



Probabilistic Metric Temporal Graph Logic

Sven Schneider^(✉) , Maria Maximova , and Holger Giese 

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{sven.schneider,maria.maximova,holger.giese}@hpi.de

Abstract. 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 System (PTGTSS) can be used to suitably capture structure dynamics as well as probabilistic and timed behavior of the system. The model checking support for PTGTSSs 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 PTGTSSs, 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.

Keywords: cyber-physical systems · probabilistic timed systems · qualitative analysis · quantitative analysis · bounded model checking

1 Introduction

Cyber-physical systems often encompass complex concurrent behavior with timing constraints and probabilistic failures on demand [23, 26]. 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) [10] 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 (TGTSSs) [4] and probabilistic aspects have been added to graph transformation in Probabilistic Graph Transformation Systems (PGTSSs) [18]. Finally, the formalism of PTGTSSs [21] combines timed and probabilistic aspects similar to Probabilistic Timed Automata (PTA) [20] and offers model checking support w.r.t. PTCTL [19, 20] properties employing the PRISM model checker [19]. 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) [12, 26].¹

However, structural changes over time in the state space cannot always be directly specified using APs that are *locally* evaluated for each graph.² To express such structural changes over time, MTGL [11, 26] 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) [3] (with limited support by the tool MONPOLY) and the non-metric timed logic EAGLE [1, 14] (with full tool support) have been introduced operating on sets of relations and JAVA objects as state descriptions, respectively. In [7–9], sequences are monitored using model queries to identify complex event patterns of interest. In [15], the Quantified Temporal Logic (QTL) is introduced, which supports bindings and state representation similarly to MFOTL but supports only properties referring to the past and does not support metric bounds in its temporal operators. Besides these logic-based approaches, a multitude of further techniques have been developed based on e.g. automata for monitoring the system’s behavior in the context of runtime monitoring (see [2] for a survey). Finally, note that runtime monitoring (as well as MTGL) focuses on the specification of single sequences whereas the analysis of probabilistic effects is meaningful only when considering a system with a branching behavior (due to non-determinism and/or probabilism).

Obviously, both logics PTCTL and MTGL have distinguishing key strengths but also lack bindings on the part of PTCTL and an operator for expressing

¹ Furthermore, UPPAAL [28] is an analysis tool for timed automata featuring model checking support for standard metric temporal properties and simulation-based support for cyber-physical systems extending timed automata but does not support probabilistic analysis. Lastly, the MODEST toolset [6, 13] also provides analysis support for more complex cyber physical systems representable by e.g. stochastic hybrid automata w.r.t. probabilistic metric temporal requirements.

² For example, tracking (structural changes to) individual graph elements allows to express and analyze deadlines for each individual graph element whereas APs cannot distinguish between individual graph elements and hence cannot help in mapping each of them to their corresponding deadline.

probabilistic requirements on the part of MTGL.³ Furthermore, specifications using both, PTCTL 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 [16] 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 [17].

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 [21] (which is based on a translation of PTGTSs into PTA) with the satisfaction checking approach for MTGL from [11, 26]. 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 Sect. 2, we recall the formalism of PTA. In Sect. 3, we discuss further preliminaries including graph transformation, graph conditions, and the formalism of PTGTSs. In Sect. 4, we recall MTGL and present the extension of MTGL to PMTGL in terms of syntax and semantics. In Sect. 5, we present our BMC approach for PTGTSs w.r.t. PMTGL properties. In Sect. 6, we evaluate our BMC approach by applying its implementation in

³ 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.

