Interval Probabilistic Timed Graph Transformation Systems

Maria Maximova, Sven Schneider, Holger Giese

Technische Berichte Nr. 134

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 | 134

Maria Maximova | Sven Schneider | Holger Giese

Interval Probabilistic Timed Graph Transformation Systems

Universitätsverlag Potsdam

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

Universitätsverlag Potsdam 2021
http://verlag.ub.uni-potsdam.de/

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

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

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

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
https://doi.org/10.25932/publishup-51289
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-512895

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-502-6

For complex distributed embedded probabilistic real-time systems, ensuring correctness of their software components is of great importance. The rule-based formalism of Probabilistic Timed Graph Transformation Systems (PTGTSs) allows for modeling and analysis of such systems where states can be represented by graphs and where timed and probabilistic behavior is important. In PTGTSs, probabilistic behavior is specified by assigning precise probabilities to rules. However, for embedded systems, only lower and upper probability bounds may be estimated because unknown physical effects may influence the probabilities possibly changing them over time.

In this paper, we (a) introduce the formalism of Interval Probabilistic Timed Graph Transformation Systems (IPTGTSs) in which rules are equipped with probability intervals rather than precise probabilities and (b) extend the preexisting model checking approach for PTGTSs to IPTGTSs w.r.t. worst-case/best-case probabilistic timed reachability properties using an encoding of probability intervals. Moreover, we ensure that this adapted model checking approach is applicable to IPTGTSs for which the finiteness of the state space may only be a consequence of the timing constraints. Finally, in our evaluation, we apply an implementation of our model checking approach in AutoGraph to a running example. [1, 2]

[1]Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — 241885098, 148420506.

[2]A shorter version of this report has been published earlier as a paper in [14].

Contents

1 Introduction

Software correctness plays an important role in the ever growing area of complex distributed embedded probabilistic real-time systems. In this context, modeling formalisms allowing for formal analysis while capturing relevant system aspects are required for designing, understanding, and improving the behavior of such systems.

Many probabilistic real-time systems with complex coordination behavior or spatial movements of components can be modeled using the formalism of PTGTSs [13] when the states of the system can be represented by graphs. The formalism of PTGTSs allows for structure dynamics by means of rule-based graph transformation (cf. [3]), timed behavior (employing clocks as in Timed Automata (TA) [1]), and probabilistic choices (as in Probabilistic Automata (PA) [19]) among alternative outcomes of graph transformation steps. As usual, nondeterministic models such as PTGTSs, where the passage of time competes with possibly multiple rule applications, implicitly cover real-time systems by the resolution of their nondeterminism. However, assigning precise probabilities to the alternative outcomes of graph transformation steps in PTGTSs is insufficient when unknown physical effects may affect the actual probabilities (possibly even over time). Hence, PTGTSs may only be employed as a suitable modeling formalism when (at least) pseudo-random variables are used to decide probabilistic choices.

The subformalisms of Timed Graph Transformation Systems (TGTSs) [2, 15] and Probabilistic Graph Transformation Systems (PGTSs) [5, 7] including tool support for their formal analysis have been introduced before. Essentially, tool-based analysis support is obtained by translating a PTGTS, TGTS, or PGTS into the corresponding Probabilistic Timed Automaton (PTA) [10], TA, or PA, respectively, and by then reusing the existing model checking support for the resulting automata. In particular, the tools Prism [7] and Uppaal [20] support the model checking of PTA and TA, respectively, with varying feature sets.

As for PTGTSs, the precise probabilities required in PTA may not be appropriate for the system at hand. To relax this precision, Interval Probabilistic Timed Automata (IPTA) [4, 21] have been defined as an extension of PTA where probability intervals are used instead of precise probabilities. These probability intervals are then resolved to precise probabilities nondeterministically at use-time to derive steps.

In this paper, we introduce the formalism of IPTGTSs as an extension of PTGTSs by integrating the handling of probability intervals from IPTA to allow for the modeling of systems where only lower and upper probability bounds can be estimated for some probabilistic steps. Following our work on PTGTSs and IPTA [4], we present a formal translation of IPTGTSs into PTA via IPTA (for the case of finite state spaces) to support the modeling of *structure dynamics*, *timed behavior*, and *interval probabilistic behavior* using IPTGTSs and their analysis w.r.t. best-case/worst-case probabilistic

reachability properties using Prism. Hereby, we improve upon our earlier work in [13] by constructing the state space of the TGTS underlying the given IPTGTS while using Uppaal to ensure that all obtained states are reachable for the given IPTGTS. Hence, we enable the analysis of an IPTGTS with a finite state space even when the state space of its underlying Graph Transformation System (GTS) is infinite. See Figure 1.1 for an overview of the translations and tool support for the involved modeling formalisms where Probabilistic Timed Systems (PTSs) and Interval Probabilistic Timed Systems (IPTSs) are underlying semantic models.

As a running example, we consider a gossiping protocol where all agents in a directed wireless network have a local Boolean value. The Boolean value true represents the information that must be propagated to all agents. At run-time, agents with Boolean value true attempt to send this value along the directed physical channels given by edges. Each sending operation is subject to probabilistic choice where the message is transported successfully through the channel with a probability between 0.7 and 0.8 possibly being affected by e.g. the available energy of the sender or the spatial distance between agents. Moreover, due to limited energy and imperfect local clocks, each agent may send a message at most every 2 to 5 time units. Lastly, we evaluate the described system in terms of e.g. the best-case/worst-case probability that each agent adopts the Boolean value true within a given time bound but also in terms of the number of sending errors processed by an observer that counts and deletes them.

This paper is structured as follows. In chapter 2, we introduce preliminaries for our approach including graph transformation and IPTA. In chapter 3, we introduce the novel formalism of IPTGTSs as an extension of PTGTSs widening probabilistic choices from precise probabilities to probability intervals. In chapter 4, we present the steps of our translation-based model checking approach for IPTGTSs. In chapter 5, we evaluate our approach by applying its implementation in the tool AutoGraph to our running example. In chapter 6, the paper is closed with a conclusion and an outlook on future work. Finally, the appendix covers additional technicals details and examples omitted in the main body for readability. In Appendix A, we present the two step translation of IPTA into PTSs via IPTSs (which complements the two step translation of IPTA to PTSs via PTA). In Appendix B, we present established means, in particular centered around Prism and PTCTL for the specification of PTSs and therefore also of IPTGTSs, PTGTSs, IPTA, and PTA. In Appendix C, we present an analysis algorithm for IPTA and best-case/worst-case probabilistic reachability properties as a foundation for the also presented translation of IPTA into PTA preserving such properties. In Appendix D, we present an example for the translation of an IPTGTS into an IPTA that is equivalent w.r.t. the considered best-case/worst-case probabilistic reachability properties.

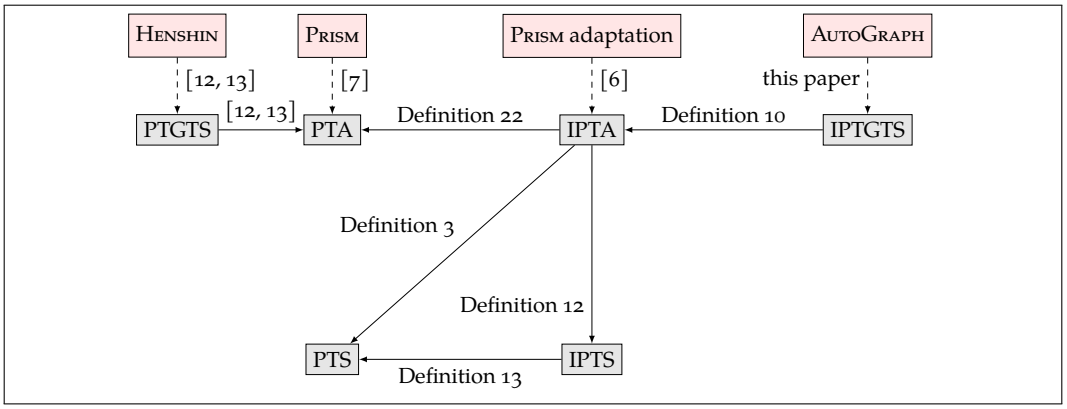


Figure 1.1: Overview of tools and formalisms: Henshin supports the modeling of PTGTS and their translation to PTA; AutoGraph supports the modeling of IPTGTS and their translation to IPTA and PTA; Prism supports the analysis of PTA and a custom adaptation of Prism supports the analysis of IPTA; PTA and IPTA induce PTS and IPTS as underlying semantic models.

