Probabilistic Metric Temporal Graph Logic

Sven Schneider, Maria Maximova, Holger Giese

Technische Berichte Nr. 146

des Hasso-Plattner-Instituts für
Digital Engineering an der 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 | 146

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 978-3-86956-532-3

Also published online on the publication server of the University of Potsdam:
https://doi.org/10.25932/publishup-54586
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-545867

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) [5] can be used for the modeling of systems when each system state can be represented by a graph and when all changes of such states to be modeled can be described using the rule-based approach to graph transformation. Moreover, timing constraints based on clocks, guards, invariants, and clock resets 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) [11]. Finally, the formalism of PTGTSs [14] combines timed and probabilistic aspects similar to Probabilistic Timed Automata (PTA) [13] and offers model checking support w.r.t. PTCTL [12, 13] properties employing the Prism model checker [12]. 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) [7, 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, MTGL [6, 17] has been introduced 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, 8] (with full tool support) have been introduced operating 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.

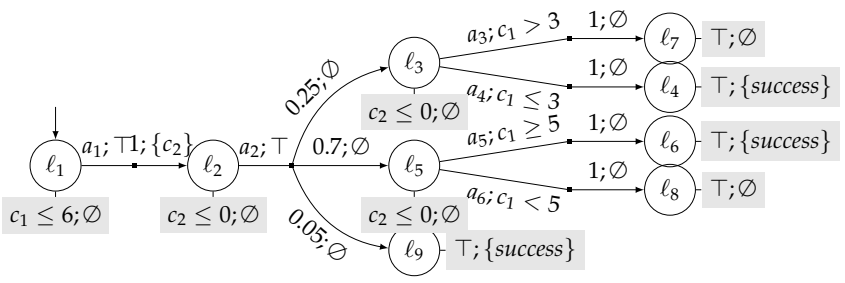
and MTGL conditions, are insufficient as they cannot capture phenomena based on probabilistic effects and the tracking of subgraphs at once. Hence, a more complex combination of both logics is required. Moreover, realistic systems often induce infinite or intractably large state spaces prohibiting the usage of standard model checking techniques. Bounded Model Checking (BMC) has been proposed in [9] for such cases implementing an on-the-fly analysis. Similarly, reachability analysis w.r.t. a bounded number of steps or a bounded duration has been discussed in [10].

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 [14] (which is based on a translation of PTGTSs into PTA) with the satisfaction checking approach for MTGL from [6, 17]. In our approach, we just support *bounded* model checking since the binding capabilities of PMTGL conditions require non-local satisfaction checks taking possibly the entire history of a (finite) path into account as for MTGL conditions. However, we obtain even *full* model checking support for two cases: (a) for the case of finite loop-free state spaces and (b) 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. In this system, transmission of messages is subject to a probabilistic failure on demand, requiring a retransmission of a message that was lost at an earlier transmission attempt. For this scenario, we employ PMTGL to express the desired system property of timely message reception. Firstly, using the capabilities inherited from MTGL, we identify messages that have just been sent, track them over time, and check whether their individual deadlines are met. Secondly, using the probabilistic operator inherited from PTCTL, we specify lower and upper bounds for the probability with which such an identified message is transmitted to the receiver before the deadline expires. During analysis, we are interested in determining the *expected best-case* and *worst-case* probabilities for a successful multi-hop message transmission from sender to receiver. For our evaluation, we also consider further variants of the considered scenario where messages are dropped after n transmission failures.

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.

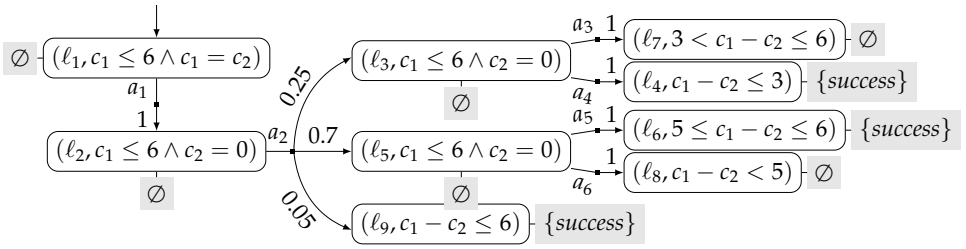
2 Probabilistic Timed Automata



(**a**) PTA A.

$$\left[\ell_1, \begin{bmatrix} c_1 = 0 \\ c_2 = 0 \end{bmatrix}\right] \xrightarrow{5;1} \left[\ell_1, \begin{bmatrix} c_1 = 5 \\ c_2 = 5 \end{bmatrix}\right] \xrightarrow{a_1;1} \left[\ell_2, \begin{bmatrix} c_1 = 5 \\ c_2 = 0 \end{bmatrix}\right] \xrightarrow{a_2;0.7} \left[\ell_5, \begin{bmatrix} c_1 = 5 \\ c_2 = 0 \end{bmatrix}\right] \xrightarrow{a_5;1} \left[\ell_6, \begin{bmatrix} c_1 = 5 \\ c_2 = 0 \end{bmatrix}\right]$$

(**b**) Path of the PTA A for some adversary (for clarity, for timed/discrete steps, labels contain here the duration/action and the probability to reach the given successor).



(**c**) Symbolic state space induced by the PTA A.

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

We briefly review PTA [13], which combine the use of clocks to capture real-time phenomena and probabilism to approximate/describe the likelihood of outcomes of certain steps, and PTA analysis as supported by Prism [12].

For a set of clocks X, clock constraints $\psi \in \mathsf{CC}(X)$ also called *zones* are finite conjunctions of clock comparisons $c_1 \sim n$ 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 $(v : X \rightarrow \mathbf{R}_0^+) \in \mathsf{CV}(X)$ satisfies a zone 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 $v(x) + \delta$. A Discrete Probability Distribution (DPD) $\mu : A \rightharpoonup [0,1]$, written $\mu \in \mathsf{DPD}(A)$, satisfies $\sum_{a \in A} \mu(a) = 1$. An element $a \in A$ is in the *support* of μ, written $a \in \mathsf{supp}(\mu)$, if $\mu(a) > 0$.

A PTA (see Figure 2.1a for an example) is of the form $A = (L, \overline{\ell} \in L, X, I : L \rightharpoonup \mathsf{CC}(X), \delta \subseteq L \times \mathcal{A} \times \mathsf{CC}(X) \times \mathsf{DPD}(2^X \times L), \mathcal{L} : L \rightharpoonup 2^{AP})$ where L is a set of locations, $\overline{\ell}$ is an initial location, X is a set of clocks, I maps each location to an invariant, δ contains edges $e = (\ell, a, \psi, \mu)$ where ℓ is the source location, $a \in \mathcal{A}$ is an action, ψ is a guard, and μ is a DPD where $\mu(X', \ell')$ is the probability to reach the target location ℓ' while resetting the clocks in X' to 0, and $\mathcal{L}$ labels each location with a set of atomic propositions from AP.

The states of a PTA are of the form $(\ell, v) \in L \times \mathsf{CV}(X)$ with $v \models I(\ell)$. The initial state is $(\overline{\ell}, \mathsf{ICV}(X))$. The labeling of a state (ℓ, v) is given by $\mathcal{L}(\ell)$. PTA allow for timed and discrete steps between states resulting in paths (such as the one in Figure 2.1b). A timed step $(\ell, v)[\delta, \overline{\mu}\rangle(\ell, v + \delta)$ of duration $\delta \in \mathbf{R}^+$ and DPD $\overline{\mu}$ must satisfy that $(\ell, v + \delta')$ is a state for every $0 < \delta' < \delta$ and $\overline{\mu}(\ell, v + \delta) = 1$. A discrete step $(\ell, v)[0, \overline{\mu}\rangle(\ell', v')$ of duration 0 and DPD $\overline{\mu}$ using some $(\ell, a, \psi, \mu) \in \delta$ and $(X', \ell') \in \mathsf{supp}(\mu)$ must satisfy $v \models \psi$, $v' = v[X' := 0]$, and $\overline{\mu}(\ell', v') = \sum_{X', v' = v[X' := 0]} \mu(X', \ell')$.