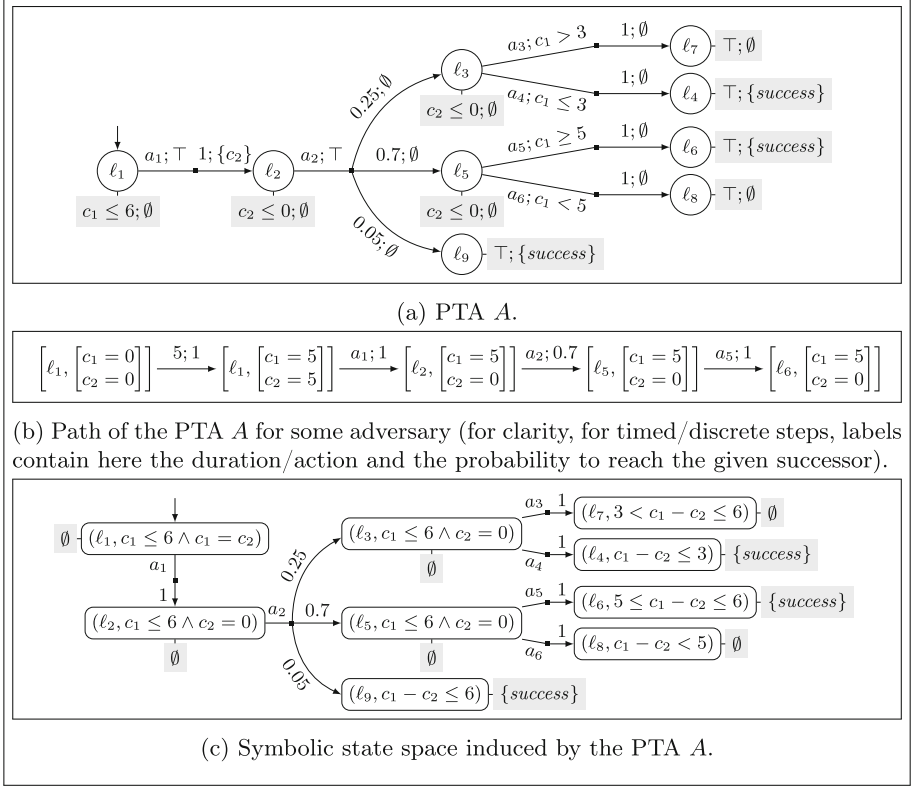


Fig. 1. PTA A , one of its paths, and its symbolic state space.

the tool AUTOGRAPH [24] to our running example. Finally, in Sect. 7, we close the paper with a conclusion and an outlook on future work. Further details are given in a technical report [25].

2 Probabilistic Timed Automata

We briefly review PTA [20], 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 [19].

For a set of clocks X , clock constraints $\psi \in \text{CC}(X)$ 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 $(v : X \rightarrow \mathbf{R}_0^+) \in \text{CV}(X)$ satisfies a zone ψ , written $v \models \psi$, as expected. The initial clock valuation $\text{ICV}(X)$ 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) $\mu : A \rightarrow [0, 1]$, written $\mu \in \text{DPD}(A)$, satisfies $\sum_{a \in A} \mu(a) = 1$. An element $a \in A$ is in the *support* of μ , written $a \in \text{supp}(\mu)$, if $\mu(a) > 0$.

A PTA (see Fig. 1a for an example) is of the form $A = (L, \bar{\ell} \in L, X, I : L \rightarrow \text{CC}(X), \delta \subseteq L \times \mathcal{A} \times \text{CC}(X) \times \text{DPD}(2^X \times L), \mathcal{L} : L \rightarrow 2^{AP})$ where L is a set of locations, $\bar{\ell}$ 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 action⁴, 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 $(\ell, v) \in L \times \text{CV}(X)$ with $v \models I(\ell)$. The initial state is $(\bar{\ell}, \text{ICV}(X))$. 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 Fig. 1b). A timed step $(\ell, v)[\delta, \bar{\mu}](\ell, v + \delta)$ of duration $\delta \in \mathbf{R}^+$ and DPD $\bar{\mu}$ must satisfy that $(\ell, v + \delta')$ is a state for every $0 < \delta' < \delta$ and $\bar{\mu}(\ell, v + \delta) = 1$. A discrete step $(\ell, v)[0, \bar{\mu}](\ell', v')$ of duration 0 and DPD $\bar{\mu}$ using some $(\ell, a, \psi, \mu) \in \delta$ and $(X', \ell') \in \text{supp}(\mu)$ must satisfy $v \models \psi$, $v' = v[X' := 0]$, and $\bar{\mu}(\ell', v') = \sum_{X', v' = v[X' := 0]} \mu(X', \ell')$.⁵

PRISM supports PTA analysis returning minimal and maximal probabilities $\mathcal{P}_{\min=?}(\text{F } ap)$ and $\mathcal{P}_{\max=?}(\text{F } ap)$ with which an ap labeled state can be reached. These two probabilities may differ due to different resolutions of the non-determinism 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 Fig. 1c) intuitively eliminating the impact of guards, invariants, and resets. In this finite symbolic state space, states are of the form $(\ell, \psi) \in L \times \text{CC}(X)$ symbolically representing all states (ℓ, v) with $v \models \psi$.

For example, the PTA A from Fig. 1a (for which adversaries only decide how much time to spend in location ℓ_1), $\mathcal{P}_{\max=?}(\text{F } success) = 0.7 + 0.05$ 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, $\mathcal{P}_{\min=?}(\text{F } success) = 0.05$ 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).