2 Preliminaries

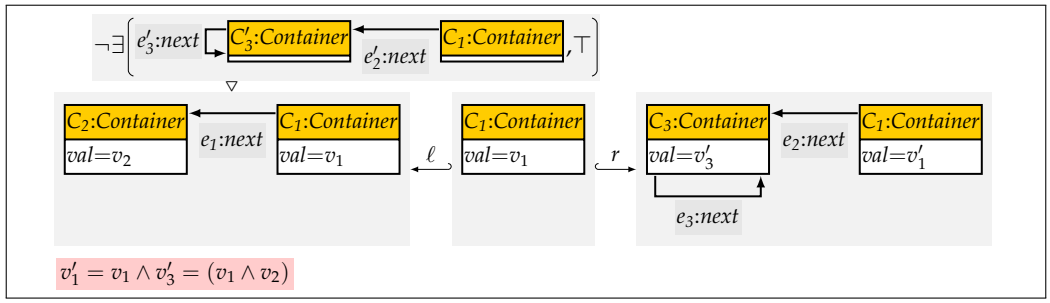
In this section, we introduce graphs, graph transformation, IPTA, and probabilistic timed reachability problems to be solved for IPTA as preliminaries for the subsequent presentation of IPTGTSs and our model checking approach.

Employing the variation of symbolic graphs [17] from [18], we consider typed attributed graphs (such as the graph G_0 in Figure 2.2d), which are typed over a type graph TG (such as the one in Figure 2.2a). The values of the variables connected to attributes are specified using Attribute Conditions (ACs) over a many sorted first-order attribute logic. The AC $\perp$ (false) in TG means thereby that the type graph does not restrict attribute values. Graph Transformation (GT) is then executed 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 [18] for technical details). A GT rule specifies (a) that 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 monomorphisms ℓ and r, respectively, according to a Double Pushout (DPO) diagram and (b) that the values of variables in the resulting graph are derived from those of G using the AC γ (e.g. $x' = x + 2$) in which the variables from L and R are used in unprimed and primed form, respectively. Nested application conditions are straightforwardly supported by our approach but, to improve readability, they are not used in the running example and omitted subsequently. For example, see Figure 2.1a for a graph transformation rule and Figure 2.1b for an example of a graph transformation step applying this rule.

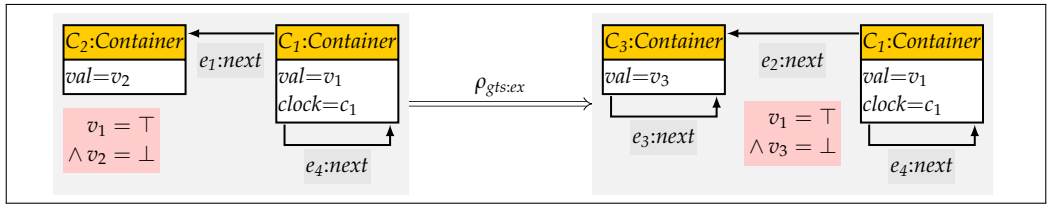
We now recall IPTA [4, 21], which subsume TA [1] where clocks are used to capture real-time phenomena and PTA [10] where probabilistic choices are used additionally to approximate/describe the likelihood of outcomes of certain steps. First, we provide required notions on clocks and (intervals of) probabilities.

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

For a countable set A, a Discrete Interval Probability Distribution (DIPD) characterizes a non-empty set of (discrete) Probability Mass Functions (PMFs) $\mu : A \longrightarrow [0, 1]$ by assigning to each $a \in A$ an interval $[x_a, y_a]$ such that it is possible to choose one element from each interval to obtain a sum of 1. Formally, a pair $(\mu_1 : A \longrightarrow [0, 1], \mu_2 : A \longrightarrow [0, 1])$ is a DIPD, written $(\mu_1, \mu_2) \in \mathsf{DIPD}(A)$, if the following two properties hold: (i) $\mu_1(a) \leq \mu_2(a)$ for each $a \in A$ and (ii) $\sum \{\!|\mu_1(a) \mid a \in A|\!\} \leq 1 \leq$



(a) Graph transformation rule $\rho_{gts:ex}$ with two monomorphisms ℓ and r and a negative application condition describing the removal of node C_2 (with its single attribute *val*) and edge e_1, the addition of node C_3 (with its single attribute *val*) and the edges e_2 and e_3, unless (application condition above) there is already such a node C_3' attached in the same way to the node C_1, the preservation of the *val* attribute of C_1, and the restriction of the attribute *val* of C_3 to the conjunction of the *val* attributes of C_1 and C_2.



(b) Graph transformation step applying the rule from Figure 2.1a. We refer to [18] for details as to how the step is obtained.

Figure 2.1: Example of a GT rule and GT step

$\sum \{\!|\mu_2(a) \mid a \in A|\!\}$ using summation over multisets. These two properties ensure non-emptiness of intervals and non-emptiness of characterized PMFs, respectively. $\mu : A \rightarrow [0,1]$ is then such a characterized PMF in the semantics of (μ_1, μ_2), written $\mu \in \langle(\mu_1, \mu_2)\rangle$, if $\mu_1(a) \leq \mu(a) \leq \mu_2(a)$ for each $a \in A$. Note that (μ, μ) is also a DIPD over A. The *support* of (μ_1, μ_2), written $\mathsf{supp}((\mu_1, \mu_2))$, contains all $a \in A$ for which the right interval border $\mu_2(a)$ is non-zero. For example, the semantics of the DIPD $(\mu_1 = \{\ell_0 \mapsto 0.2, \ell_1 \mapsto 0.7\}, \mu_2 = \{\ell_0 \mapsto 0.3, \ell_1 \mapsto 0.8\})$ contains, among others, the two PMFs $\mu = \{(\ell_0, 0.3), (\ell_1, 0.7)\}$ and $\mu' = \{(\ell_0, 0.2), (\ell_1, 0.8)\}$. The *point DIPD* for a distinguished elment $a \in A$, written $\mathsf{DIPD}_1(A, a)$, is the unique DIPD (μ, μ) with $\mu(a) = 1$ and mapping all other elements of A to 0. Lastly, a PMF $\mu : A \rightarrow [0,1]$ can be used to scale a probability vector $p : A \rightarrow [0,1]$, written $(\mu \times p) : A \rightarrow [0,1]$, where $(\mu \times p)(a) = \mu(a) \times p(a)$.

An IPTA (such as A_1 from Figure 2.3) consists of (a) a set of locations with a distinguished initial location (such as ℓ_0), (b) a set of clocks (such as c) which are initially set to 0, (c) an assignment of a set of APs (such as $\{done\}$) to each location (for subsequent analysis of e.g. reachability properties), (d) an assignment of constraints over clocks to each location as invariants (such as $c \leq 5$), and (e) a set of interval probabilistic timed edges. Each interval probabilistic timed edge consists thereby of (i) a single source location, (ii) at least one target location, (iii) an action

(such as a), (iv) a clock constraint (such as $c \geq 2$) specifying as a guard when the edge is enabled based on the current values of the clocks, and (v) a DIPD assigning an interval probability (such as $[0.2, 0.3]$) to each pair consisting of a set of clocks to be reset (such as $\{c\}$) and a target location to be reached.

Definition 1 (Interval Probabilistic Timed Automaton (IPTA))**.** An *interval probabilistic timed automaton* (*IPTA*) A is a tuple with the following components.
- $\mathsf{locs}(A)$ is a finite set of locations,
- $\mathsf{iloc}(A)$ is the unique initial location from $\mathsf{locs}(A)$,
- $\mathsf{acts}(A)$ is a finite set of actions disjoint from $\mathbf{R}_0^+$,
- $\mathsf{clocks}(A)$ is a finite set of clocks,
- $\mathsf{invs}(A) : \mathsf{locs}(A) \rightarrow \mathsf{CC}(\mathsf{clocks}(A))$ maps each location to an invariant for that location such that the initial clock valuation satisfies the invariant of the initial location (i.e., $\mathsf{ICV}(\mathsf{clocks}(A)) \models \mathsf{invs}(A)(\mathsf{iloc}(A))$),
- $\mathsf{edges}(A) \subseteq \mathsf{locs}(A) \times \mathsf{acts}(A) \times \mathsf{CC}(\mathsf{clocks}(A)) \times \mathsf{DIPD}(2^{\mathsf{clocks}(A)} \times \mathsf{locs}(A))$ is a finite set of IPTA edges of the form $(\ell_1, a, \psi, (\mu_1, \mu_2))$ where ℓ_1 is the source location, a is an action, ψ is a guard, and (μ_1, μ_2) is a DIPD mapping pairs (Res, ℓ_2) of clocks to be reset and target locations to probability intervals,
- $\mathsf{aps}(A)$ is a finite set of APs, and
- $\mathsf{lab}(A) : \mathsf{locs}(A) \rightarrow 2^{\mathsf{aps}(A)}$ maps each location to a set of APs.

