MTGC Metric Temporal Graph Condition.

MTGL Metric Temporal Graph Logic.

PGTS Probabilistic Graph Transformation System.

PMTGC Probabilistic Metric Temporal Graph Condition.

PMTGL Probabilistic Metric Temporal Graph Logic.

PTA Probabilistic Timed Automaton.

PTCTL Probabilistic Timed Computation Tree Logic.

PTGTS Probabilistic Timed Graph Transformation System.

PTS Probabilistic Timed System.

TGTS Timed Graph Transformation System.

A Proofs

In this appendix, we provide proof sketches omitted in the main body of this paper.

Lemma 1, p. 22: Soundness of PTA Construction. The nonidentification of isomorphic states and the addition of time point clocks does not affect the possible steps of the resulting PTA. This PTA is therefore, following [13], equivalent to the given PTGTS S' w.r.t. the analysis problems from Definition 3. □

Theorem 1, p. 25: Soundness of BMC Approach. We conclude that the presented BMC approach computes the correct results (a) by encoding the time bound T properly in step 1, (b) from the soundness of the operation PTGTStoPTA according to Lemma 1 (following [13]), (c) from the soundness of the adapted translation of MTGC satisfaction problem into an equivalent GC satisfaction problem along the lines of [5, 17], and (d) from the correct computation of zones in Prism. □

B Details for Simplified Running Example

In this appendix, we present the steps of our BMC approach for a simplified form of our running example where only a single message is transmitted to the receiver.

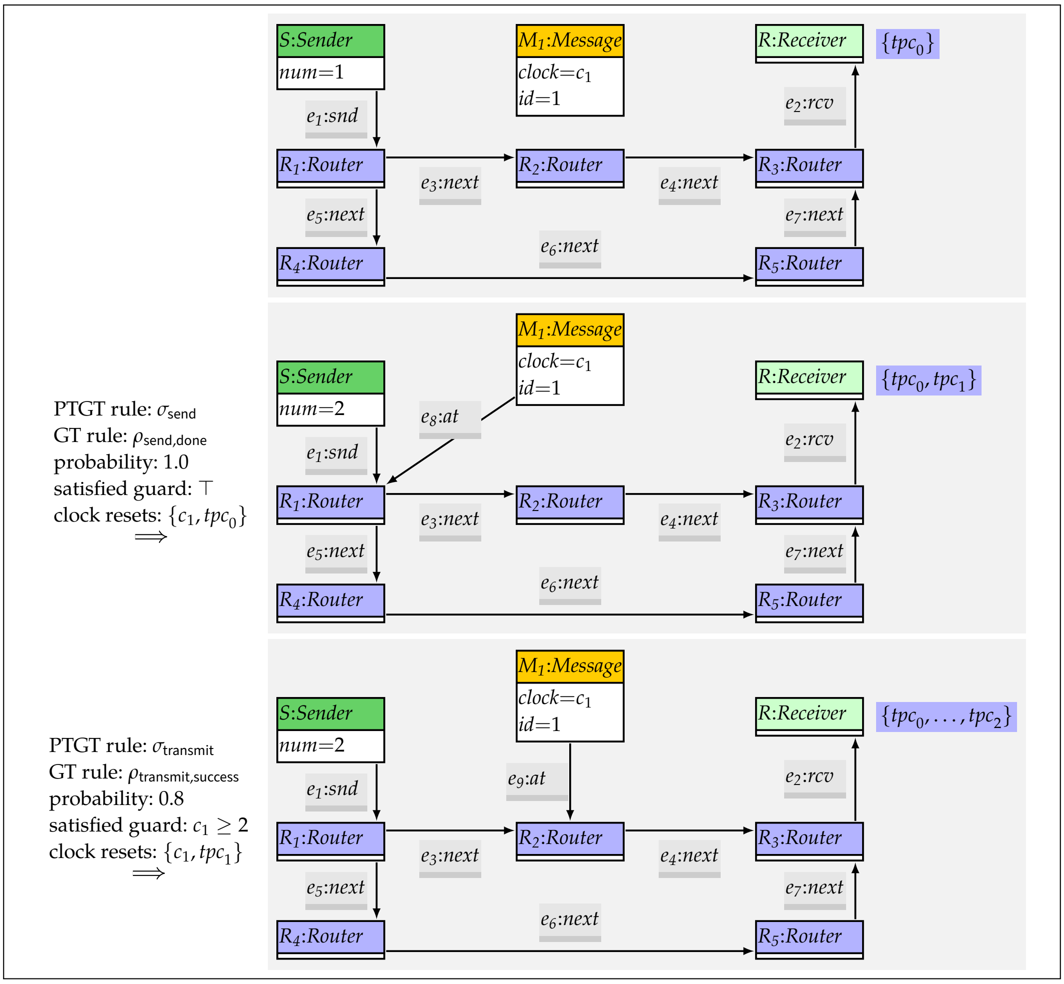


Figure B.1: Visualization for step 2 of our BMC approach: A structural path π of the PTA A (using an adapted initial graph with a single message) (1/2)