⁴ Actions in edges can be used to describe the purpose of the edge during modeling but also allow to define PTA based on a parallel composition of multiple PTA where these PTA synchronize on common actions. In [21], actions are used to store in PTA edges information about the PTGT steps from which they originate (as also stated in Sect. 3) while actions (being a standard part of PTA) do not play an important role in our BMC approach presented in Sect. 5.

⁵ Here, $\bar{\mu}$ captures the unique successor state for timed steps and sums (possibly multiple non-zero) probabilities of domain elements of μ leading to a common successor state for discrete steps (in particular for the case that clocks to be reset have already the value 0).

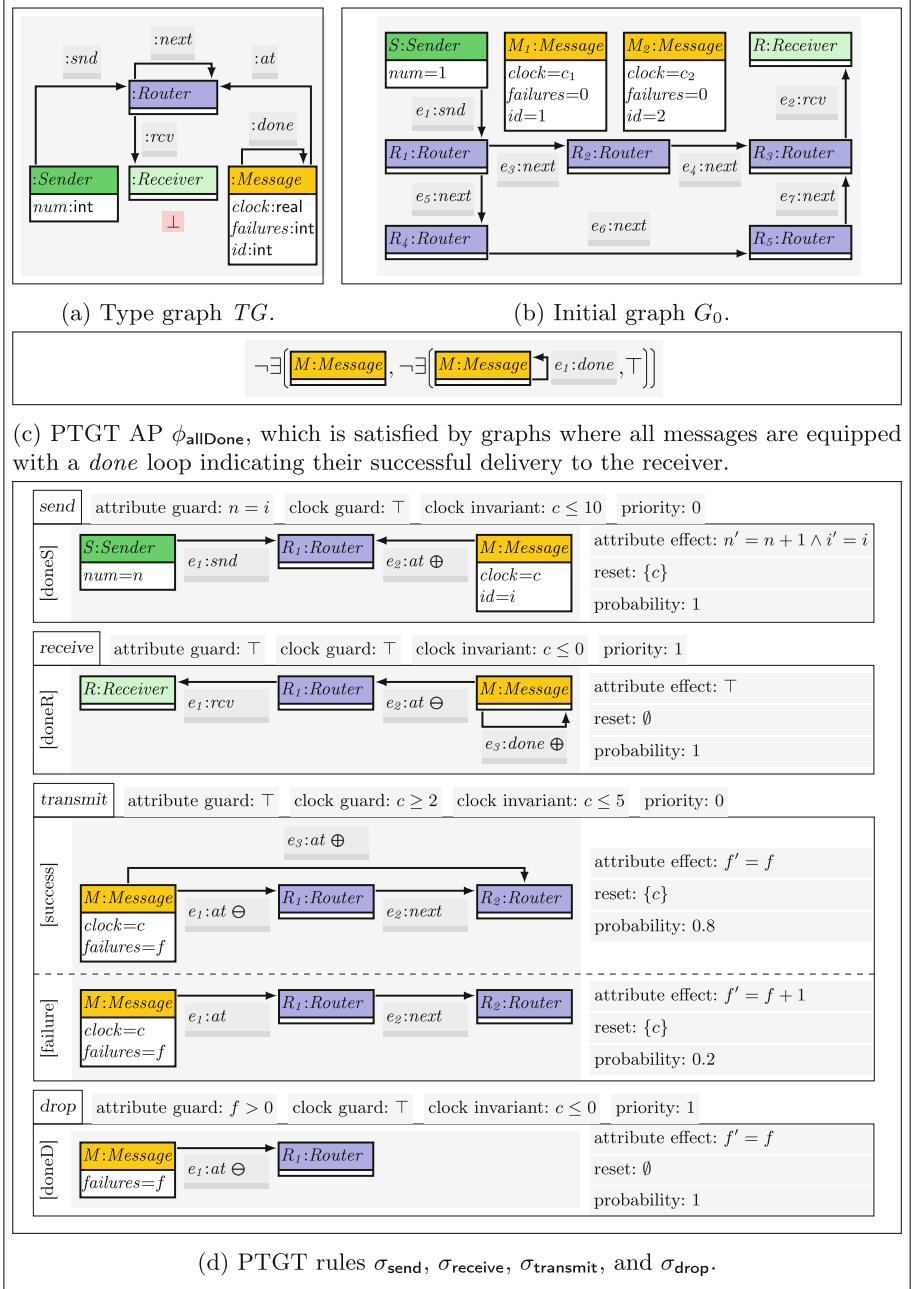


Fig. 2. Components of the PTGTs for the running example.

3 Probabilistic Timed Graph Transformation Systems

We briefly recall graphs, graph conditions, and PTGTSs in our notation.