Prism supports PTA analysis, returning minimal probabilities $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ ap)$ and maximal probabilities $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ ap)$ with which an *ap* labeled state can be reached. These two probabilities may differ due to different resolutions of the nondeterminism among timed and discrete steps for which adversaries are employed as usual. For effective analysis, Prism does not compute the (usually infinite) induced state space but computes instead a finite symbolic state space (such as the one in Figure 2.1c) intuitively eliminating the impact of guards, invariants, and resets. In this finite symbolic state space, states are of the form $(\ell, \psi) \in L \times \mathsf{CC}(X)$ symbolically representing all states (ℓ, v) with $v \models \psi$.

For example, the PTA A from Figure 2.1a (for which adversaries only decide how much time to spend in location ℓ_1), $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ \mathit{success}) = 0.7 + 0.05$ using a probability maximizing adversary that lets $5 \leq \delta \leq 6$ time units elapse in ℓ_1 (0.25 is not added as ℓ_4 is not reachable using this adversary). Similarly, $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ \mathit{success}) = 0.05$ using a probability minimizing adversary that lets $3 < \delta < 5$ time units elapse in ℓ_1 (0.25 and 0.7 are not added as ℓ_4 and ℓ_6 are not reachable using this adversary).

3 Probabilistic Timed Graph Transformation Systems

We briefly recall graphs, graph conditions, and 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 local variables. Morphisms $m : G_1 \rightarrowtail G_2$ must ensure that the AC of G_2 (e.g. $y = 4$) implies the AC of G_1 (e.g. $m(x \geq 2) = (y \geq 2)$). Lastly, monomorphisms (short monos), denoted by $m : G_1 \hookrightarrow G_2$, map all elements injectively.

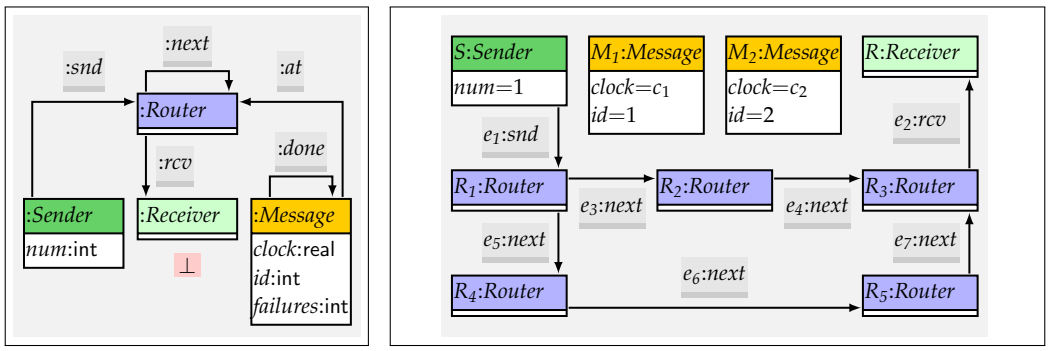
Graph Conditions (GCs) [7, 17] of GL are used to state properties on graphs requiring the presence or absence of certain subgraphs in a host graph using propositional connectives and (nested) existential quantification over graph patterns. For example, the GC ϕ_{allDone} from Figure 3.1c is satisfied by all graphs, in which all messages are equipped with a *done* loop.

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 $L - \ell(K)$ 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.[1]

PTGTSs introduced in [14] are a probabilistic real-time extension of Graph Transformation Systems (GTSs) [5]. PTGTSs can be translated into equivalent PTA according to [14], and hence, PTGTSs can be understood as a high-level language for PTA following similar mechanics.

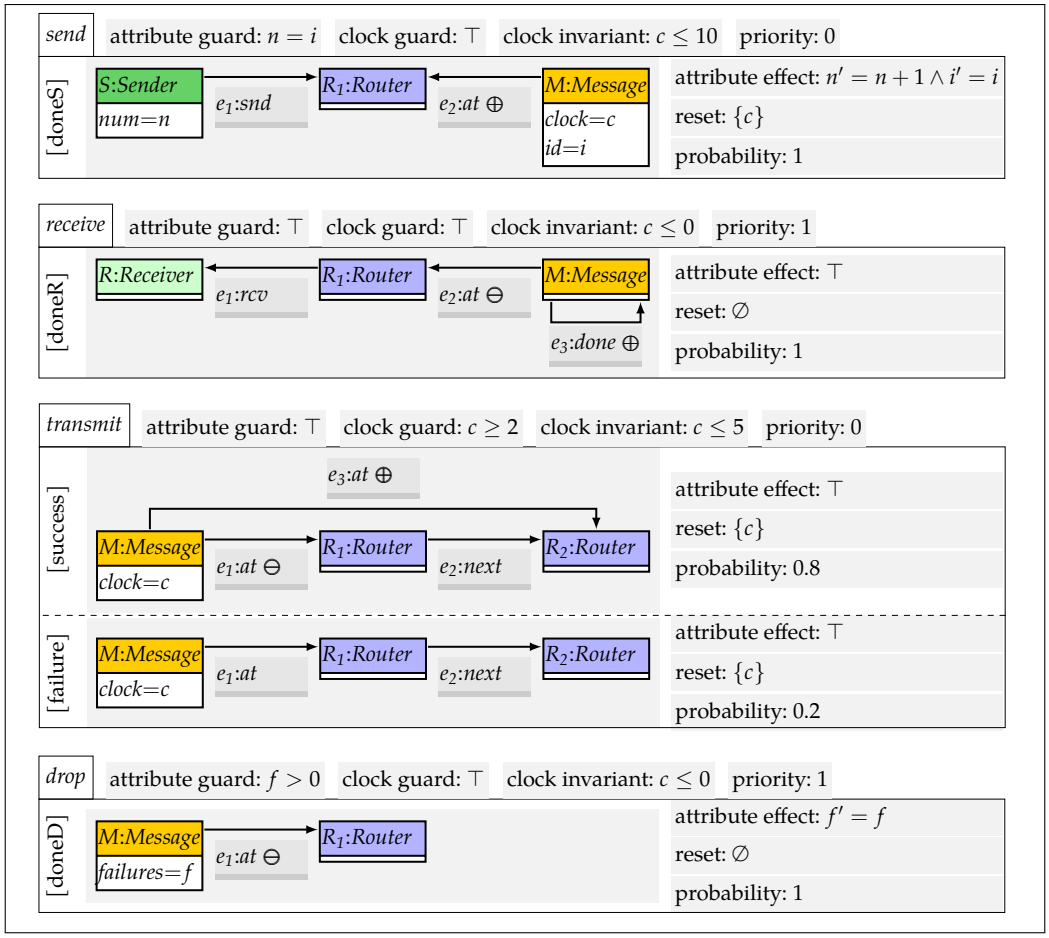
PTGT states are pairs (G, v) of a graph and a clock valuation. The *initial state* is given by a distinguished initial graph and a valuation mapping all clocks to 0. For our running example, the initial graph G_0 (given in Figure 3.1b) captures a sender, which is connected via a network of routers to a receiver, and two messages to be sent. The type graph of a PTGTS also identifies attributes representing clocks, which are the *clock* attributes of a message in Figure 3.1a.

[1] Nested application conditions given by GCs to further restrict rule applicability are straightforwardly supported by our approach but, to improve readability, not used in the running example and omitted subsequently.



(**a**) Type graph TG. (**b**) Initial graph G_0.

(**c**) PTGT AP ϕ_{allDone}, which is satisfied by graphs where all messages are equipped with a *done* loop indicating their successful delivery to the receiver.