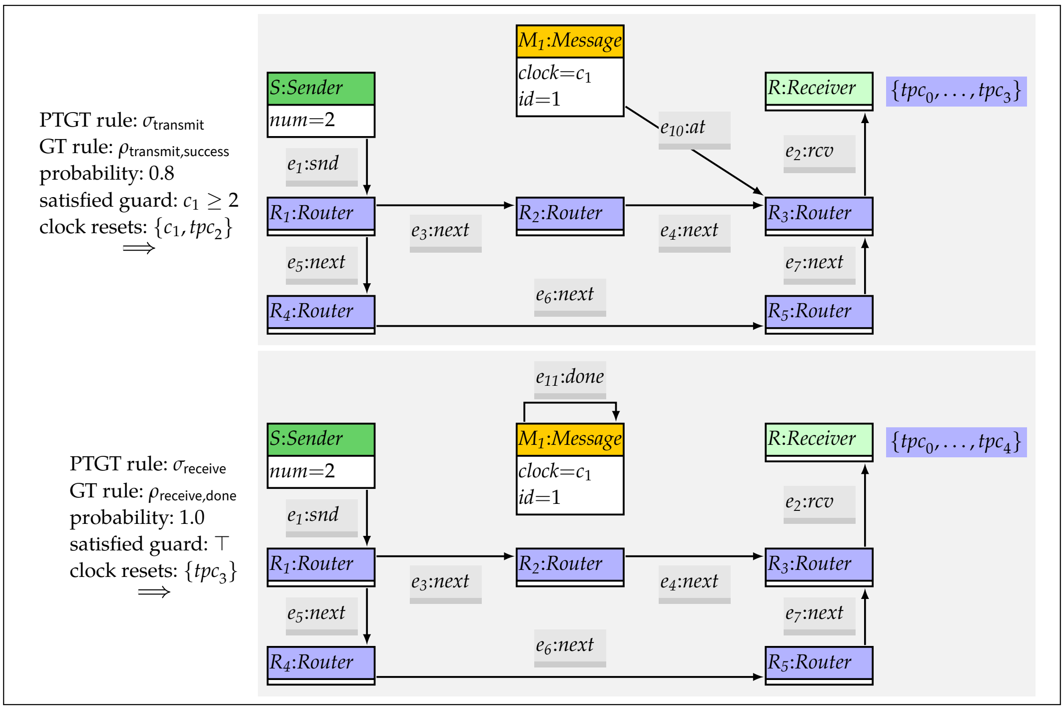


Figure B.2: Visualization for step 2 of our BMC approach: A structural path π of the PTA A (using an adapted initial graph with a single message) (2/2)