Using the variation of symbolic graphs [22] from [26], we consider typed attributed graphs (short graphs) (such as G_0 in Fig. 2b), which are typed over a type graph TG (such as TG in Fig. 2a). 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.⁶ Morphisms $m : G_1 \rightarrow G_2$ 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) [12, 26] 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 Fig. 2c 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 [26] 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.⁷

PTGTSs introduced in [21] are a probabilistic real-time extension of Graph Transformation Systems (GTSs) [10]. PTGTSs can be translated into equivalent PTA according to [21] 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 Fig. 2b) captures a sender, which is connected via a network of routers to a receiver, and two messages to be send. The type graph of a PTGTS also identifies attributes representing clocks, which are the *clock* attributes of a message in Fig. 2a.

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

⁶ In our implementation, we employ the SMT solver Z3 to determine satisfiability of ACs. When more complex operations are used in ACs, SMT solvers can be unable to return definitive judgements in time, which does not happen for the running example. If this case would occur, the users would be inform accordingly.

⁷ 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.

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 $(\ell:K \hookrightarrow L, r:K \hookrightarrow R, \gamma, C, p)$ 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.

For our running example, the PTGTS contains the four PTGT rules from Fig. 2d. The PTGT rules σ_{send} , σ_{receive} , and σ_{drop} have each a unique underlying GT rule $\rho_{\text{send,doneS}}$, $\rho_{\text{receive,doneR}}$, and $\rho_{\text{drop,doneD}}$, respectively, whereas the PTGT rule σ_{transmit} has two alternative underlying GT rules $\rho_{\text{transmit,success}}$ and $\rho_{\text{transmit,failure}}$. 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 σ_{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 σ_{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 σ_{transmit} (together with the fact that the clock of the message is reset to 0 whenever σ_{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 Sect. 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 \perp and $f > 1$, respectively. PTGTS steps $(G, v)[\delta, \mu](G', v')$ 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 [21], we can generate directly a symbolic state space (cf. Fig. 1c for the PTA case) using the tool AUTOGRAPH [24] 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) [5].

4 Probabilistic Metric Temporal Graph Logic

Before introducing PMTGL, we recall MTGL [11, 26] 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 [26] as well as with the orthogonal MTGL developments in [27].

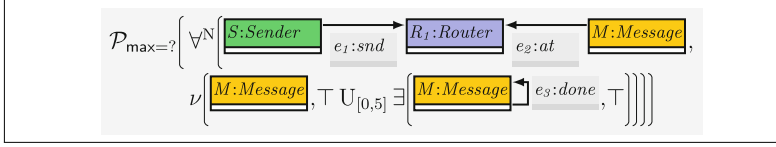


Fig. 3. PMTGC χ_{\max} where the additional MTGL operator *forall-new* (written \forall^N) is derived from the operator *exists-new* by $\forall^N(f, \theta) = \neg \exists^N(f, \neg\theta)$.

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^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 1 (MTGCs). For a graph H , $\theta_H \in \text{MTGC}(H)$ is a metric temporal graph condition (MTGC) over H defined as follows:

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

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 Fig. 3 inside the operator $\mathcal{P}_{\max=?}(\cdot)$. 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 [11, 26], 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.⁸

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 the auxiliary function pos defined below returns $\text{pos}(\pi, i, t, s, \delta) = G$ for the i th step of π and delay δ (since the last change of the step index s).

- If $\pi_0 = ((G, v)[\delta, \mu](G', v'))$, then $\text{pos}(\pi, 0, 0, 0, 0) = G$.
- If $\pi_i = ((G, v)[\delta, \mu](G', v'))$, $\text{pos}(\pi, i, t, s, 0) = G$, and $\delta > 0$, then $\text{pos}(\pi, i, t + \delta', 0, \delta') = G$ for each $\delta' \in (0, \delta)$ and $\text{pos}(\pi, i + 1, t + \delta, 0, 0) = G'$

⁸ To compare positions, we define $(t, s) < (t', s')$ if either $t < t'$ or $t = t'$ and $s < s'$.