Moreover, we define PTA, TA, Interval Probabilistic Automaton (IPA), and PA as IPTA restrictions as follows.
- A is a *PTA* if for all $(\ell, a, \psi, (\mu_1, \mu_2)) \in \mathsf{edges}(A)$: $\langle (\mu_1, \mu_2) \rangle$ is a singleton.
- A is a *TA* if for all $(\ell, a, \psi, (\mu_1, \mu_2)) \in \mathsf{edges}(A)$: $\mathsf{supp}((\mu_1, \mu_2))$ is a singleton.
- A is an IPA if $\mathsf{clocks}(A)$ is empty.
- A is a PA if it is an IPA and a PTA.

Consider the following IPTA example of a server communicating with clients.

Example 1 (Server as IPTA)**.** See Figure 2.4 for two IPTA characterizing the behavior of a server. The required significant adaptation of the IPTA from Figure 2.4a to the IPTA Figure 2.4b demonstrates that IPTA (as for finite automata) are not able to count occurrences e.g. using additional variables. Note that some unfolding-like adaptations may be specified using parallel composition of IPTA as in [6, Def. 3.11, p. 22, Chapter 4, pp. 25–].

The semantics of an IPTA is given in terms of the induced PTS, which defines timed probabilistic paths as expected. The states of the induced PTS are pairs of locations and clock valuations. The steps between such states then either (a) nondeterministically advance time or (b) nondeterministically select an enabled IPTA edge, nondeterministically determine a PMF from the given DIPD, and probabilistically determine a reset set and target location based on that PMF.

Definition 2 (Probabilistic Timed System (PTS))**.** A *probabilistic timed system* (*PTS*) P is a tuple with the following components.
- $\mathsf{states}(P)$ is a (possibly infinite) set of states,
- $\mathsf{istate}(P)$ is the unique initial state from $\mathsf{states}(P)$,

- $\mathsf{acts}(P)$ is a finite set of actions disjoint from $\mathbf{R}_0^+$,
- $\mathsf{steps}(P) \subseteq \mathsf{states}(P) \times (\mathsf{acts}(P) \cup \mathbf{R}_0^+) \times \mathsf{DIPD}(\mathsf{states}(P))$ is a (possibly infinite) set of PTS steps of the form $(s_1, a, (\mu, \mu))$ where s_1 is the source state, a is an action or a duration, and where (μ, μ) is a DIPD on the set $\mathsf{states}(P)$ containing all potential target states s_2,
- $\mathsf{aps}(P)$ is a finite set of APs, and
- $\mathsf{lab}(P) : \mathsf{states}(P) \rightarrow 2^{\mathsf{aps}(P)}$ maps each state to a set of APs.

We now define the PTS induced by a given IPTA along the lines of our explanations from above. Note that we present an indirect approach in Appendix A using IPTSs as an intermediate semantic model.

Definition 3 (PTS Induced by IPTA)**.** Every IPTA A induces a unique *probabilistic timed system* (*PTS*) $\mathsf{IPTAtoPTS}(A) = P$ consisting of the following components.

- $\mathsf{states}(P) = \{(\ell, v) \in \mathsf{locs}(A) \times \mathsf{CV}(\mathsf{clocks}(A)) \mid v \models \mathsf{invs}(A)(\ell)\}$ contains as PTS states pairs of locations and clock valuations satisfying the location's invariant,
- $\mathsf{istate}(P) = (\mathsf{iloc}(A), \mathsf{ICV}(\mathsf{clocks}(A)))$ is the unique initial state from $\mathsf{states}(P)$,
- $\mathsf{acts}(P) = \mathsf{acts}(A)$ is the same set of actions,
- $\mathsf{steps}(P) \subseteq \mathsf{states}(P) \times (\mathsf{acts}(P) \cup \mathbf{R}_0^+) \times \mathsf{DIPD}(\mathsf{states}(P))$ is the set of PTS steps where $((\ell, v), a, (\mu, \mu)) \in \mathsf{steps}(P)$, if one of the two following cases applies.
  - Timed step:
    - $a \in \mathbf{R}_0^+$ is a duration,
    - $(\ell, v + t') \in \mathsf{states}(P)$ for all $t' \in [0, a]$, i.e., the invariant of ℓ is also satisfied for all intermediate time points t', and
    - $\mu(\ell, v + a) = 1$ identifies the unique PTS target state $(\ell, v + a)$.
  - Discrete step:
    - $a \in \mathsf{acts}(A)$ is an action of the IPTA,
    - $(\ell, a, \psi, (\mu_1', \mu_2')) \in \mathsf{edges}(A)$ is an edge of the IPTA,
    - $v \models \psi$, i.e., the guard ψ of the edge is satisfied by the valuation v,
    - $\mu_i(\ell', v') = \sum \{\!|\mu_i'(X, \ell') \mid X \subseteq \mathsf{clocks}(A), v' = v[X := 0]|\!\}$ for $i \in \{1, 2\}$ is the DIPD on the PTS target states (ℓ', v'), and
    - $\mu \in \langle(\mu_1, \mu_2)\rangle$ is a PMF from the semantics of the DIPD (μ_1, μ_2).
- $\mathsf{aps}(P) = \mathsf{aps}(A)$ is the same set of APs, and
- $\mathsf{lab}(P)(\ell, v) = \mathsf{lab}(A)(\ell)$ labels states in P according to the location labeling of the IPTA.

For the special case of PTA, we will use in chapter 5 the Prism model checker [7] to solve the following analysis problems defined for the induced PTSs. Intuitively, these analysis problems ask for the probability with which states labeled with a given AP can be reached (possibly within a given time bound). However, the probability to be computed depends on how the nondeterminism in the PTS is resolved. Technically, this nondeterminism is resolved using an adversary, which selects for the finite path constructed so far the next timed/discrete step where the target state of a discrete step is subject to an additional probabilistic choice. The probability values obtained for all such adversaries result in a unique lower and a unique upper bound. These unique lower and upper bounds intuitively correspond to the worst-case or the best-case probabilities depending on whether reaching a state labeled with the given AP

is desirable. The worst-case and the best-case probabilities can then be computed using Prism.

Definition 4 (Min/Max Probabilistic Timed Reachability Problems)**.** Evaluate the expression $\mathcal{P}_{op=?}(\mathsf{F}_{\sim c}\ ap)$ for a PTS P with $op \in \{\mathsf{min}, \mathsf{max}\}$, $\sim \in \{\leq, <\}$, $c \in \mathbf{N} \cup \{\infty\}$, and $ap \in \mathsf{aps}(P)$ to obtain the infimal/supremal probability (depending on op) over all adversaries to reach some state in P labeled with ap within $t \sim c$ time units.

For example, for the PTS $P = \mathsf{IPTAtoPTS}(A_1)$ induced by the IPTA A_1 from Figure 2.3, (a) $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}_{\leq 5}\ \mathit{done})$ is evaluated to probability $0.96 = 0.8 + 0.2 \times 0.8$ since the probability maximizing adversary would enable two discrete steps e.g. at time points 2 and 4 with the maximal probability of 0.8 to reach ℓ_1 in each case and (b) $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}_{\leq 5}\ \mathit{done})$ is evaluated to probability 0.7 since the probability minimizing adversary would enable only one discrete step (e.g. at time point 5) where the minimal probability to reach ℓ_1 would be 0.7.

Formally, the semantics of a PTS is now given in terms of its paths (cf. [10]).

Definition 5 (Semantics of PTS)**.** A PTS P induces the set $\mathsf{Paths}(P, s)$ of non-empty finite and infinite paths[1] π starting in $s \in \mathsf{states}(P)$ where each step (s_1, a, μ, s_2) in that path satisfies $(s_1, a, (\mu, \mu)) \in \mathsf{steps}(P)$ and $\mu(s_2) > 0$.
Moreover, we define the following notions.

- $\mathsf{PathsFinI}(P)$ denotes the finite paths starting in the initial state $\mathsf{istate}(P)$.
- $\mathsf{last}(\pi)$ denotes the last state of a finite path.
- $\mathsf{first}(\pi)$ denotes the first state of a path.
- $\mathsf{state}(\pi, t)$ denotes the state at time t in the path π.
- $\mathsf{dur}(\pi, i) = \sum \{\!|a \in \mathbf{R}_0^+ \mid 0 \leq j < i, \pi_j = (s_1, a, \mu, s_2)|\!\}$ is the duration of π up to reaching the state at index i.
- $(s_1, i, \delta) \in \mathsf{states}(P) \times \mathbf{N} \times \mathbf{R}_0^+$ is a *position of* π, written $(s_1, i, \delta) \in \mathsf{pos}(\pi)$, if[2] $\pi_i = (s_1, a, \mu, s_2)$ and ($\delta = 0$ and $a \in \mathsf{acts}(P)$) or ($\delta < a$ and $a \in \mathbf{R}_0^+$) or ($\delta = a = 0$ and $a \in \mathbf{R}_0^+$).
- $(s_i, i, \delta_i) < (s_j, j, \delta_j)$ for two positions of π, if $i < j$ or ($i = j$ and $\delta_i < \delta_j$).
- A function $A : \mathsf{PathsFinI}(P) \rightarrow \mathsf{steps}(P)$ is a simple *adversary of* P resolving all nondeterminism but not probabilism[3], written $A \in \mathsf{AdvS}(P)$, if $\pi \in \mathsf{PathsFinI}(P)$ and $A(\pi) = (s, a, (\mu_1, \mu_2))$ implies $s = \mathsf{last}(\pi)$ and $\mu_1 = \mu_2$.
- For a simple adversary A, a path $\pi \in \mathsf{Paths}(P, s)$ is compatible with A, written $\pi \in \mathsf{Paths}(P, A, s)$ if for all $i \in \mathbf{N}$ it holds that if π' is the prefix of π of length i, $\mathsf{last}(\pi') = s_1$, and $\pi_i = (s_1, a, \mu, s_2)$, then $A(\pi') = (s_1, a, (\mu, \mu))$.