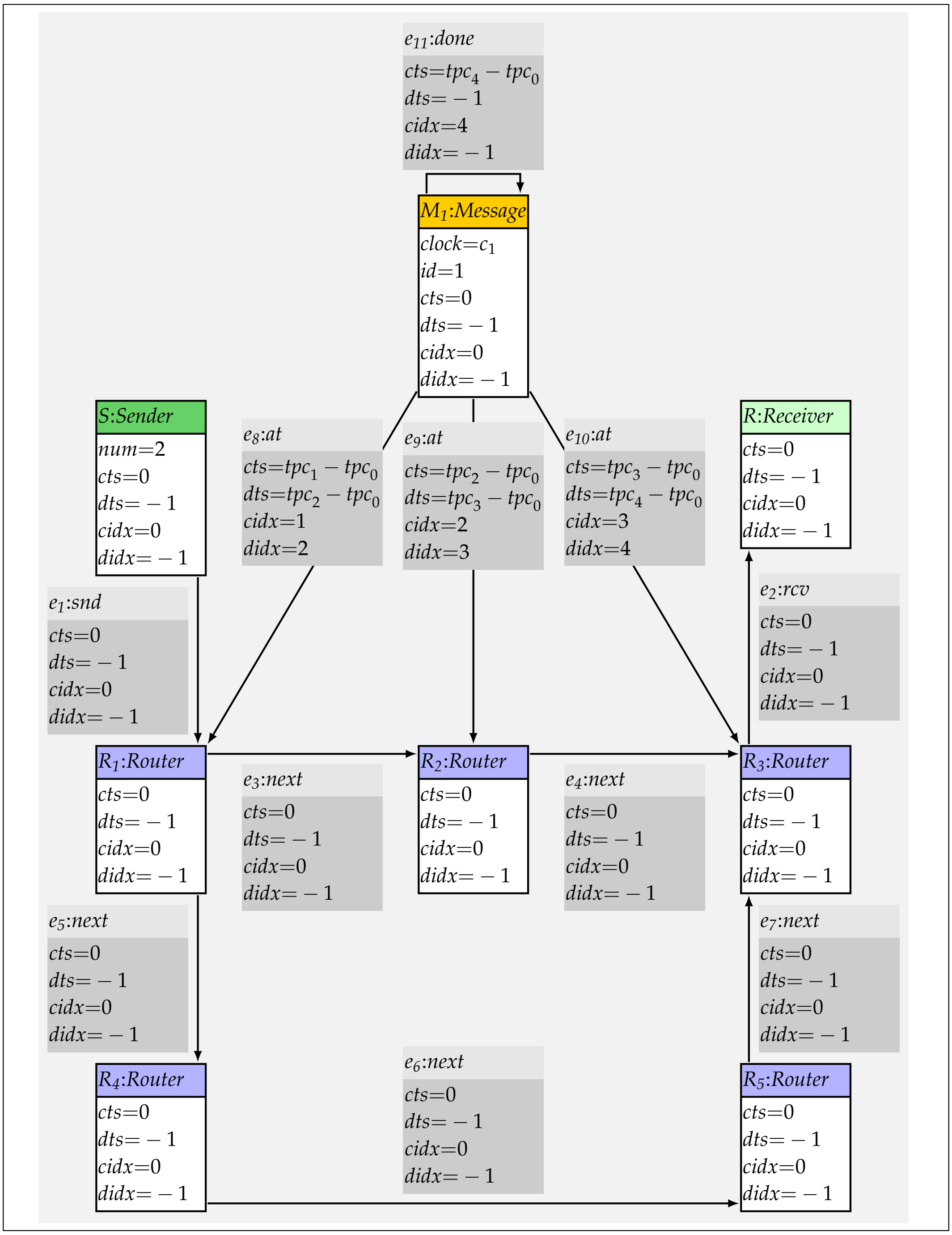
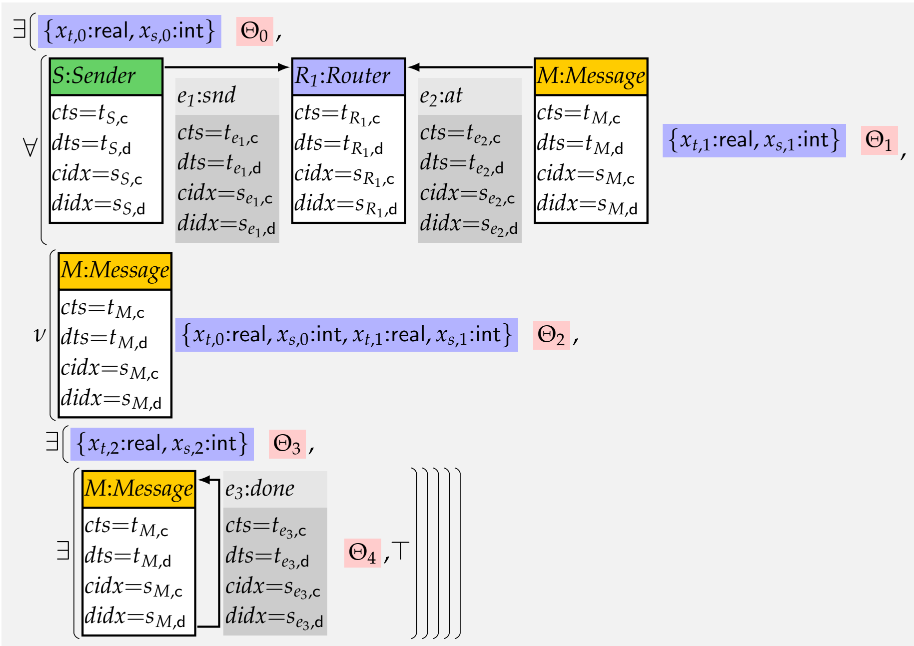