- If $\pi_i = ((G, v)[0, \mu](G', v'))$ and $\text{pos}(\pi, i, t, s, \delta) = G$, then $\text{pos}(\pi, i + 1, t, s + 1, 0) = G'$

A match $m : H \hookrightarrow \pi(\tau)$ into the graph at position τ can be propagated forwards/backwards over the steps in a path to the graph $\pi(\tau')$. Such a propagated match $m' : H \hookrightarrow \pi(\tau')$, written $m' \in \text{PM}(\pi, m, \tau, \tau')$, 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, $\text{PM}(\pi, m, \tau, \tau')$ 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 $\theta \in \text{MTGC}(H)$ 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 $(\pi, \tau, m \circ g) \models \theta'$.
- $\theta = \exists^{\mathbf{N}}(f : H \hookrightarrow H', \theta')$ and there are $\tau' \geq \tau$, $m' \in \text{PM}(\pi, m, \tau, \tau')$, 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 $\text{PM}(\pi, m'', \tau', \tau'') = \emptyset$.
- $\theta = \theta_1 \cup_I \theta_2$, $\tau = (t, s)$, and there are $\delta \in I$ and $\tau' = (t + \delta, s')$ s.t.
 - $s' \geq s$ if $\delta = 0$,
 - there is $m' \in \text{PM}(\pi, m, \tau, \tau')$ s.t. $(\pi, \tau', m') \models \theta_2$, and
 - for every $\tau \leq \tau'' < \tau'$ there is $m'' \in \text{PM}(\pi, m, \tau, \tau'')$ s.t. $(\pi, \tau'', m'') \models \theta_1$.

Moreover, if $\theta \in \text{MTGC}(\emptyset)$, $\tau = (0, 0)$, and $(\pi, \tau, i(\pi(\tau))) \models \theta$, 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 $\theta \in \text{MTGC}(\emptyset)$ is an MTGC over the empty graph. Moreover, we also call expressions of the form $\mathcal{P}_{\min=?}(\theta)$ and $\mathcal{P}_{\max=?}(\theta)$ 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, $\mathcal{P}_{\min=?}(\theta)$ and $\mathcal{P}_{\max=?}(\theta)$ 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 Fig. 3 for the PTGTS from Fig. 2 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 σ_{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 σ_{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 for each of the messages succeeded, the maximal probability to satisfy the inner MTGC is $(0.8 \times 0.8)^2 = 0.8^4$. Using $\mathcal{P}_{\min=?}(\cdot)$ 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 $\mathcal{P}_{\max=?}(\theta)$ or $\mathcal{P}_{\min=?}(\theta)$ for computing expected probabilities since they are sufficient to analyze PMTGCs of the form $\mathcal{P}_{\sim c}(\theta)$.⁹ In the subsequent presentation, we focus on the case of $\mathcal{P}_{\max=?}(\theta)$ and point out differences for the case of $\mathcal{P}_{\min=?}(\theta)$ 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 and that the PTGT invariants are then disabled. 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 $\rho = (\ell:K \hookrightarrow L, r:K \hookrightarrow R, \gamma)$ 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

⁹ For example, $\mathcal{P}_{\min=?}(\theta) = c$ implies satisfaction of $\mathcal{P}_{\geq c'}(\theta)$ for any $c' \leq c$.

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).¹⁰ 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 .*

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. Sect. 2)), we may construct a symbolic state space for a 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 $s_1[\mu_2]s_2[\mu_3] \dots s_n$ 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 $s_1[\delta_2, \mu_2]s_2[\delta_3, \mu_3] \dots s_n$ 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 Fig. 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 path¹¹ $\hat{\pi}$ of M a *timing specification* $\text{TS}(\hat{\pi})$. 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 $\text{TS}(\hat{\pi})$, 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 $\text{TS}(\hat{\pi})$ the time point when state i has been just reached in $\hat{\pi}$. Hence, $\text{TS}(\hat{\pi})$ 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* $\text{ttv}(\pi)$ to be the AC equating the time point clock tpc_i and the time point $\sum_{1 \leq k < i} \delta_k$ of the i th step in π . Using this notion, we characterize that π is a timed realization of $\hat{\pi}$ (performing the same discrete steps) when $\text{TS}(\hat{\pi}) \wedge \text{ttv}(\pi)$ is satisfiable.

To define $\text{TS}(\hat{\pi})$, we use a map $\text{LastReset}(k, c) = k'$ 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

¹⁰ 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.

¹¹ We only consider paths starting in the initial state and ending in a leaf state.

can be easily computed by iterating once through $\hat{\pi}$). Recall that all clocks c are reset in the initial state, i.e., $\text{LastReset}(1, c) = 1$. We include the ACs in $\text{TS}(\hat{\pi})$ as follows for each state s_i . Firstly, when $i = 1$ (i.e., s_i is the initial state), we add $\text{tpc}_1 = 0$ to $\text{TS}(\hat{\pi})$. Secondly, when $i > 1$, we add $\text{tpc}_{i-1} \leq \text{tpc}_i$ to $\text{TS}(\hat{\pi})$. Thirdly, when s_i was reached by respecting a guard ψ (implying $i > 1$), we add ψ to $\text{TS}(\hat{\pi})$ after replacing each clock c contained in ψ by $\text{tpc}_i - \text{tpc}_{k'}$ where $k' = \text{LastReset}(i - 1, c)$.¹² Fourthly, when s_i was reached by respecting an invariant ψ' , we add ψ' to $\text{TS}(\hat{\pi})$ after replacing each clock c contained in ψ' by $\text{tpc}_{i+1} - \text{tpc}_{k'}$ where $k' = \text{LastReset}(i, c)$.¹³