[1] Intuitively, a path is a resolution of all nondeterminism and all probabilism of the PTS.

[2] Note that we require $\delta < a$ when $a \neq 0$ since the PTA is not really in state s_1 then anymore.

[3] If there is a path $\pi \in \mathsf{PathsFinI}(P)$ ending in state $s = \mathsf{last}(\pi)$ such that there is no step $(s, a, \mu) \in \mathsf{steps}(P)$, then there is no adversary for P. In a static analysis, the possible valuations v of reachable states $s = (\ell, v)$ are unknown and therefore static analysis is in general unable to decide whether a location ℓ may be part of a deadlocked state $s = (\ell, v)$. Examples of threatening locations are locations ℓ with an invariant such as $x_0 \leq 20$ that eventually prohibit timed steps but without exiting edges or with only exiting edges with guards such as $x_0 > 30$ that cannot be satisfied due to the invariant of ℓ. Similarly to the divergence assumption for infinite paths, we hereby rule out PTSs with finite paths that cannot be extended to infinite time-diverging paths.

- If $A \in \mathsf{AdvS}(P)$ and $s \in \mathsf{states}(P)$, then $\mathsf{p}(P, A, s) : \mathsf{Paths}(P, A, s) \longrightarrow [0, 1]$ is a PMF with $\mathsf{p}(P, A, s)(\pi) = \prod_{i \in \mathbf{N}, \pi_i = (s_1, a, \mu, s_2)} \mu(s_2)$.
- Also, $\mathsf{p}(P, A, s)$ induces the probability measure $\mathsf{Prob}(P, A, s) : 2^{\mathsf{Paths}(P,A,s)} \longrightarrow [0, 1]$ where $\mathsf{Prob}(P, A, s)(PS) = \sum \{\!|\mathsf{p}(P, A, s)(\pi) \mid \pi \in PS|\!\}$.
- (Divergent) adversaries $A \in \mathsf{AdvD}(P)$ are simple adversaries from $\mathsf{AdvS}(P)$ for which $\mathsf{Prob}(P, A, \mathsf{istate}(P))(\{\pi \in \mathsf{Paths}(P, A, \mathsf{istate}(P)) \mid \mathsf{diverges}(\pi)\}) = 1$ where $\mathsf{diverges}(\pi)$ is satisfied for an infinite path if $\forall t \in \mathbf{R}_0^+.\ \exists i.\ \mathsf{dur}(\pi, i) > t$.
- If ap is an AP of the PTS P, $c \in \mathbf{N} \cup \{\infty\}$ is a time bound, $\sim \in \{<, \leq\}$ is a comparison operation for the time bound, and $R = \{\mathsf{Prob}(P, A, \mathsf{istate}(P))(\{\pi \in \mathsf{Paths}(P, A, \mathsf{istate}(P)) \mid \exists t \sim c.\ ap \in \mathsf{lab}(P)(\mathsf{state}(\pi, t))\}) \mid A \in \mathsf{AdvD}(P)\}$, then the PTS P results in
  - $\mathsf{sup}(R)$ for $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}_{\sim c}\ ap)$, written $\langle\!\langle \mathcal{P}_{\mathsf{max}=?}(\mathsf{F}_{\sim c}\ ap), P \rangle\!\rangle = \mathsf{sup}(R)$, and
  - $\mathsf{inf}(R)$ for $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}_{\sim c}\ ap)$, written $\langle\!\langle \mathcal{P}_{\mathsf{min}=?}(\mathsf{F}_{\sim c}\ ap), P \rangle\!\rangle = \mathsf{inf}(R)$.

  This item is extended in Definition 17 to more general properties.

The notion of nondeterminism for IPTA can be defined in two ways. According to the presentation above, an adversary resolves the nondeterminism by selecting a unique step to be applied. Hence, an IPTA would be nondeterministic if there would be two viable steps. However, since there are in general an uncountable number of timed steps enabled, this definition would be meaningless. Instead, we define an IPTA to be deterministic if whenever two steps are enabled, at least one of them must be a timed step. That is, in the induced PTS P, every reachable state s and every two steps $(s, a, (\mu_1, \mu_2)), (s, a', (\mu_1', \mu_2')) \in \mathsf{steps}(P)$ with $(s, a, (\mu_1, \mu_2)) \neq (s, a', (\mu_1', \mu_2'))$ satisfy that $a \in \mathbf{R}_0^+$ or $a' \in \mathbf{R}_0^+$. This can be justified because if there are two timed steps of duration δ_1 and δ_2 with $\delta_1 < \delta_2$ to states s_1 and s_2, then there is also a timed step of duration $\delta_2 - \delta_1$ from s_1 to s_2.

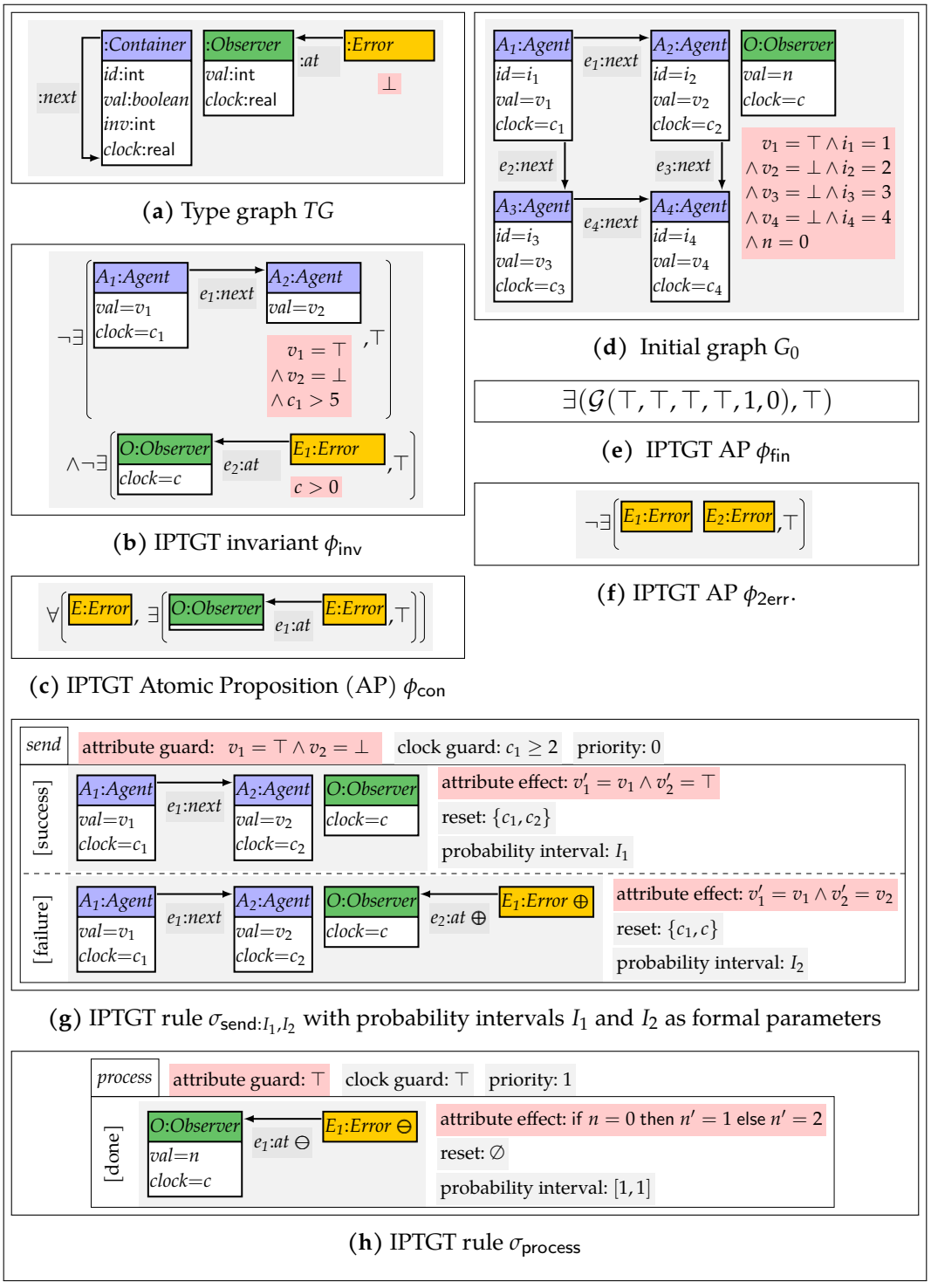
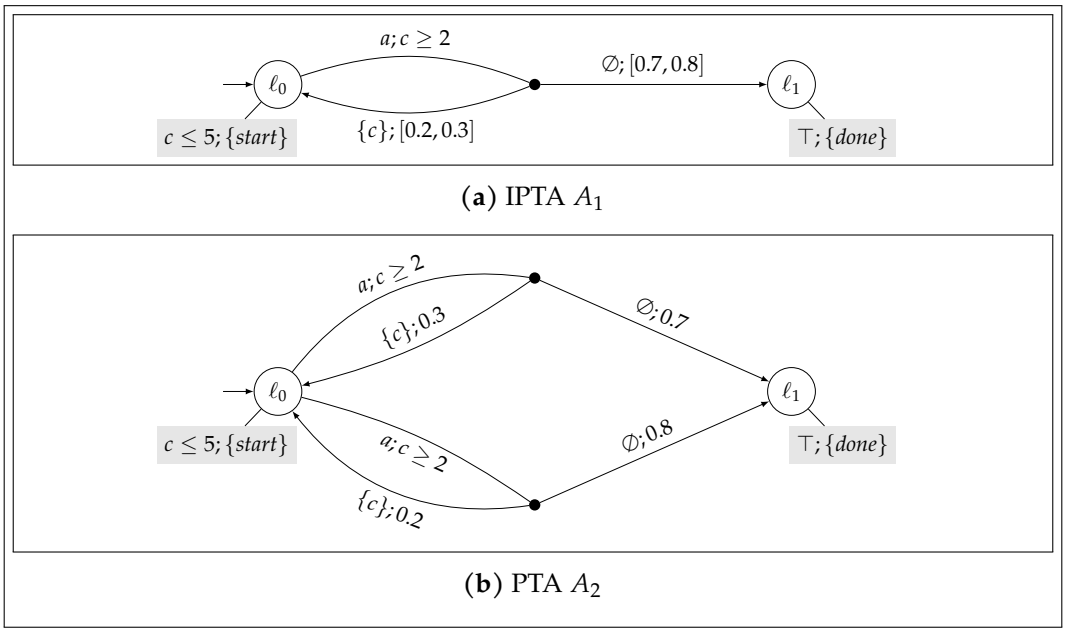