(**d**) PTGT rules σ_{send}, σ_{receive}, σ_{transmit}, and σ_{drop}.

Figure 3.1: Components of the PTGTS for the running example

PTGT rules of a PTGTS contain (a) a left-hand side graph L, (b) an AC specifying as an *attribute guard* non-clock attributes of L that must be satisfied by any match of L, (c) an AC specifying as a *clock guard* clock attributes of L that must be satisfied to permit the application of the PTGT rule, (d) an AC specifying as a *clock invariant* clock attributes of L that must never be violated for a match of L, (e) a natural number describing a *priority* preventing the application of the PTGT rule when a PTGT rule with higher priority can be applied, and (f) a nonempty set of tuples of the form $(\ell : K \rightarrowtail L, r : K \rightarrowtail R, \gamma, C, p)$ 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.

For our running example, the PTGTS contains the four PTGT rules from Figure 3.1d. The PTGT rules σ_{send}, $\sigma_{\mathsf{receive}}$, and σ_{drop} have each a unique underlying GT rule $\rho_{\mathsf{send,doneS}}$, $\rho_{\mathsf{receive,doneR}}$, and $\rho_{\mathsf{drop,doneD}}$, respectively, and the PTGT rule $\sigma_{\mathsf{transmit}}$ has two alternative 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 and its underlying GT rules are given in gray boxes. 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. 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 and the clock invariant 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 happen within 2–5 time units. Lastly, the PTGT rule σ_{drop} has priority 1 and is used to drop messages for which transmission has failed. In our evaluation in chapter 6, we also consider the cases that messages are never dropped or not dropped before the second transmission failure by changing the attribute guard of σ_{drop} from $f > 0$ to $\bot$ and $f > 1$, respectively. PTGTS steps $(G, v)[\delta, \mu\rangle(G', v')$ are timed and discrete steps as for PTA.

PTGT APs are GCs ϕ and PTGT states (G, v) are labeled by ϕ when G satisfies ϕ. For our running example, the AP ϕ_{allDone} labels states where *each* message has been successfully delivered. Subsequently, we introduce PMTGL to identify relevant target states for analysis not relying on PTGT APs.

Besides translating a PTGTS into a PTA following [14], we can generate directly a symbolic state space (cf. Figure 2.1c for the PTA case) using the tool AutoGraph where each symbolic state (G, ψ) represents all states (G, v) with $v \models \psi$ and where ψ is encoded as a Difference Bound Matrix (DBM) [4].

4 Probabilistic Metric Temporal Graph Logic

Before introducing PMTGL, we recall MTGL [6, 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 within a given time interval in the future while another MTGC θ_1 is satisfied until then.

Definition 1 (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 4.1 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 by match restriction (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 [6, 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 multiple graphs may occur at the same time point.

For tracking subgraphs in a 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 starting at 0 after every non-zero timed step.[1]

Definition 2 (Graph at Position)**.** A graph G is at position $\tau = (t, s)$ in a path π of a PTGTS S, written $\pi(\tau) = G$, if $\mathsf{pos}(\pi, i, t, s, \delta) = G$ for the *ith* step of π and delay δ (since the last change of the step index s) is defined as follows.

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

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

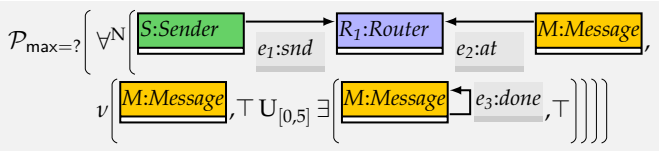


Figure 4.1: PMTGC $\chi_{\max}$ where the additional MTGL operator *forall-new* (written $\forall^{\mathrm{N}}$) is derived from the operator *exists-new* by $\forall^{\mathrm{N}}(f,\theta) = \neg\exists^{\mathrm{N}}(f,\neg\theta)$

- If $\pi_i = ((G,v)[\delta,\mu\rangle(G',v'))$, $\mathsf{pos}(\pi,i,t,s,0) = G$, and $\delta > 0$, then $\mathsf{pos}(\pi,i,t+\delta',0,\delta') = G$ for each $\delta' \in (0,\delta)$ and $\mathsf{pos}(\pi,i+1,t+\delta,0,0) = G'$
- If $\pi_i = ((G,v)[0,\mu\rangle(G',v'))$ and $\mathsf{pos}(\pi,i,t,s,\delta) = G$, then $\mathsf{pos}(\pi,i+1,t,s+1,0) = G'$

A match $m : H \hookrightarrow \pi(\tau)$ into the graph at position τ can be propagated forwards (or backwards) over the 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 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 3 (Satisfaction of MTGCs)**.** An MTGC $\theta \in \mathsf{MTGC}(H)$ over a graph H is *satisfied* by a path π of the PTGTS S, a position $\tau \in \mathbf{R}_0^+ \times \mathbf{N}$, and a mono $m : H \hookrightarrow \pi(\tau)$, written $(\pi,\tau,m) \models \theta$, 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 $\mathrm{PM}(\pi,m'',\tau',\tau'') = \emptyset$.
- $\theta = \theta_1 \,\mathrm{U}_I\, \theta_2$, $\tau = (t,s)$, and there are $\delta \in I$ and $\tau' = (t+\delta, s')$ s.t.
  - $s' \geq s$ if $\delta = 0$,
  - 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 4 (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 PTGTS satisfies a PMTGC.
Definition 5 (Satisfaction of PMTGCs)**.** A PTGTS S *satisfies* the PMTGC $\chi = \mathcal{P}_{\sim c}(\theta)$, written $S \models \chi$, 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.

For our running example, the evaluation of the PMTGC χ_{max} from Figure 4.1 for the PTGTS from Figure 3.1 results in the probability of $0.8^4 = 0.4096$, 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 the matched 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 the guard for the next transmission step 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 at least $2 + 2$ 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 once both transmission attempts for each of the messages have succeeded, the maximal probability to satisfy the inner MTGC is $(0.8 \times 0.8)^2 = 0.8^4$. Using $\mathcal{P}_{\mathsf{min}=?}(\cdot)$ results in a probability of 0 since there is e.g. the adversary Adv' that only allows a transmission with 3 hops via router R_4 exceeding the deadline.

5 Bounded Model Checking Approach

We now present our BMC approach in terms of an analysis algorithm for a fixed PTGTS S, PMTGC $\chi = \mathcal{P}_{\sim c}(\theta)$, and time bound $T \in \mathbf{R}_0^+ \cup \{\infty\}$. Using this algorithm, we analyze whether S satisfies χ when restricting the discrete behavior of S to the time interval $[0, T)$. In fact, we consider in this algorithm PMTGCs of the form $\mathcal{P}_{\mathsf{max}=?}(\theta)$ or $\mathcal{P}_{\mathsf{min}=?}(\theta)$ for computing expected probabilities since they are sufficient to analyze PMTGCs of the form $\mathcal{P}_{\sim c}(\theta)$.[1] In the subsequent presentation, we focus on the case of $\mathcal{P}_{\mathsf{max}=?}(\theta)$ and point out differences for the case of $\mathcal{P}_{\mathsf{min}=?}(\theta)$ where required.

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 $T = \infty$, to be used when all paths derivable for the PTGTS are sufficiently short, we use $S' = S$). In S', we ensure that all discrete PTGT steps are disabled when time bound T is reached (also then no longer requiring the satisfaction of the PTGT invariants). For this purpose, we (a) create a fresh local variable x_T of sort real and a fresh clock variable x_c (for which fresh types are added to the type graph to ensure non-ambiguous matching of variables during GT rule application), (b) add both variables and the attribute constraint $x_T = T$ to the initial graph of S, (c) add both variables to the graphs L, K, and R of each underlying GT rule $\rho = (\ell : K \rightarrowtail L, r : K \rightarrowtail R, \gamma)$ of each PTGT rule σ of S and add $x_c < x_T$ as an additional clock guard to each PTGT rule to prevent the application of PTGT rules beyond time bound T, and (d) add a PTGT rule σ_{BMC} with a clock guard $x_c \geq x_T$ and a clock invariant $x_c \leq x_T$, which (in its single underlying GT rule) deletes the variable x_T from the matched graph. The application of σ_{BMC} at time x_T ensures that no PTGT rule can be applied subsequently and that all PTGT invariants are disabled due to step (c).[2] For the resulting PTGTS S', we then solve the model checking problem for the given PMTGC χ.