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 $\text{TS}(\hat{\pi})$ and the time points at which states are reached in the timed realizations π of $\hat{\pi}$.*

For our running example (considering the restriction to a single message in the initial graph), for a path $\hat{\pi}_{\text{ex}}$ 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) $\text{TS}(\hat{\pi}_{\text{ex}})$ as the conjunction of $\text{tpc}_1 = 0$, $0 \leq \text{tpc}_2 \leq 10$, $\text{tpc}_2 + 2 \leq \text{tpc}_3 \leq \text{tpc}_2 + 5$, and $\text{tpc}_3 + 2 \leq \text{tpc}_4 = \text{tpc}_5 \leq \text{tpc}_3 + 5$ essentially encoding the guards and invariants as expected.¹⁴

In the next two steps of our algorithm, we derive for the MTGC θ (contained in the given PMTGC $\mathcal{P}_{\min=?}(\theta)$ or $\mathcal{P}_{\max=?}(\theta)$) and a path $\hat{\pi}$ an AC describing timed realizations π of $\hat{\pi}$ satisfying θ . For our running example and the path $\hat{\pi}_{\text{ex}}$ from above, this derived AC will be $\text{tpc}_5 - \text{tpc}_2 \leq 5$ 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 $\text{TS}(\hat{\pi})$ and the derived AC are satisfiable at once and (b) failing paths $\hat{\pi}$ to be those where $\text{TS}(\hat{\pi})$ and the negated derived AC are satisfiable together.

Step 3: From MTGC Satisfaction to GC Satisfaction

Following the satisfaction checking approach for MTGL from [11, 26], 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.¹⁵ Technically, the MTGC θ is

¹² Intuitively, $\text{tpc}_i - \text{tpc}_{k'}$ is the duration between the last reset of c and the time point when the guard was checked upon state transition to s_i .

¹³ Intuitively, $\text{tpc}_{i+1} - \text{tpc}_{k'}$ 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} .

¹⁴ Note that $\text{tpc}_4 = \text{tpc}_5$ since the message reception by R takes no time.

¹⁵ The operations `fold` and `encode` presented here are adaptations of the corresponding operations from [11, 26] to the modified MTGL satisfaction relation for PTGTSs from Definition 3 allowing for successive discrete steps with zero-time delay in-between.

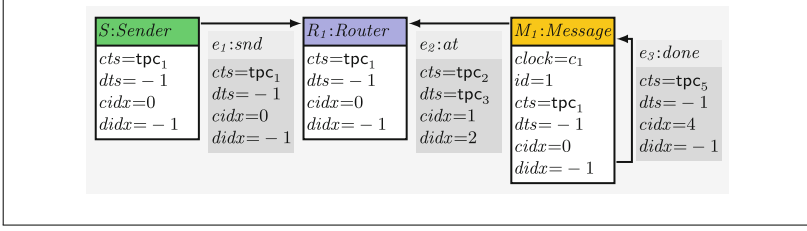


Fig. 4. A part of the GH G_H obtained using the operation fold for the path $\hat{\pi}_{ex}$ of the running example.

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 $\text{ttv}(\pi)$ 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, $\text{encode}(\theta) = \phi$, $\hat{\pi}$ is a path through the symbolic state space constructed for the PTGTS S' , $\text{fold}(\hat{\pi}) = G_H$, 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 $\text{ttv}(\pi)$, then $\pi \models \theta$ iff $G'_H \models \phi$.*

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 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 $cts = tpc_1$ and $cidx = 0$, (ii) nodes/edges added in step i of $\hat{\pi}$ are equipped with attributes $cts = tpc_i$ and $cidx = i$, (iii) nodes/edges deleted in step i of $\hat{\pi}$ are equipped with attributes $dts = tpc_i$ 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 $\hat{\pi}_{ex}$ from our running example, see Fig. 4 depicting the part of the GH G_H that is matched when checking the GC $\text{encode}(\theta)$ 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 $\text{TS}(\hat{\pi})$ 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 $\text{encode}(\theta) = \phi$ against the GH $\text{fold}(\hat{\pi}) = G_H$. The conjunction of $\text{TS}(\hat{\pi})$ and $\gamma_{\hat{\pi}}$ is then recorded in the set of *state conditions* $\text{SC}(s)$ 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 .¹⁶ In Step 5, we also use the notion of *state probability* $\text{SP}(s)$ assigning a probability of 1 to a state s when the AC in $\text{SC}(s)$ 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 $\text{TS}(\hat{\pi}) \wedge \gamma_{\hat{\pi}} \wedge \text{ttv}(\pi)$ is satisfiable.*