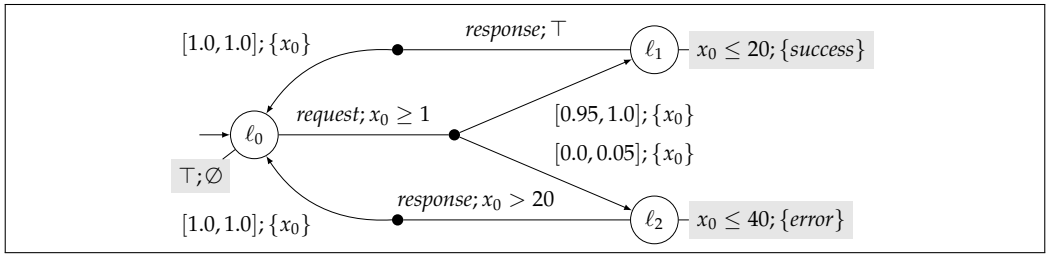
Figure 2.2: Elements of the IPTGTS for the running example



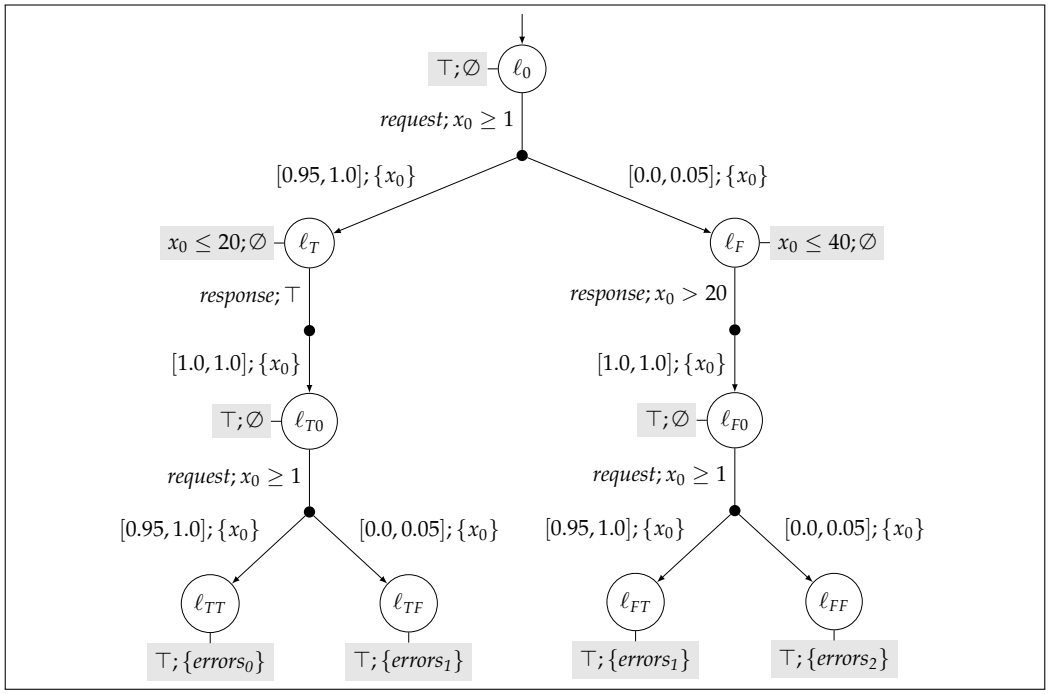
(**a**) IPTA A_1

(**b**) PTA A_2

Figure 2.3: IPTA A_1 and PTA A_2 induced by it



(**a**) A visualization of an IPTA representing a server provides a response to a request within 20 time units with a probability of 0.95 to 1.0. With a probability of 0.0 to 0.05 the server requires up to 40 time units. The clock x_0 is used to measure durations: in ℓ_0 the server blocks requests for up to 1 time unit, in ℓ_1 the server waits for 0 to 20 time units before sending the response, and in ℓ_2 the server waits for 20 to 40 time units before sending the response.



(**b**) A visualization of an IPTA obtained from the IPTA from Figure 2.4a. In this adaptation, precisely two requests are sent to the server using additional locations. We are able to count the number of times where the server fails to send the response within 20 time units. The sending of the second response has been omitted here because it is not relevant to the counting.

Figure 2.4: Visualization for two IPTA

3 Interval Probabilistic Timed GTSs

In this section, we introduce the new formalism of IPTGTSs, which allows for the modeling and analysis of systems exhibiting structure dynamics, timed behavior, and interval probabilistic behavior. IPTGTSs extend PTGTSs, which are a combination of PGTSs and TGTSs, by allowing interval probabilities as in IPTA instead of precise probabilities as in PTA.

As usual, we assume that all graphs in IPTGTSs are typed over some fixed type graph TG. Moreover, we denote the set of all variables of sort real that represent clocks of a given graph G by $\mathsf{C}(G)$ (for the graph G_0 from Figure 2.2d, $\mathsf{C}(G_0) = \{c_1, c_2, c_3, c_4, c\}$). Note that, in this paper, we employ variables of the symbolic graphs to represent clocks rather than clock nodes as in [13] to simplify the technical presentation.

For our running example, introduced in chapter 1, the type graph TG is given in Figure 2.2a and the initial graph G_0 is given in Figure 2.2d. In the following, we use the abbreviation of the form $\mathcal{G}(v_1, v_2, v_3, v_4, n, e)$ for all reachable graphs where v_1, v_2, v_3, v_4, and n correspond to the values of variables and e is the number of *Error* nodes connected to the *Observer* node. Using this abbreviation, we denote the initial graph G_0 by $\mathcal{G}(\top, \bot, \bot, \bot, 0, 0)$.

An IPTGT rule σ contains a set of GT rules $\mathsf{rules}(\sigma)$ with a common left-hand side graph $\mathsf{lhs}(\sigma)$, which is matched into the graph under transformation. A clock constraint over the clocks from $\mathsf{lhs}(\sigma)$ is used as a guard and is evaluated w.r.t. a considered match. A DIPD assigns a non-empty probability interval to each GT rule ρ. Each GT rule ρ is equipped with a set of clocks to be reset ranging over the clocks from the right-hand side graph $\mathsf{rhs}(\rho)$ of ρ.