Lemma 1 (Encoded BMC Bound)**.** If π is a path of the PTGTS S', then the time point of the last discrete step (if any exists) precedes T. See appendix for a proof sketch.

Step 2: Construction of Symbolic State Space and Timing Specification
Following the construction of a symbolic state space for a given PTA by the Prism model checker (where states are given by pairs of locations and zones over the clocks of the PTA (cf. chapter 2)), we may construct a symbolic state space for a

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

[2] The additional PTGT rule σ_{BMC} is used since PTGT invariants cannot be disabled by changing them from γ to $\gamma \vee x_c \geq T$ due to the limited syntax of zones.

given PTGTS where states are given by pairs of graphs and zones over the clocks contained in the graph. Paths $\hat{\pi}$ through such a symbolic state space are of the form $s_1[\mu_2\rangle s_2[\mu_3\rangle \dots s_n$ consisting of states and (nondeterministically selected) DPDs on successor states (i.e., $\mu_i(s_i) > 0$). Note again that each such path $\hat{\pi}$ is symbolic itself by not specifying the amount of time that elapses in each state. We call a path π of the form $s_1[\delta_2, \mu_2\rangle s_2[\delta_3, \mu_3\rangle \dots s_n$ a *timed realization* of $\hat{\pi}$ when the added delays $\delta_i \geq 0$ are a viable selection according to the zones contained in the states (e.g. for the symbolic state space in Figure 2.1c, the zone $c_1 = c_2 \leq 6$ of the initial state allows any selection $\delta_1 \leq 6$).

As a deviation from the symbolic state space generation approach for PTA, we generate a *tree-shaped* symbolic state space M by not identifying isomorphic states. The absence of loops in M guaranteed by the tree-shaped form ensures that, as required by Step 3, every path of M is finite (on time diverging paths). Moreover, for each path $\hat{\pi}$ of M, guards, invariants, and clock resets have been encoded in the zones of the states also ensuring the existence of at least one timed realization π for each $\hat{\pi}$. For our analysis algorithm, ultimately deriving the resulting probabilities in Step 5, we now use the guards, invariants, and clock resets again to derive for each path[3] $\hat{\pi}$ of M a *timing specification* $\mathsf{TS}(\hat{\pi})$. This timing specification captures for a path $\hat{\pi}$ when each of its states has been reached (which may be impossible without the tree-shaped form of the symbolic state space) thereby characterizing all viable timed realizations π of $\hat{\pi}$. To define $\mathsf{TS}(\hat{\pi})$, we use time point clocks tpc_i for $1 \leq i \leq n$ where n is the maximal length of any path of M. For a path $\hat{\pi}$, tpc_i then represents in $\mathsf{TS}(\hat{\pi})$ the time point when state i has been just reached in $\hat{\pi}$. Hence, $\mathsf{TS}(\hat{\pi})$ ranges over tpc_i for $1 \leq i \leq m$ where m is the length of $\hat{\pi}$. In the following, we also use the notion of the *total time valuation* $\mathsf{ttv}(\pi)$ to be the AC equating the time point clock tpc_i and the time point $\sum_{1 \leq k < i} \delta_i$ of the ith step in π. Using this notion, we characterize that π is a timed realization of $\hat{\pi}$ (performing the same discrete steps) when $\mathsf{TS}(\hat{\pi}) \wedge \mathsf{ttv}(\pi)$ is satisfiable.

To define $\mathsf{TS}(\hat{\pi})$, we use a map $\mathsf{LastReset}(k, c) = k'$ returning for an index $1 \leq k \leq m$ and a clock c the largest index $k' \leq k$ where c was reset in $\hat{\pi}$ (which can be easily computed by iterating once through $\hat{\pi}$). Recall that all clocks c are reset in the initial state, i.e., $\mathsf{LastReset}(1, c) = 1$. We include the ACs in $\mathsf{TS}(\hat{\pi})$ as follows for each state s_i. Firstly, when $i = 1$ (i.e., s_i is the initial state), we add $\mathsf{tpc}_1 = 0$ to $\mathsf{TS}(\hat{\pi})$. Secondly, when $i > 1$, we add $\mathsf{tpc}_{i-1} \leq \mathsf{tpc}_i$ to $\mathsf{TS}(\hat{\pi})$. Thirdly, when s_i was reached by respecting a guard ψ (implying $i > 1$), we add ψ to $\mathsf{TS}(\hat{\pi})$ after replacing each clock c contained in ψ by $\mathsf{tpc}_i - \mathsf{tpc}_{k'}$ where $k' = \mathsf{LastReset}(i-1, c)$.[4] Fourthly, when s_i was reached by respecting an invariant ψ', we add ψ' to $\mathsf{TS}(\hat{\pi})$ after replacing each clock c contained in ψ' by $\mathsf{tpc}_{i+1} - \mathsf{tpc}_{k'}$ where $k' = \mathsf{LastReset}(i, c)$.[5]

[3] We only consider paths starting in the initial state and ending in a leaf state.

[4] Intuitively, $\mathsf{tpc}_i - \mathsf{tpc}_{k'}$ is the duration between the last reset of c and the time point when the guard was checked upon state transition to s_i.

[5] Intuitively, $\mathsf{tpc}_{i+1} - \mathsf{tpc}_{k'}$ is the duration between the last reset of c and the time point at which the invariant was no longer checked due to the state transition to s_{i+1}.

Lemma 2 (Sound Timing Specification)**.** If $\hat{\pi}$ is a path of the symbolic state space M constructed for the PTGTS S', then there is a one-to-one correspondence between valuations of the time point clocks tpc_i satisfying $\mathsf{TS}(\hat{\pi})$ and the time points at which states are reached in the timed realizations π of $\hat{\pi}$. See appendix for a proof sketch.

For our running example (considering the restriction to a single message in the initial graph), for a path $\hat{\pi}_{\mathsf{ex}}$ where the message is sent to router R_1, transmitted to router R_2, transmitted to router R_3, and then received by receiver R, we derive (after simplification) $\mathsf{TS}(\hat{\pi}_{\mathsf{ex}})$ as the conjunction of $\mathsf{tpc}_1 = 0$, $0 \leq \mathsf{tpc}_2 \leq 10$, $\mathsf{tpc}_2 + 2 \leq \mathsf{tpc}_3 \leq \mathsf{tpc}_2 + 5$, and $\mathsf{tpc}_3 + 2 \leq \mathsf{tpc}_4 = \mathsf{tpc}_5 \leq \mathsf{tpc}_3 + 5$ essentially encoding the guards and invariants as expected.[6]

In the next two steps of our algorithm, we derive for the MTGC θ (contained in the given PMTGC $\mathcal{P}_{\mathsf{min}=?}(\theta)$ or $\mathcal{P}_{\mathsf{max}=?}(\theta)$) and a path $\hat{\pi}$ an AC describing timed realizations π of $\hat{\pi}$ satisfying θ. For our running example and the path $\hat{\pi}_{\mathsf{ex}}$ from above, this derived AC will be $\mathsf{tpc}_5 - \mathsf{tpc}_2 \leq 5$ expressing that the time elapsed between the sending of the message and its reception by the receiver is at most 5 time units as required by θ. Then, in Step 5 of the algorithm, we will identify (a) successful paths $\hat{\pi}$ to be those where $\mathsf{TS}(\hat{\pi})$ and the derived AC are satisfiable at once and (b) failing paths $\hat{\pi}$ to be those where $\mathsf{TS}(\hat{\pi})$ and the negated derived AC are satisfiable together.