For our running example, when checking the encoded MTGC (cf. Fig. 3) for the GH partially given in Fig. 4, (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* $(\text{tpc}_2, 1)$ characterizing all positions $(t, 1)$ where $\text{tpc}_2 = t$. The ACs in the encoded MTGC then ensure that e.g. e_1 is alive since it was created not after tpc_2 ($\text{cts} = \text{tpc}_1 \leq \text{tpc}_2$ and $\text{cidx} = 0 \leq 1$) and it has never been deleted ($\text{dts} = -1$) whereas e.g. e_2 is alive since it was created at $(\text{tpc}_2, 1)$ and it has been deleted strictly later ($\text{dts} = \text{tpc}_2$ but $1 < \text{didx}$). Moreover, the matched graph elements are not alive earlier since e_2 was created at $(\text{tpc}_2, 1)$. For (b), all matched graph elements are alive at $(\text{tpc}_5, 4)$. Overall, we obtain (after simplification) the AC requiring that $\text{tpc}_5 - \text{tpc}_2 \leq 5$ as the encoded MTGC expresses the time bound ≤ 5 used in the *until* operator. For the last state of the path $\hat{\pi}_{\text{ex}}$, we obtain the AC $\text{TS}(\hat{\pi}_{\text{ex}}) \wedge \text{tpc}_5 - \text{tpc}_2 \leq 5$, which is e.g. satisfied by the valuation $\{\text{tpc}_1 = 0, \text{tpc}_2 = 0, \text{tpc}_3 = 2, \text{tpc}_4 = 4, \text{tpc}_5 = 4\}$ representing a timed realization π_{ex} of $\hat{\pi}_{\text{ex}}$ 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 $\text{SC}(s)$. However, this kind of specification of target states is not supported by PRISM, which requires a clock-independent

¹⁶ For the case of $\mathcal{P}_{\min=?}(\theta)$, we define $\text{SC}(s) = \{\text{TS}(\hat{\pi}) \wedge \neg\gamma_{\hat{\pi}}\}$.

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 $SC(s)$. For the symbolic state space in Fig. 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 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 $SC(s)$ 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 , $S' \subseteq \text{supp}(\mu)$ is a subset of the target states of μ , f selects for each target state $s' \in S'$ an AC from $SC(s')$, $\gamma = \exists \text{tpc}_{I(s)} \cdot \bigwedge_{s' \in S'} f(s')$ is the AC derived for the state s based on the selections S' and f , γ is satisfiable, and $p = \sum_{s' \in S'} (\mu(s') \times SP(s'))$ is the new probability for s based on the selections S' and f , then (a) $SC(s)$ and $SP(s)$ are changed to $\{\gamma\}$ and p when $p > SP(s)$ recording the AC γ and the new maximal probability p derived for s and (b) $SC(s)$ is changed to $SC(s) \cup \{\gamma\}$ when $p = SP(s)$ recording an additional AC γ and not changing the probability $SP(s)$.*

Finally, using our BMC approach introduced in this section, we derive the expected maximal probability.¹⁷

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 .*

¹⁷ For $\mathcal{P}_{\min=?}(\theta)$, the procedure from Definition 6 returns $1 - \mathcal{P}_{\min=?}(\theta)$ maximizing the probability of failing paths by minimizing the probability for successful paths.

6 Evaluation

Our implementation of the presented BMC approach in AUTOGRAPH [24] 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 [21, 23] 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 [27].