Figure B.3: Visualization for step 3 of our BMC approach: GH G_H obtained for the structural path π from Figure B.2



$$\Theta_0 = \{x_{t,0} = 0, x_{s,0} = 0\}$$
$$\Theta_1 = \Theta_0 \cup \{x_{t,0} < x_{t,1} \vee (x_{t,0} = x_{t,1} \wedge x_{s,0} < x_{s,1}), \text{alive}((x_{t,1}, x_{s,1}), \{S, R_1, M, e_1, e_2\}), \text{earliest}((x_{t,1}, x_{s,1}), \{S, R_1, M, e_1, e_2\})\}$$
$$\Theta_2 = \Theta_0 \cup \{\text{alive}((x_{t,1}, x_{s,1}), \{M\})\}$$
$$\Theta_3 = \Theta_2 \cup \{x_{t,1} < x_{t,2} \vee (x_{t,1} = x_{t,2} \wedge x_{s,1} < x_{s,2}), x_{t,2} \leq x_{t,1} + 5\}$$
$$\Theta_4 = \Theta_3 \cup \{\text{alive}((x_{t,2}, x_{s,2}), \{M, e_3\})\}$$

Figure B.4: Visualization for step 4 of our BMC approach: GC ϕ obtained by encoding of the MTGC from the PMTGC χ_{max}

$$
\begin{array}{l}
\neg\exists x_{t,0}\text{:real}, x_{s,0}\text{:int}.\\
\quad x_{t,0} = 0 \wedge x_{s,0} = 0\\
\quad \wedge \forall x_{t,1}\text{:real}, x_{s,1}\text{:int}.\\
\qquad x_{t,0} < x_{t,1} \vee (x_{t,0} = x_{t,1} \wedge x_{s,0} < x_{s,1})\\
\qquad \wedge\\
\qquad \text{alive}((x_{t,1}, x_{s,1}), \{S, R_1, M_1, e_1, e_8\})\\
\qquad \wedge\\
\qquad \text{earliest}((x_{t,1}, x_{s,1}), \{S, R_1, M_1, e_1, e_8\})\\
\qquad \rightarrow \exists x_{t,2}\text{:real}, x_{s,2}\text{:int}.\\
\qquad\quad x_{t,1} < x_{t,2} \vee (x_{t,1} = x_{t,2} \wedge x_{s,1} < x_{s,2})\\
\qquad\quad \wedge\\
\qquad\quad x_{t,2} \leq x_{t,1} + 5\\
\qquad\quad \wedge\\
\qquad\quad \text{alive}((x_{t,2}, x_{s,2}), \{M_1, e_{11}\})
\end{array}
$$

Intuitively, this expression captures an untimely reception in the sense of:

$$(tpc_4 - tpc_0) > (tpc_1 - tpc_0) + 5$$

or even in the simplest form:

$$tpc_4 > tpc_1 + 5$$

Technically, it refers to all attributes of the GH G_H (in the alive and earliest ACs), which makes the usage of G_H in step 7 necessary.

Figure B.5: Visualization for step 5 of our BMC approach: AC-restriction of violations (result of evaluating the negation of the GC ϕ from Figure B.4 for the GH G_H from Figure B.3)

$$\begin{aligned}&tpc_1 - tpc_0 \geq 0\\ &\wedge tpc_2 - tpc_1 \geq 2\\ &\wedge tpc_2 - tpc_1 \leq 5\\ &\wedge tpc_3 - tpc_2 \geq 2\\ &\wedge tpc_3 - tpc_2 \leq 5\\ &\wedge tpc_4 - tpc_3 \leq 0\\ &\wedge c_1 \geq 0\end{aligned}$$

Intuitively, the guards and invariants stated for the clock of the message result in a restriction of the time point clock variables.

Figure B.6: Visualization for step 6 of our BMC approach: Zone-restriction of violations (result for the structural path π from Figure B.2)

For the case of $\mathcal{P}_{\mathsf{max}=?}(\theta)$, we construct the AC γ_{check} using the AC from Figure B.5, the AC from Figure B.6, and the AC of the GH from Figure B.3 (given by the conjunction of all ACs contained in the graph). γ_{check} is equivalent to the following simplified AC.

$$\gamma_{check} \quad \equiv \quad 4 \leq tpc_4 - tpc_1 \leq 10 \wedge \neg(tpc_4 > tpc_1 + 5)$$

This AC γ_{check} is satisfiable. In fact, it is satisfied by the clock valuation $\{tpc_1 \mapsto 0, tpc_4 \mapsto 4\}$ describing the fastest transmission of the message M_1. From the satisfiability, we obtain the labeling of G_H from Figure B.3 using the APs *success* and *maybe*.