Step 3: From MTGC Satisfaction to GC Satisfaction
Following the satisfaction checking approach for MTGL from [6, 17], we translate the MTGC satisfaction problem into an equivalent, yet much easier to check, GC satisfaction problem using the operations fold and encode (presented below). The operation fold aggregates the information about the nature and timing of all GT steps of $\hat{\pi}$ into a single *Graph with History* (*GH*). The operation encode translates the MTGC into a corresponding GC.[7] Technically, the MTGC θ is satisfied by a timed realization π of a path $\hat{\pi}$ of M precisely when the encoded MTGC is satisfied by the folded GH G_H once the total time valuation AC $\mathsf{ttv}(\pi)$ is added to G_H (incorporating the precise timing of steps in π).

Theorem 1 (Soundness of fold and encode)**.** If θ is an MTGC over the empty graph, $\mathsf{encode}(\theta) = \phi$, $\hat{\pi}$ is a path through the symbolic state space constructed for the PTGTS S', $\mathsf{fold}(\hat{\pi}) = G_H$, π is a timed realization of $\hat{\pi}$ (i.e., a path through S'), and G'_H is obtained from G_H by adding the AC $\mathsf{ttv}(\pi)$, then $\pi \models \theta$ iff $G'_H \models \phi$. See appendix for a proof sketch.

The operation fold generates for a path $\hat{\pi}$ the corresponding GH G_H by (a) constructing the union of all graphs of $\hat{\pi}$ where nodes/edges preserved in steps are identified and (b) recording for each node/edge in the resulting GH the position τ (cf. Definition 2) when it was created and deleted (if the node/edge is deleted at

[6] Note that $\mathsf{tpc}_4 = \mathsf{tpc}_5$ since the message reception by R takes no time.

[7] The operations fold and encode presented here are adaptations of the corresponding operations from [6, 17] to the modified MTGL satisfaction relation for PTGTSs from Definition 3 allowing for successive discrete steps with zero-time delay in-between.

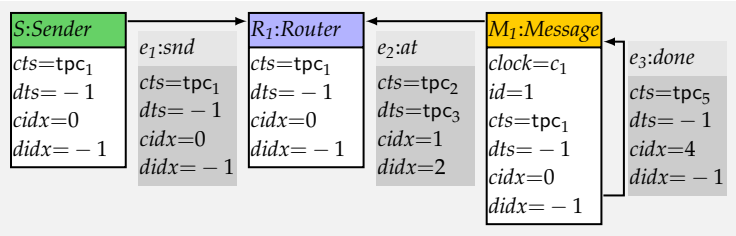


Figure 5.1: A part of the GH G_H obtained using the operation fold for the path $\hat{\pi}_{\mathsf{ex}}$ of the running example

some point) in $\hat{\pi}$ using additional *creation/deletion time stamp attributes cts/dts* and *creation/deletion index attributes cidx/didx*. In particular, (i) nodes/edges contained in the initial state of $\hat{\pi}$ are equipped with attributes $cts = \mathsf{tpc}_1$ and $cidx = 0$, (ii) nodes/edges added in step i of $\hat{\pi}$ are equipped with attributes $cts = \mathsf{tpc}_i$ and $cidx = i$, (iii) nodes/edges deleted in step i of $\hat{\pi}$ are equipped with attributes $dts = \mathsf{tpc}_i$ and $didx = i$, and (iv) nodes/edges contained in the last state of $\hat{\pi}$ are equipped with attributes $dts = -1$ and $didx = -1$. For the path $\hat{\pi}_{\mathsf{ex}}$ from our running example, see Figure 5.1 depicting the part of the GH G_H that is matched when checking the GC $\mathsf{encode}(\theta)$ against G_H.

The operation encode generates for the MTGC θ contained in the given PMTGC χ the corresponding GC ϕ (note that encode does not depend on a path and is therefore executed precisely once). Intuitively, it recursively encodes the requirements expressed using MTGL operators (see the items of Definition 3) on a timed realization π of a path $\hat{\pi}$ by using GL operators on the GH (obtained by folding $\hat{\pi}$) with additional integrated ACs. In particular, quantification over positions $\tau = (t, s)$ of global time t and step index 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. Also, matching of graphs, as for the operators *exists* and *exists-new*, is encoded by an additional AC alive. This AC requires that each matched node/edge in the GH 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 using the *exists-new* operator is encoded by an additional AC earliest. This AC requires that one of the matched graph elements has *cts* and *cidx* attributes equal to x_t and x_s, respectively.

Step 4: Construction of AC-Restrictions for Satisfaction
In this step, we obtain for each leaf state s of M a symbolic characterization in terms of an AC over the time point clocks tpc_i of all timed realizations π of the path $\hat{\pi}$ ending in s satisfying the given MTGC θ. Firstly, the timed realizations π of the path $\hat{\pi}$ ending in s are characterized by the timing specification $\mathsf{TS}(\hat{\pi})$ as discussed in Step 2. Secondly, we refine the set of such timed realizations using an AC $\gamma_{\hat{\pi}}$ over the time point clocks tpc_i symbolically describing when such a timed realization satisfies the given MTGC θ. The AC $\gamma_{\hat{\pi}}$ is obtained by checking the GC $\mathsf{encode}(\theta) = \phi$ against the GH $\mathsf{fold}(\hat{\pi}) = G_H$. The conjunction of $\mathsf{TS}(\hat{\pi})$ and $\gamma_{\hat{\pi}}$ is then recorded in the set of *state conditions* $\mathsf{SC}(s)$ and is satisfied by precisely those valuations of the time point

clocks tpc_i that correspond to timed realizations π ending in s satisfying the MTGC θ.[8] In Step 5, we also use the notion of *state probability* $\mathsf{SP}(s)$ assigning a probability of 1 to a state s when the AC in $\mathsf{SC}(s)$ is satisfiable and 0 otherwise.

Lemma 3 (Correct ACs)**.** If θ is an MTGC over the empty graph, $\hat{\pi}$ is a path of the symbolic state space constructed for the PTGTS S' ending in state s, and π is a timed realization of $\hat{\pi}$, then $\pi \models \theta$ iff $\mathsf{TS}(\hat{\pi}) \wedge \gamma_{\hat{\pi}} \wedge \mathsf{ttv}(\pi)$ is satisfiable. See appendix for a proof sketch.

For our running example, when checking the encoded MTGC (cf. Figure 4.1) for the GH partially given in Figure 5.1, (a) the graph elements S, R_1, M_1, e_1, and e_2 are matched for the *forall-new* operator and (b) the graph elements M_1 and e_3 are matched for the *exists* operator. For (a), all matched graph elements are alive at the *symbolic position* $(\mathsf{tpc}_2, 1)$ characterizing all positions $(t, 1)$ where $\mathsf{tpc}_2 = t$. The ACs in the encoded MTGC then ensure that e.g. e_1 is alive since it was created not after tpc_2 ($cts = \mathsf{tpc}_1 \leq \mathsf{tpc}_2$ and $cidx = 0 \leq 1$) and it has never been deleted ($dts = -1$) whereas e.g. e_2 is alive since it was created at $(\mathsf{tpc}_2, 1)$ and it has been deleted strictly later ($dts = \mathsf{tpc}_2$ but $1 < didx$). Moreover, the matched graph elements are not alive earlier since e_2 was created at $(\mathsf{tpc}_2, 1)$. For (b), all matched graph elements are alive at $(\mathsf{tpc}_5, 4)$. Overall, we obtain (after simplification) the AC requiring that $\mathsf{tpc}_5 - \mathsf{tpc}_2 \leq 5$ as the encoded MTGC expresses the time bound ≤ 5 used in the *until* operator. For the last state of the path $\hat{\pi}_{\mathsf{ex}}$, we obtain the AC $\mathsf{TS}(\hat{\pi}_{\mathsf{ex}}) \wedge \mathsf{tpc}_5 - \mathsf{tpc}_2 \leq 5$, which is e.g. satisfied by the valuation $\{\mathsf{tpc}_1 = 0, \mathsf{tpc}_2 = 0, \mathsf{tpc}_3 = 2, \mathsf{tpc}_4 = 4, \mathsf{tpc}_5 = 4\}$ representing a timed realization π_{ex} of $\hat{\pi}_{\mathsf{ex}}$ where the message is transmitted as early as possible in both transmission steps.