Definition 6 (IPTGT Rule)**.** An *interval probabilistic timed graph transformation rule* (*IPTGT rule*) σ is a tuple with the following components.

- $\mathsf{lhs}(\sigma)$ is a common left-hand side graph,
- $\mathsf{rules}(\sigma)$ is a finite set of GT rules ρ with $\mathsf{lhs}(\rho) = \mathsf{lhs}(\sigma)$ where $\mathsf{lhs}(\rho)$ is the left-hand side graph of the GT rule ρ,
- $\mathsf{guard}(\sigma) \in \mathsf{CC}(\mathsf{C}(\mathsf{lhs}(\sigma)))$ is a guard defined as a clock constraint over the clocks from the left-hand side graph $\mathsf{lhs}(\sigma)$,
- $\mathsf{dipd}(\sigma) \in \mathsf{DIPD}(\mathsf{rules}(\sigma))$ is a DIPD on $\mathsf{rules}(\sigma)$ with $\mathsf{supp}(\mathsf{dipd}(\sigma)) = \mathsf{rules}(\sigma)$,
- $\mathsf{reset}(\sigma)(\rho) \subseteq \mathsf{C}(\mathsf{rhs}(\rho))$ identifies the clocks to be reset for each $\rho \in \mathsf{rules}(\sigma)$, and
- $\mathsf{prio}(\sigma) \in \mathbf{N}$ is the priority assigned to σ.

For our running example, consider the two IPTGT rules $\sigma_{\mathsf{send}:I_1,I_2}$[1] and $\sigma_{\mathsf{process}}$ in Figure 2.2g and Figure 2.2h, respectively. The two GT rules $\rho_{\mathsf{send:success}}$ and $\rho_{\mathsf{send:failure}}$ of $\sigma_{\mathsf{send}:I_1,I_2}$ and the single GT rule $\rho_{\mathsf{process:done}}$ of $\sigma_{\mathsf{process}}$ are given in integrated notation where graph elements to be added/removed are marked with $\oplus/\ominus$. To limit rule application, each IPTGT rule has an *attribute guard* on the current attribute values, a *clock guard* on the current clock values, and a *priority*. Also, each of the underlying GT rules has an AC describing an attribute modification (called *attribute effect*), a *reset* set of clocks to be reset after rule application, and a *probability interval*. Intuitively, the IPTGT rule $\sigma_{\mathsf{send}:I_1,I_2}$ is used to attempt the sending of the Boolean value true ($\top$) from one agent to another agent with Boolean value $\bot$ when the clock value of the sending agent is at least 2. If sending succeeds, the receiving agent adopts the Boolean value $\top$ and may then send that value as well. If sending fails, an error is created and connected to the observer. The IPTGT rule $\sigma_{\mathsf{process}}$ (which has a higher priority than $\sigma_{\mathsf{send}:I_1,I_2}$) is used to allow the observer to process (and delete) pending errors counting processed errors up to a maximal number of 2.

An IPTGT invariant ϕ is a nested graph condition over the empty graph $\emptyset$. IPTGT invariants are used to rule out invalid potential IPTGT configurations. Potential IPTGT configurations (G, v) are given by a finite graph G and a clock valuation $v \in \mathsf{CV}(\mathsf{C}(G))$ on its clocks. For our running example, the IPTGT invariant ϕ_{inv} in Figure 2.2b states that (a) a sending agent must send its Boolean value $\top$ after waiting not longer than 5 time units and (b) an observer with a pending error must process that error urgently. Note that the clocks of the agents and the observer are reset to 0 whenever they send their Boolean value or process an error using the two IPTGT rules. An IPTGT invariant ϕ is satisfied by a potential IPTGT configuration (G, v), written $(G, v) \models \phi$, if $v \models \exists V.\ \mathsf{ac}(G) \wedge \gamma$ where V is the set of all non-clock variables of G, $\mathsf{ac}(G)$ is the AC of G, and γ is an attribute constraint on the variables of G obtained by evaluating ϕ for G. For our running example, the potential IPTGT configuration $(\mathcal{G}(\top, \bot, \bot, \bot, 0, 0), v)$ where $v = \mathsf{ICV}(\{c_1, c_2, c_3, c_4\})$ satisfies the PTGT invariant ϕ_{inv} since v satisfies the derived AC equivalent to $c_1 \leq 5$ where V contains all *id* and *val* variables, $\mathsf{ac}(G)$ is the AC given in Figure 2.2d, and γ states for each of the four clocks c_1–c_4 that their value must be less equal 5 when the corresponding *val* attribute equals $\top$. This reasoning is covered more technically in the following example.

Example 2 (Evaluation of IPTGT Invariant)**.** We give an example on how IPTGT invariants and IPTGT APs are evaluated for a given (potential) IPTGT configuration. We obtain the clock constraint $c_1 \leq 5$ for the initial graph $G_0 = \mathcal{G}(\top, \bot, \bot, \bot, 0, 0)$ from Figure 2.2d and the IPTGT invariant ϕ_{inv} from Figure 2.2b as follows.

- The non-clock variables V of G_0 are given by $\{v_1, v_2, v_3, v_4, n, i_1, i_2, i_3, i_4\}$.
- The AC $\mathsf{ac}(G)$ of G_0 is $v_1 = \top \wedge v_2 = \bot \wedge v_3 = \bot \wedge v_4 = \bot \wedge n = 0 \wedge i_1 = 1 \wedge i_2 = 2 \wedge i_3 = 3 \wedge i_4 = 4$.

[1] In our evaluation in chapter 5, we consider different instantiations of the two probability intervals I_1 and I_2.

- The evaluation of ϕ_{inv} for G_0 results in $\neg(v_1 = \top \wedge v_2 = \bot \wedge c_1 > 5) \wedge \neg(v_1 = \top \wedge v_3 = \bot \wedge c_1 > 5) \wedge \neg(v_2 = \top \wedge v_4 = \bot \wedge c_2 > 5) \wedge \neg(v_3 = \top \wedge v_4 = \bot \wedge c_3 > 5)$ capturing all four possible structural matches of the graph from the invariant into G_0 each time obtaining an AC describing when that structural match would satisfy the requirements on the attributes.

Using the information gathered in these three steps, we construct and simplify the following AC to $c_1 \leq 5$ stating that only clock c_1 has an upper bound in G_0.

$$\begin{aligned}
&\exists\{v_1, v_2, v_3, v_4, n, i_1, i_2, i_3, i_4\}.\\
&(v_1 = \top \wedge v_2 = \bot \wedge v_3 = \bot \wedge v_4 = \bot \wedge n = 0\\
&\wedge i_1 = 1 \wedge i_2 = 2 \wedge i_3 = 3 \wedge i_4 = 4)\\
&\wedge \neg(v_1 = \top \wedge v_2 = \bot \wedge c_1 > 5) \wedge \neg(v_1 = \top \wedge v_3 = \bot \wedge c_1 > 5)\\
&\wedge \neg(v_2 = \top \wedge v_4 = \bot \wedge c_2 > 5) \wedge \neg(v_3 = \top \wedge v_4 = \bot \wedge c_3 > 5)\\
\equiv\ & c_1 \leq 5
\end{aligned}$$

Similarly, an IPTGT AP (IPTGT AP) ϕ is a nested graph condition over the empty graph $\emptyset$ labeling a (potential) IPTGT configuration (G, v) if G satisfies ϕ, written $G \models \phi$. Note that IPTGT APs may not depend on the clock valuation v as for the labeling in IPTA. For our running example, we employ the IPTGT APs ϕ_{fin}, ϕ_{con}, and $\phi_{2\mathsf{err}}$ from Figure 2.2. ϕ_{fin} (given using the previously introduced abbreviation) checks whether the value $\top$ has been successfully adopted by all agents, precisely one error was processed by the observer, and no errors are pending. ϕ_{con} checks whether all errors are connected to some observers and $\phi_{2\mathsf{err}}$ checks whether no two errors are present.

We now define IPTGTSs based on the notions introduced above for a fixed type graph. For our running example, the components of the considered IPTGTS are given in Figure 2.2.

Definition 7 (IPTGTS)**.** An *interval probabilistic timed graph transformation system* (*IPTGTS*) S is a tuple with the following components.

- $\mathsf{iG}(S)$ is a finite initial graph,
- $\mathsf{rules}(S)$ is a finite set of IPTGT rules,
- $\mathsf{invs}(S)$ is a finite set of IPTGT invariants, which are all satisfied for the initial graph and the initial clock valuation $(\mathsf{iG}(S), \mathsf{ICV}(\mathsf{C}(\mathsf{iG}(S))))$, and
- $\mathsf{aps}(S)$ is a finite set of IPTGT APs.

Moreover, we define the following notions.