Figure B.7: Visualization for step 7 of our BMC approach: Derivation of labeling

The probability maximizing adversary will find at least the path to the location given by the GH G_H from Figure B.3. This path has a probability of $1 \times 0.8 \times 0.8 \times 1$ and is labeled with the APs *success* and *maybe*. Prism returns the probability interval $I = [0.64, 0.64]$ since all other paths will not be labeled with one of these APs because the timing constraint of at most 5 time units from the PMTGC χ_{max} is not satisfied by the other paths.

Figure B.8: Visualization for step 8 of our BMC approach: Derivation of probabilities

C Example for Step 7 of the BMC Approach

In this appendix, we provide a short example on why step 7 is defined as described. For this purpose, we consider different combinations of zone-restrictions and AC-restrictions for the two cases of $\mathcal{P}_{\mathsf{max}=?}(\theta)$ and $\mathcal{P}_{\mathsf{min}=?}(\theta)$.

Example 1 (Computation of Labeling in Step 7)**.** We consider a zone-restriction $4 \leq x \leq 10$ as well as AC-restrictions $x \geq 3$, $x \geq 5$, and $x \geq 12$. For the two cases from above, we then determine whether the corresponding leaf-location should be labeled with *success* and *maybe*.

max		
$(4 \leq x \leq 10) \wedge \neg(x \geq 3)$	is unsatisfiable,	hence no labeling
$(4 \leq x \leq 10) \wedge \neg(x \geq 5)$	is satisfiable,	hence labeling with $\{success, maybe\}$
$(4 \leq x \leq 10) \wedge \neg(x \geq 12)$	is satisfiable,	hence labeling with $\{success, maybe\}$
min		
$(4 \leq x \leq 10) \wedge (x \geq 3)$	is satisfiable,	hence no labeling
$(4 \leq x \leq 10) \wedge (x \geq 5)$	is satisfiable,	hence no labeling
$(4 \leq x \leq 10) \wedge (x \geq 12)$	is unsatisfiable,	hence labeling with $\{success, maybe\}$

For the case of $\mathcal{P}_{\mathsf{max}=?}(\theta)$, satisfiability means that some interleaving with timed steps does not result in a violation.
For the case of $\mathcal{P}_{\mathsf{min}=?}(\theta)$, unsatisfiability means that each interleaving with timed steps does not result in a violation.

Current Technical Reports from the Hasso-Plattner-Institut

Vol.	ISBN	Title	Authors/Editors
139	978-3-86956-514-9	**Deep learning for computer vision in the art domain : proceedings of the master seminar on practical introduction to deep learning for computer vision, HPI WS 20/21**	Christian Bartz, Ralf Krestel
138	978-3-86956-513-2	**Proceedings of the HPI Research School on service-oriented systems engineering 2020 Fall Retreat**	Christoph Meinel, Jürgen Roland Friedrich Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch, Tobias Friedrich, Erwin Böttinger, Christoph Lippert, Christian Dörr, Anja Lehmann, Bernhard Renard, Tilmann Rabl, Falk Uebernickel, Bert Arnrich, Katharina Hölzle
137	978-3-86956-505-7	**Language and tool support for 3D crochet patterns : virtual crochet with a graph structure**	Klara Seitz, Jens Lincke, Patrick Rein, Robert Hirschfeld
136	978-3-86956-504-0	**An individual-centered approach to visualize people's opinions and demographic information**	Wanda Baltzer, Theresa Hradilak, Lara Pfennigschmidt, Luc Maurice Prestin, Moritz Spranger, Simon Stadlinger, Leo Wendt, Jens Lincke, Patrick Rein, Luke Church, Robert Hirschfeld
135	978-3-86956-503-3	**Fast packrat parsing in a live programming environment : improving left-recursion in parsing expression grammars**	Friedrich Eichenroth, Patrick Rein, Robert Hirschfeld
134	978-3-86956-502-6	**Interval probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
133	978-3-86956-501-9	**Compositional analysis of probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese

ISBN 978-3-86956-517-0
ISSN 1613-5652