Step 5: Computation of Resulting Probabilities
In this step, we compute the maximal/minimal probability for the satisfaction of the given MTGC θ, i.e., for reaching states s with clock valuation v satisfying the AC contained in the state conditions $\mathsf{SC}(s)$. However, this kind of specification of target states is not supported by Prism, which requires a clock-independent specification of target states. Therefore, we propose a custom analysis procedure to solve the analysis problem from above.

In the following, we first discuss, on an example, an analysis procedure for the case of a clock-independent labeling of states and then expand this procedure to the additional use of state conditions $\mathsf{SC}(s)$. For the symbolic state space in Figure 2.1c, the maximal probability to reach a state labeled with *success* can be computed by propagating restrictions of valuations given by zones backwards. Initially, each state is equipped only with the zone given in the state space and the probability 1 when it is a target state. The zone/probability pairs $(c_1 - c_2 \leq 3, 1)$ and $(5 \leq c_1 - c_2 \leq 6, 1)$ of the ℓ_4-state and the ℓ_6-state are then propagated backwards without change to the ℓ_3-state and the ℓ_5-state, respectively. However, when steps have multiple target states, any subset of the target states is considered and the probabilities of pairs for the considered target states are summed up when the conjunction of their

[8] For the case of $\mathcal{P}_{\mathsf{min}=?}(\theta)$, we define $\mathsf{SC}(s) = \{\mathsf{TS}(\hat{\pi}) \wedge \neg\gamma_{\hat{\pi}}\}$.

zones is satisfiable. For example, we obtain $(5 \leq c_1 - c_2 \leq 6, 0.75)$ for the ℓ_2-state since the conjunction of the zones obtained for the ℓ_5- and ℓ_9-states is satisfiable, whereas the other subsets of target states result in unsatisfiable conjunctions or lower probabilities. When multiple zone/probability pairs with a common maximal probability are obtained, they are all retained for the source state of the step.

We now introduce our backward analysis procedure by adapting the procedure from above to the usage of the ACs contained in the state condition $\mathsf{SC}(s)$ instead of zones. Technically, our (fixed-point) backward analysis procedure updates the state conditions SC and state probability SP, which record the AC/probability pairs, until no further modifications can be performed according to the following definition.

Definition 6 (Backward Analysis Procedure)**.** The subsequent operation updating SC and SP is performed until a fixed-point is reached. When SC, SP, and I assign to each state s of M a set of ACs, a probability, and the depth of s in the tree-shaped state space M, respectively, (s, μ) is an edge of M, $S' \subseteq \mathsf{supp}(\mu)$ is a subset of the target states of μ, f selects for each target state $s' \in S'$ an AC from $\mathsf{SC}(s')$, $\gamma = \exists \mathsf{tpc}_{I(s)}.\ \bigwedge_{s' \in S'} f(s')$ is the AC derived for the state s based on the selections S' and f, γ is satisfiable, and $p = \sum_{s' \in S'} (\mu(s') \times \mathsf{SP}(s'))$ is the new probability for s based on the selections S' and f, then (a) $\mathsf{SC}(s)$ and $\mathsf{SP}(s)$ are changed to $\{\gamma\}$ and p when $p > \mathsf{SP}(s)$ recording the AC γ and the new maximal probability p derived for s and (b) $\mathsf{SC}(s)$ is changed to $\mathsf{SC}(s) \cup \{\gamma\}$ when $p = \mathsf{SP}(s)$ recording an additional AC γ and not changing the probability $\mathsf{SP}(s)$.

Finally, using our BMC approach introduced in this section, we derive the expected maximal probability.[9]

Theorem 2 (Soundness of BMC Approach)**.** The presented BMC approach in terms of the presented 5-step analysis algorithm returns the correct probability for a given PTGTS S, PMTGC χ, and time bound T. See appendix for a proof sketch.

[9] For $\mathcal{P}_{\mathsf{min}=?}(\theta)$, the procedure from Definition 6 returns $1 - \mathcal{P}_{\mathsf{min}=?}(\theta)$ maximizing the probability of failing paths by minimizing the probability for successful paths.

6 Evaluation

Our implementation of the presented BMC approach in the tool AutoGraph reports for all considered variations of our running example the expected best-case probability for timely message transmission of 0.8^{2n} (and the worst-case probability of 0) for n messages to be transmitted. For our experiments, we employed the time bound $T = 20$ corresponding to the maximum duration required for sending the message and transmitting it via the shortest connection. Note that an unbounded number of transmission retries for $T = \infty$ is unrealistic and would not allow for a finite state space M to be generated in Step 2. Also, any message transmission failure inevitably leads to a non-timely transmission of that message due to the time bound used in the PMTGC χ_{max}. However, the size of M is exponential in the number of messages to be transmitted as their transmission is independent from each other resulting in any resolution of their concurrent behavior to be contained in M. Hence, allowing for up to 10 transmission attempts via time bound $T = 20$ resulted in 31 states for $n = 1$ but exceeded our memory at 83000 states for $n = 2$. Using the drop rule to further limit the number of transmission retries allowed to analyze the variation of our running example in which two messages are transmitted but dropped after the second transmission failure, resulting in 12334 states.

However, as of now, the bottle neck of our current implementation, which is faithful to our presentation from the previous section, is not the runtime but the memory consumption. To overcome this limitation, we plan to generate the tree-shaped state space M in a depth-first manner performing the subsequent steps of the analysis algorithm (Step 3–Step 5) on entirely generated subtrees of M (before continuing with the state space generation). This would allow to dispose paths from M that are no longer needed in subsequent steps of the algorithm. Also, when the memory consumption has been drastically reduced along this line, a multithreaded implementation would be highly beneficial due to the tree-shaped form of M and the independent analysis for its subtrees.

7 Conclusion and Future Work

We introduced 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 BMC approach for PTGTSs w.r.t. PMTGL properties.

In the future, we plan to apply PMTGL and our BMC approach to the case study [14, 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. Moreover, we plan to extend our BMC approach by supporting the analysis of so-called optimistic violations introduced in [18].

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] J. Bengtsson and W. Yi. "Timed Automata: Semantics, Algorithms and Tools". In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets* [*This tutorial volume originates from the 4th Advanced Course on Petri Nets, ACPN 2003, held in Eichstätt, Germany in September 2003. In addition to lectures given at ACPN 2003, additional chapters have been commissioned*]. Edited by J. Desel, W. Reisig, and G. Rozenberg. Volume 3098. Lecture Notes in Computer Science. Springer, 2003, pages 87–124. DOI: 10.1007/978-3-540-27755-2_3.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[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] W. Schäfer and H. Wehrheim. "The Challenges of Building Advanced Mechatronic Systems". In: *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*. Edited by L. C. Briand and A. L. Wolf. IEEE Computer Society, 2007, pages 72–84. ISBN: 0-7695-2829-5. DOI: 10.1109/FOSE.2007.28.

[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.

DBM Difference Bound Matrix.

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.

TGTS Timed Graph Transformation System.