- A potential IPTGT configuration (G, v) given by a finite graph G and a clock valuation $v \in \mathsf{CV}(\mathsf{C}(G))$ is an *IPTGT configuration* of S, written $(G, v) \in \mathsf{Confs}(S)$, if (G, v) satisfies all IPTGT invariants of S, i.e., $(G, v) \models \phi$ for each $\phi \in \mathsf{invs}(S)$.
- Two given IPTGT configurations (G_1, v_1) and (G_2, v_2) are *equivalent*, written $(G_1, v_1) \equiv (G_2, v_2)$, if there is some isomorphism $m : G_1 \xrightarrow{\sim} G_2$ such that $v_2 \circ m = v_1$. The equivalence relation $\equiv$ also induces equivalence classes denoted by $[(G_1, v_1)]_{\equiv}$.

The semantics of IPTGTSs is defined below in terms of an induced PTS, for which we first define a step relation. As for IPTA, IPTGTSs can execute, on the one hand, *timed steps* advancing all clocks while respecting the invariants and, on the other hand,

discrete steps by applying some IPTGT rule where the IPTGT rule, the match, and the Probability Mass Function (PMF) of the DIPD are chosen nondeterministically and the GT rule to be used is chosen probabilistically. Moreover, for the discrete steps, we ensure that (a) the guard of the IPTGT rule is satisfied by the current clock valuation and match, (b) no discrete step using an IPTGT rule with higher priority can be applied, and (c) all GT rules of the IPTGT rule are applicable using the same match. Then, considering a GT rule ρ of the IPTGT rule σ, we define a discrete step based on the corresponding GT step and ensure that the clock valuation is adapted as expected also enforcing the clock resets specified in the IPTGT rule.

Definition 8 (IPTGT Step)**.** An IPTGTS S defines the following two kinds of steps.

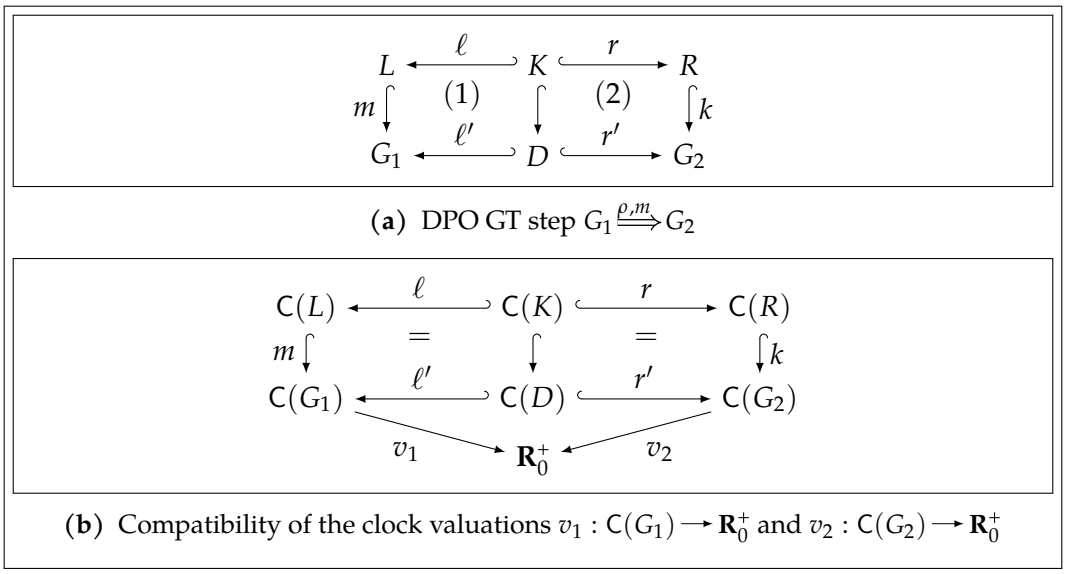
- TIMED STEP: $(G, v) \xrightarrow{\delta} (G, v + \delta)$, if
  - $\delta \in \mathbf{R}_0^+$ is a duration and
  - $(G, v + \delta') \in \mathsf{Confs}(S)$ for each $\delta' \in [0, \delta]$, i.e., the IPTGT invariants are also satisfied for all intermediate time points.
- DISCRETE STEP: $(G_1, v_1) \xrightarrow{\sigma,\rho,m} (G_2, v_2)$, if
  - $\sigma \in \mathsf{rules}(S)$ is an IPTGT rule,
  - $m : \mathsf{lhs}(\sigma) \hookrightarrow G_1$ is a match,
  - $v_1 \models \mathsf{guard}(\sigma)$, i.e., the guard of the IPTGT rule σ is satisfied by the given valuation v_1,
  - $\rho \in \mathsf{rules}(\sigma)$ is a GT rule of σ,
  - no IPTGT rule σ' with higher priority is applicable, i.e., there are no G_2', v_2', σ', ρ', and m' such that $(G_1, v_1) \xrightarrow{\sigma',\rho',m'} (G_2', v_2')$ and $\mathsf{prio}(\sigma') > \mathsf{prio}(\sigma)$,
  - the GT rule ρ is applicable and results in the IPTGT configuration (G_2, v_2), i.e., $(G_1, v_1) \xrightarrow{\sigma,\rho,m} (G_2, v_2)$,
  - every GT rule of σ is applicable for the match m, i.e., for all $\rho' \in \mathsf{rules}(\sigma)$ there are G_2' and v_2' such that $(G_1, v) \xrightarrow{\sigma,\rho',m} (G_2', v_2')$,

  where $(G_1, v_1) \xrightarrow{\sigma,\rho,m} (G_2, v_2)$, if
  - $(G_1, v_1), (G_2, v_2) \in \mathsf{Confs}(S)$,
  - $\rho = (\ell : K \hookrightarrow L, r : K \hookrightarrow R, \gamma)$ is a GT rule[2],
  - $G_1 \overset{\rho,m}{\Longrightarrow} G_2$ is the DPO GT step from Figure 3.1a,
  - v_2 is obtained from v_1 by preserving values of preserved clocks unless they are reset to 0 and by assigning the clock value 0 to all clocks created by the GT step, i.e., for each $c \in \mathsf{C}(G_2)$ we define (a) $v_2(c) = x$ when $c \notin k(\mathsf{reset}(\sigma)(\rho))$ and there is some $c' \in \mathsf{C}(D)$ with $r'(c') = c$ such that there is some $c'' \in \mathsf{C}(G_1)$ with $\ell'(c') = c''$ such that $v_1(c'') = x$ and (b) $v_2(c) = 0$ otherwise.

In the following example, we provide a short sequence of IPGT steps for our running example.

Example 3 (Steps for Running Example)**.** A sequence of IPTGT steps (timed steps and discrete steps) for the running example (cf. Figure 2.2). The sequence starts from the initial graph from Figure 2.2d and the initial clock valuation. We omit a full presentation of the employed matches m_1, m_2, and m_3. The matches m_1 and m_3 match

[2]We omit here the handling of attribute modifications given by γ for brevity.



(a) DPO GT step $G_1 \overset{\rho,m}{\Longrightarrow} G_2$

(b) Compatibility of the clock valuations $v_1 : \mathsf{C}(G_1) \rightarrow \mathbf{R}_0^+$ and $v_2 : \mathsf{C}(G_2) \rightarrow \mathbf{R}_0^+$

Figure 3.1: Visualizations for Definition 8

the *Container* nodes C_1 and C_2 with ids 1 and 2 in the graph under transformation. The match m_2 matches the *Error* node to be removed.

$$
\begin{aligned}
& (\mathcal{G}(\top,\bot,\bot,\bot,0,0), \{c_1 \mapsto 0, c_2 \mapsto 0, c_3 \mapsto 0, c_4 \mapsto 0, c \mapsto 0\}) \\
\xrightarrow{2} \quad & (\mathcal{G}(\top,\bot,\bot,\bot,0,0), \{c_1 \mapsto 2, c_2 \mapsto 2, c_3 \mapsto 2, c_4 \mapsto 2, c \mapsto 2\}) \\
\xrightarrow{\sigma_{\mathsf{send}:[0.3,0.8],[0.2,0.7]}, \rho_{\mathsf{send}:\mathsf{failure}}, m_1} \quad & (\mathcal{G}(\top,\bot,\bot,\bot,0,1), \{c_1 \mapsto 0, c_2 \mapsto 2, c_3 \mapsto 2, c_4 \mapsto 2, c \mapsto 0\}) \\
\xrightarrow{\sigma_{\mathsf{process}}, \rho_{\mathsf{process}:\mathsf{done}}, m_2} \quad & (\mathcal{G}(\top,\bot,\bot,\bot,1,0), \{c_1 \mapsto 0, c_2 \mapsto 2, c_3 \mapsto 2, c_4 \mapsto 2, c \mapsto 0\}) \\
\xrightarrow{2} \quad & (\mathcal{G}(\top,\bot,\bot,\bot,1,0), \{c_1 \mapsto 2, c_2 \mapsto 4, c_3 \mapsto 4, c_4 \mapsto 4, c \mapsto 2\}) \\
\xrightarrow{\sigma_{\mathsf{send}:[0.3,0.8],[0.2,0.7]}, \rho_{\mathsf{send}:\mathsf{success}}, m_3} \quad & (\mathcal{G}(\top,\top,\bot,\bot,1,0), \{c_1 \mapsto 0, c_2 \mapsto 0, c_3 \mapsto 4, c_4 \mapsto 4, c \mapsto 2\})
\end{aligned}
$$

We now define the semantics of IPGTSs in terms of an induced PTS as before for IPTA. Note that, for the following definition, IPTGT steps as well as the notions of IPTGT satisfaction for APs, guards, and invariants are preserved by equivalence of IPTGT configurations. Hence, the choice of a representant from an equivalence class is not important.