References

1. Barringer, H., Goldberg, A., Havelund, K., Sen, K.: Rule-based runtime verification. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 44–57. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_5
2. Bartocci, E., et al.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification. LNCS, vol. 10457, pp. 135–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_5
3. Basin, D.A., Klaedtke, F., Müller, S., Zalinescu, E.: Monitoring metric first-order temporal properties. *J. ACM* **62**(2), 15:1–15:45 (2015). <https://doi.org/10.1145/2699444>
4. Becker, B., Giese, H.: 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, pp. 203–210. IEEE Computer Society (2008). ISBN 978-0-7695-3132-8. <https://doi.org/10.1109/ISORC.2008.13>. <https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4519543>
5. Bengtsson, J., Yi, W.: Timed automata: semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_3
6. Bohnenkamp, H.C., D’Argenio, P.R., Hermanns, H., Katoen, J.: MODEST: a compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.* **32**(10), 812–830 (2006). <https://doi.org/10.1109/TSE.2006.104>
7. Búr, M., Szilágyi, G., Vörös, A., Varró, D.: Distributed graph queries for runtime monitoring of cyber-physical systems. In: Russo, A., Schürr, A. (eds.) FASE 2018. LNCS, vol. 10802, pp. 111–128. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89363-1_7
8. Dávid, I., Ráth, I., Varró, D.: Foundations for streaming model transformations by complex event processing. *Softw. Syst. Model.* **17**(1), 135–162 (2016). <https://doi.org/10.1007/s10270-016-0533-1>
9. Dávid, I., Ráth, I., Varró, D.: Streaming model transformations by complex event processing. In: Dingel, J., Schulte, W., Ramos, I., Abrahão, S., Insfran, E. (eds.) MODELS 2014. LNCS, vol. 8767, pp. 68–83. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11653-2_5
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Springer, Heidelberg (2006). <https://doi.org/10.1007/3-540-31188-2>
11. Giese, H., Maximova, M., Sakizoglou, L., Schneider, S.: Metric temporal graph logic over typed attributed graphs. In: Hähnle, R., van der Aalst, W. (eds.) FASE 2019. LNCS, vol. 11424, pp. 282–298. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16722-6_16
12. Habel, A., Pennemann, K.: Correctness of high-level transformation systems relative to nested conditions. *Math. Struct. Comput. Sci.* **19**(2), 245–296 (2009). <https://doi.org/10.1017/S0960129508007202>
13. Hahn, E.M., Hartmanns, A., Hermanns, H., Katoen, J.: A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods Syst. Des.* **43**(2), 191–232 (2013). <https://doi.org/10.1007/s10703-012-0167-z>
14. Havelund, K.: Rule-based runtime verification revisited. *Int. J. Softw. Tools Technol. Transf.* **17**(2), 143–170 (2015). <https://doi.org/10.1007/s10009-014-0309-2>

15. Havelund, K., Peled, D.: Efficient runtime verification of first-order temporal properties. In: Gallardo, M.M., Merino, P. (eds.) SPIN 2018. LNCS, vol. 10869, pp. 26–47. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94111-0_2
16. Jansen, N., Dehnert, C., Kaminski, B.L., Katoen, J.-P., Westhofen, L.: Bounded model checking for probabilistic programs. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 68–85. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_5
17. Katoen, J.: The probabilistic model checking landscape. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, 5–8 July 2016, pp. 31–45. ACM, New York (2016). <https://doi.org/10.1145/2933575.2934574>
18. Krause, C., Giese, H.: Probabilistic graph transformation systems. In: Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2012. LNCS, vol. 7562, pp. 311–325. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33654-6_21
19. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
20. Kwiatkowska, M., Norman, G., Sproston, J., Wang, F.: Symbolic model checking for probabilistic timed automata. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT-2004. LNCS, vol. 3253, pp. 293–308. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_21
21. Maximova, M., Giese, H., Krause, C.: Probabilistic timed graph transformation systems. *J. Log. Algebr. Meth. Program.* **101**, 110–131 (2018). <https://doi.org/10.1016/j.jlamp.2018.09.003>
22. Orejas, F.: Symbolic graphs for attributed graph constraints. *J. Symb. Comput.* **46**(3), 294–315 (2011). <https://doi.org/10.1016/j.jsc.2010.09.009>
23. RailCab Project. <https://www.hni.uni-paderborn.de/cim/projekte/railcab>
24. Schneider, S.: AutoGraph. <https://github.com/schneider-sven/AutoGraph>
25. Schneider, S., Maximova, M., Giese, H.: Probabilistic metric temporal graph logic. Technical report, 146. Hasso Plattner Institute, University of Potsdam (2022)
26. Schneider, S., Maximova, M., Sakizoglou, L., Giese, H.: Formal testing of timed graph transformation systems using metric temporal graph logic. *Int. J. Softw. Tools Technol. Transfer* **23**(3), 411–488 (2021). <https://doi.org/10.1007/s10009-020-00585-w>
27. Schneider, S., Sakizoglou, L., Maximova, M., Giese, H.: Optimistic and pessimistic on-the-fly analysis for metric temporal graph logic. In: Gadducci, F., Kehrer, T. (eds.) ICGT 2020. LNCS, vol. 12150, pp. 276–294. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51372-6_16
28. UPPAAL. Department of Information Technology at Uppsala University, Sweden and Department of Computer Science at Aalborg University, Denmark (2021). <https://uppaal.org/>