A Proofs

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

Proof for Lemma 1, p. 18: Encoded BMC Bound. No PTGT rule of the PTGTS S' can be applied at time $> T$ since at time T the additional PTGT rule σ_{BMC} is applied deleting the variable x_T, which is contained in every left-hand side graph of every rule of the PTGTS S'. This PTGT rule is applied because every other PTGT rule of the PTGTS S' has the additional guard $x_c < x_T$ forbidding rule application. Hence, only σ_{BMC} may be applicable at time T. The PTGT rule σ_{BMC} is applicable because (a) the variables x_c and x_T could not have been matched and deleted before by any other PTGT rule of the PTGTS S' since we added fresh types for both variables to the type graph, (b) the guard $x_c \geq x_T$ of σ_{BMC} is satisfied at time T, and (c) the PTGT invariant $x_c \leq x_T$ of σ_{BMC} does not exclude reaching time T with the rule being applicable. □

Proof for Lemma 2, p. 19: Sound Timing Specification. For a given path $\hat{\pi}$ of the symbolic state space, collecting the guards, invariants, and resets as explained, gathers exactly the same information that is inserted into the DBM representation for the zones contained in the states of the path $\hat{\pi}$. By induction on the length of the path, we observe that the restriction of tpc_i computed for an additional step of the path $\hat{\pi}$ precisely encodes the time when the target state of that step is reached. Note that $\mathsf{tpc}_1 = 0$ is added as well ensuring that this first variable is restricted to the correct global time when the path starts. By then defining each successive variable tpc_{i+1} based on the previous variable tpc_i, we only need to ensure that the successive variable is properly restricted to the duration that could have elapsed in state i, which is ensured by including precisely the constraints given by the guards (before the reset) and invariants (after the previous reset) for each step. □

Proof for Theorem 1, p. 20: Soundness of fold *and* encode . Here, we mostly follow the proofs for the corresponding operations from [6, 17] for the case of timed graph sequences where a non-zero amount of time elapsed between two states. For the adaptation of MTGL to the clock-based setting of PTGTSs in this paper, we then only need to ensure that the extended notion of positions (from pure time points to pairs of a time point and a step index) is properly reflected in the two operations.

For the operation fold, we note that adding the correct global time points for *cts* and *dts* variables in terms of the tpc_i variables instead of the explicitly computed accumulated global time is even a simplification. Also, for the operation fold, adding the step index for each step to the *cidx* and *didx* variables is a trivial extension (where we do not reset the step index counter at any time as this is then correctly recovered by the encode operation). Hence, for the operation fold, there is no information loss

for a given $\hat{\pi}$ that is folded into a GH meaning that this conversion can be reversed obtaining the same path $\hat{\pi}$ from the resulting GH again.

For the operation encode, we basically need to quantify over the extended form of positions, which is trivially supported by GCs in the expected way. However, checking for graph element to be alive (alive and earliest) for graph matching (for new graph matching) is complicated by the additional use of a step index. In fact, allowing that multiple discrete steps happen at the same time (deviating from the setting where a non-zero delay had to elapse between any two successive states) complicates both checks. For the AC alive, for checking whether certain graph elements in the GH are alive for a virtual position (x_t, x_s), we need to account for the fact that some graph elements may have *cts and dts* variables equal to x_t but *cidx* and *didx* variables satisfying $cidx \leq x_s < didx$, which implies that the matched graph elements are indeed alive in the given $\hat{\pi}$. For the AC earliest, we simply have to check that some graph element has not only a *cts* variable equal to x_t but also a *cidx* variable equal to x_s.

All remaining steps of the correctness proof for the two operations fold and encode together reducing the MTGC satisfaction problem to a GC satisfaction problem are not altered by the transition of MTGL to clock-based PTGTSs. □

Proof for Lemma 3, p. 22: Correct ACs. This lemma is a direct consequence of Theorem 1 since the state conditions $\mathsf{SC}(\hat{\pi})$ obtained for a path $\hat{\pi}$ of the symbolic state space M capture the requirements of the given MTGC θ in terms of an AC over the tpc_i variables. These tpc_i variables are then associated in operation fold with the *cts*/*dts* attributes of the graph elements matched in the GC that is obtained from the MTGC θ using operation encode. □

Proof for Theorem 2, p. 23: Soundness of BMC Approach. We now conclude that the presented BMC approach computes the correct results (a) by encoding the time bound T properly in Step 1 (leading to a finite symbolic tree-shaped state space M in Step 2 according to Lemma 1), (b) by generating the correct symbolic state space M in Step 2 (deriving also a correct timing specification for all paths and therefore all leaf states of M according to Lemma 2), (c) from the soundness of the adapted translation of MTGC satisfaction problem into an equivalent GC satisfaction problem along the lines of [6, 17] in Step 3 (according to Theorem 1), (d) from the correct derivation of zone-restricting ACs for each leaf state according to Lemma 3 in Step 4 (based on Theorem 1), and (e) from the soundness of the backward analysis procedure in Step 5 correctly combining symbolic path suffixes (given by ACs over the tpc_i variables) to longer path suffixes (given by ACs over tpc_i variables omitting/hiding the tpc_i variable representing the time point where the next state has been reached, which allows for arbitrary time points of such later steps in the symbolically represented path suffix). □

B Details for Simplified Running Example

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

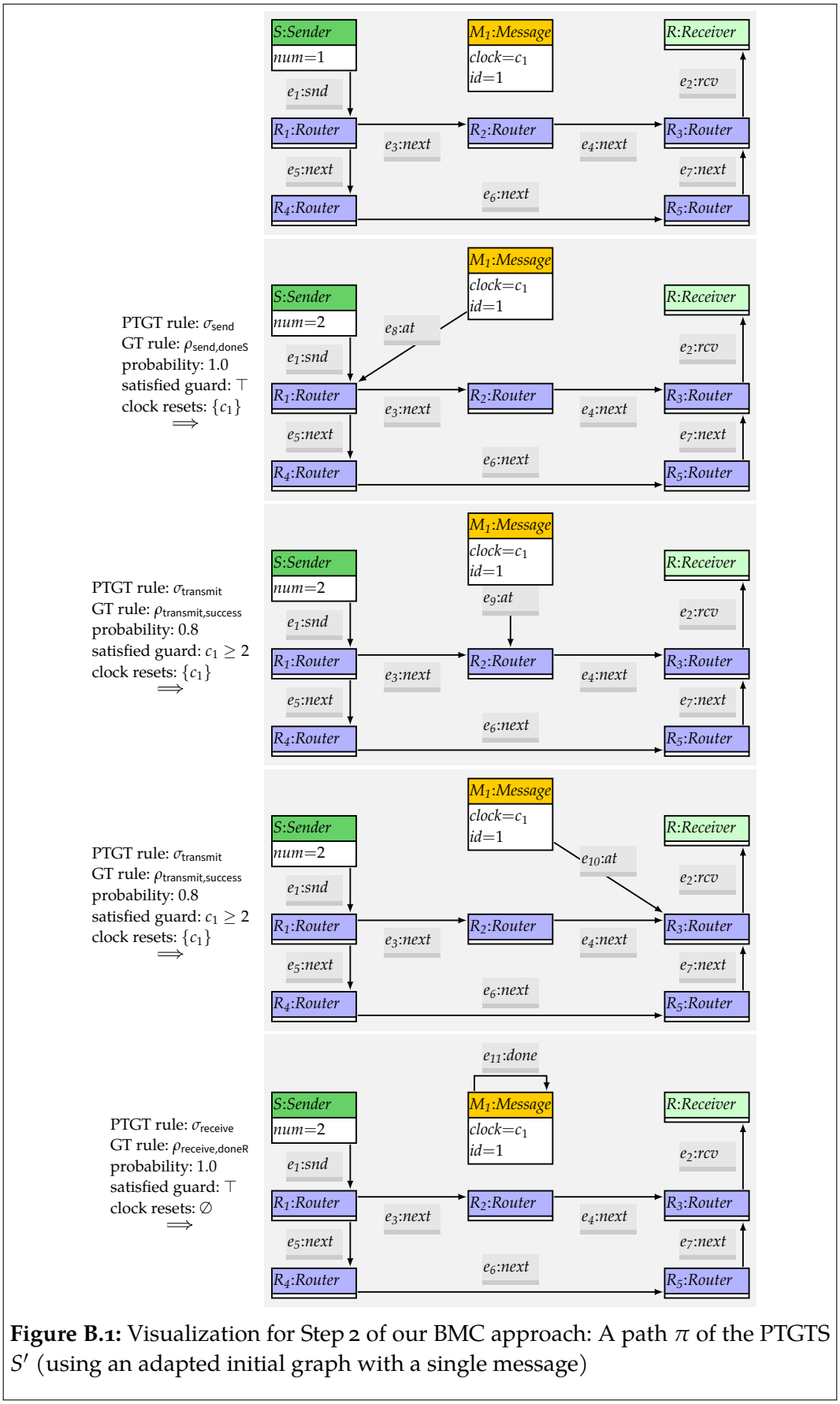


Figure B.1: Visualization for Step 2 of our BMC approach: A path π of the PTGTS S' (using an adapted initial graph with a single message)

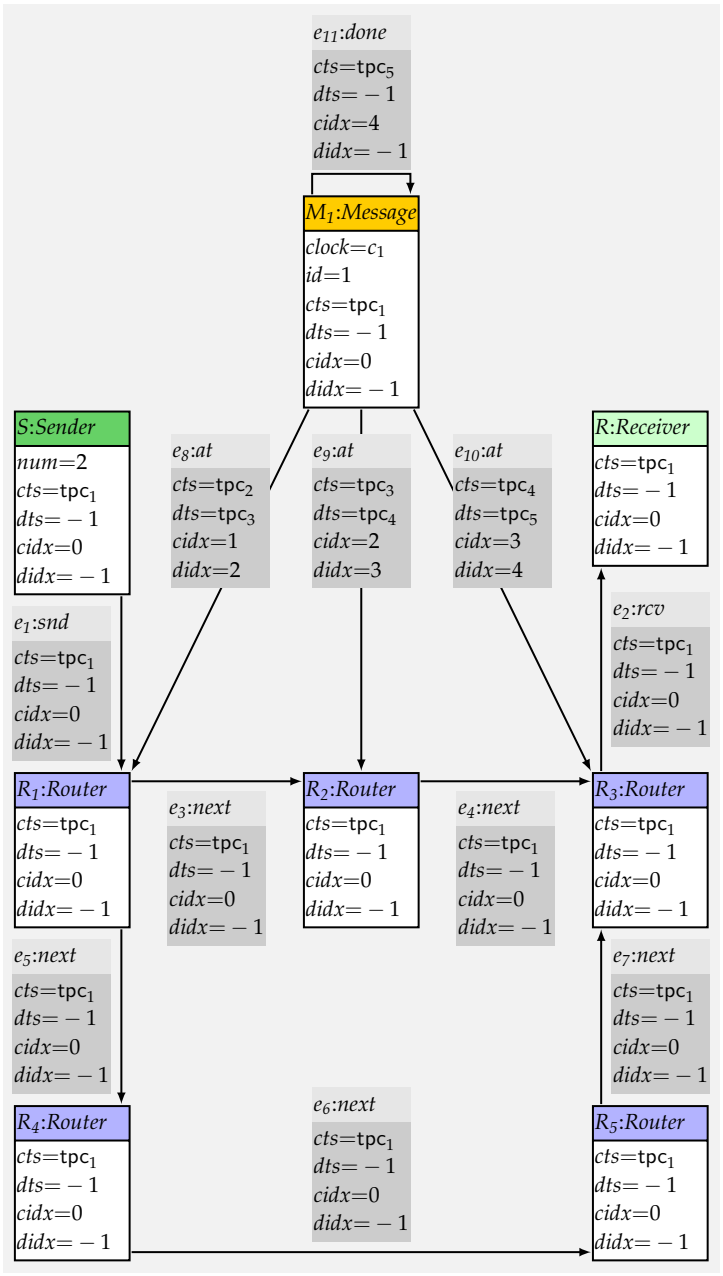
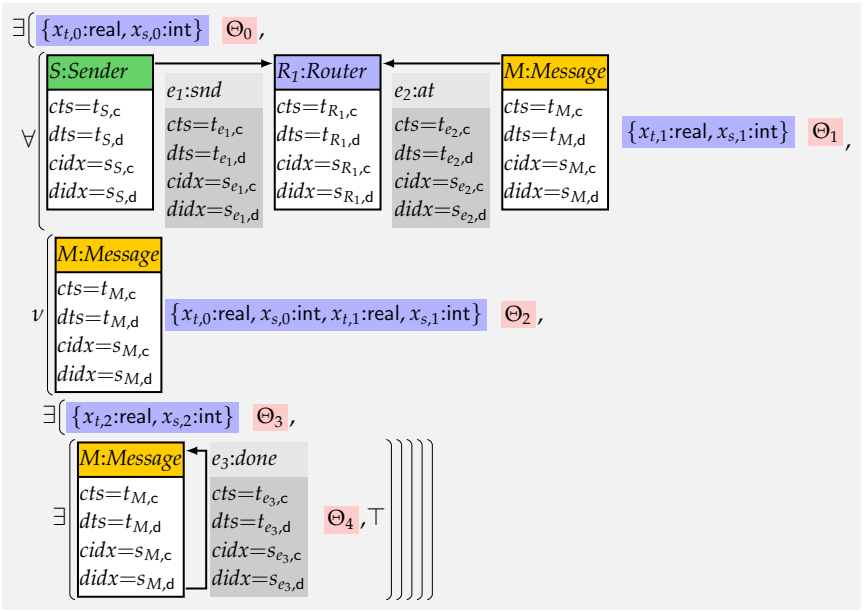


Figure B.2: Visualization for Step 3 of our BMC approach: GH G_H obtained for the path π from Figure B.1



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

Figure B.3: Visualization for Step 3 of our BMC approach: GC ϕ obtained by encoding the MTGC from the PMTGC χ_{max}

Current Technical Reports of the Hasso-Plattner-Institute

Vol.	ISBN	Title	Authors/Editors
145	978-3-86956-528-6	**Learning from Failure : A History-based, Lightweight Test Prioritization Technique Connecting Software Changes to Test Failures**	Falco Dürsch, Patrick Rein, Toni Mattis, Robert Hirschfeld
144	978-3-86956-526-2	**Die HPI Schul-Cloud – Von der Vision zur digitalen Infrastruktur für deutsche Schulen**	Christoph Meinel, Catrina John, Tobias Wollowski, HPI Schul-Cloud Team
143	978-3-86956-531-6	**Invariant Analysis for Multi-Agent Graph Transformation Systems using k-Induction**	Sven Schneider, Maria Maximova, Holger Giese
142	978-3-86956-524-8	**Quantum computing from a software developers perspective**	Marcel Garus, Rohan Sawahn, Jonas Wanke, Clemens Tiedt, Clara Granzow, Tim Kuffner, Jannis Rosenbaum, Linus Hagemann, Tom Wollnik, Lorenz Woth, Felix Auringer, Tobias Kantusch, Felix Roth, Konrad Hanff, Niklas Schilli, Leonard Seibold, Marc Fabian Lindner, Selina Raschack
141	978-3-86956-521-7	**Tool support for collaborative creation of interactive storytelling media**	Paula Klinke, Silvan Verhoeven, Felix Roth, Linus Hagemann, Tarik Alnawa, Jens Lincke, Patrick Rein, Robert Hirschfeld
140	978-3-86956-517-0	**Probabilistic metric temporal graph logic**	Sven Schneider, Maria Maximova, Holger Giese
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 2021**	Christian Bartz, Ralf Krestel

ISBN 978-3-86956-532-3
ISSN 1613-5652