Definition 9 (PTS Induced by IPTGTS)**.** Every IPTGTS S induces a unique *PTS* $\mathsf{IPTGTStoPTS}(S) = P$ consisting of the following components.

- $\mathsf{states}(P)$ is given by the smallest set of equivalence classes $[(G,v)]_{\equiv}$ where $(G,v) \in \mathsf{Confs}(S)$ such that $\mathsf{steps}(P)$ below is well-defined and $\mathsf{istate}(P)$ (see the next item) is in $\mathsf{states}(P)$,
- $\mathsf{istate}(P) = [(\mathsf{iG}(S), \mathsf{ICV}(\mathsf{C}(\mathsf{iG}(S))))]_{\equiv}$ is the unique initial state from $\mathsf{states}(P)$ given by the equivalence class containing the initial configuration of S,
- $\mathsf{acts}(P)$ is the smallest set of tuples (σ, m) where $\sigma \in \mathsf{rules}(S)$ and m is a match such that $\mathsf{steps}(P)$ below is well-defined,
- $([(G,v)]_{\equiv}, a, (\mu,\mu)) \in \mathsf{steps}(P)$, if one of the two following cases applies.
  - Timed step:
    - $a \in \mathbf{R}_0^+$ is a duration,
    - $(G,v) \xrightarrow{a} (G, v+a)$ is a timed step of S, and

  - ▷ $\mu([G, v + a]_{\equiv}) = 1$ identifies the unique target state $[(G, v + a)]_{\equiv}$.
  - ○ Discrete step:
    - ▷ $a = (\sigma, m) \in \mathsf{acts}(P)$ is a partial step label,
    - ▷ $(G, v) \xrightarrow{\sigma,\rho,m} (G', v')$ for some $\rho \in \mathsf{rules}(\sigma)$ is a discrete step of S,
    - ▷ $\mathsf{dipd}(\sigma) = (\mu'_1, \mu'_2)$ is the DIPD of σ,
    - ▷ $\mu_i([(\bar{G}, \bar{v})]_{\equiv}) = \sum \{\!|\mu'_i(\rho') \mid \rho' \in \mathsf{rules}(\sigma), (G, v) \xrightarrow{\sigma,\rho',m} \equiv (\bar{G}, \bar{v})|\!\}$ for all $[(\bar{G}, \bar{v})]_{\equiv} \in \mathsf{states}(P)$ and $i \in \{1, 2\}$ is the DIPD on the target states, and
    - ▷ $\mu \in \langle(\mu_1, \mu_2)\rangle$ is a PMF from the semantics of the DIPD (μ_1, μ_2).
- $\mathsf{aps}(P) = \mathsf{aps}(S)$ is the same set of APs, and
- $\mathsf{lab}(P)([(G, v)]_{\equiv}) = \{\phi \in \mathsf{aps}(P) \mid G \models \phi\}$ labels states in P.

By defining the induced PTS of an IPTGTS, we can now consider the PTS analysis problems from Definition 4 also for IPTGTSs.

4 Model Checking Approach

The definition of the induced PTS of an IPTGTS from the previous section does not lead to an implementable analysis algorithm because the set of states of that PTS is (due to the valuations of real-valued clocks) not even countable. To obtain analysis support for the min/max probabilistic timed reachability properties from Definition 4, we now follow the path taken for PTGTSs in [13] and translate a given IPTGTS into its underlying automata-based model preserving its semantics in terms of the induced PTS (see Figure 4.1). As a first step, in section 4.1, we introduce the operation IPTGTStoIPTA translating an IPTGTS into an IPTA. As a second step, in section 4.2, we translate the obtained IPTA into a PTA using the operation IPTAtoPTA. This operation is defined based on the translation procedure from [4], which is shown to preserve the semantics in terms of the analysis problems from Definition 4. For the resulting PTA, the analysis problems from Definition 4 can then be analyzed using the PRISM model checker. To accommodate for IPTGTSs with infinite underlying GT state spaces but finite underlying timed GT state spaces, we present in section 4.3 an online filtering technique using the UPPAAL model checker. Lastly, we briefly discuss the analysis of non-probabilistic properties for IPTGTSs in section 4.4 before revisiting the min/max probabilistic timed reachability properties from Definition 4 in our evaluation in chapter 5.

4.1 Translation of IPTGTS into IPTA

We now present how IPTGTSs can be translated into IPTA using the operation IPTGTStoIPTA. This translation is an adaptation of the translation of PTGTSs into PTA presented in [13].

For a given IPTGTS S, the following four steps describe the basic idea of its translation into the corresponding IPTA. In step 1, the underlying GTS S' of the IPTGTS with the rule set $\cup\{\mathsf{rules}(\sigma) \mid \sigma \in \mathsf{rules}(S)\}$ and the initial graph $\mathsf{iG}(S)$ is determined where no priorities or timing constraints of the IPTGT rules are integrated into the GT rules. In step 2, the GT state space (Q, E) of S' is constructed where Q is the set of all reachable graphs and E contains the corresponding GT steps between these graphs. In step 3, a smallest set of so-called *global clocks* Y from the GT state space (Q, E) is derived where the underlying GT spans from E are used to track such global clocks. Finally, in step 4, the resulting IPTA is obtained from the IPTGTS S and the GT state space (Q, E) where the set Y of global clocks is employed to convert and annotate GT steps from E.

Note the following for the steps 2–4 of the translation. In step 2, the GT state space (Q, E) will often contain additional steps (and also states) that are not permitted in

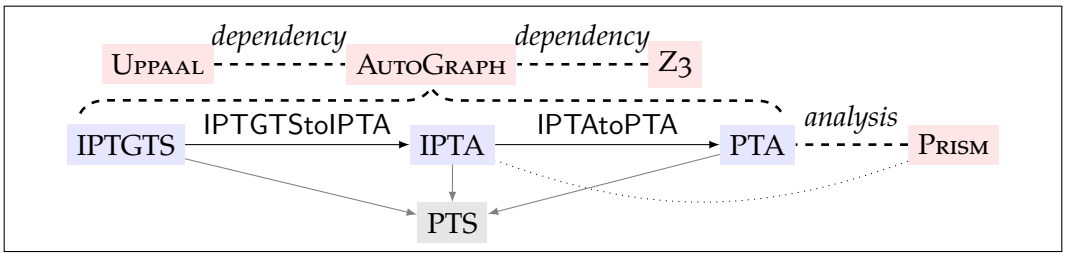


Figure 4.1: Overview of the model checking approach.

the IPTGTS due to (a) priorities and (b) timing constraints (i.e., guards or invariants). Formally, for every discrete step $(G, v) \xrightarrow{\sigma,\rho,m} (G', v')$ of the IPTGTS there is some step $G \overset{\rho,m}{\Longrightarrow} G'$ in the GT state space (Q, E) but not vice versa. Cases where the GT state space (Q, E) is not finite are discussed in section 4.3. Moreover, to ensure in step 3 that Y contains a finite number of clocks, the initial graph should contain all clocks to be used at some point and IPTGT rules should not add further clocks. Also, to prevent that clocks are swapped by isomorphisms during the GT state space construction, we use *id* attributes such as those in *Agent* nodes in our running example. For step 4, relying on the set of global clocks Y, (a) invariants are obtained for each graph in the GT state space and checked for satisfiability, (b) GT steps with common source location and match belonging to one IPTGT rule are grouped into one IPTA edge, and (c) the guard of an IPTA edge is obtained by ensuring that the IPTA edge is not disabled by invariants of its target locations and by also requiring the negation of the guards of IPTA edges starting in the same location but resulting from IPTGT rules with higher priorities.

Definition 10 (IPTA Induced by IPTGTS)**.** Every IPTGTS S induces a unique *IPTA* $\mathsf{IPTGTStoIPTA}(S) = A$ consisting of the following components where (Q, E) is the GT state space of the underlying GTS of S and Y is its set of global clocks.

- $\mathsf{locs}(A) = Q$ contains all graphs of the GT state space,
- $\mathsf{iloc}(A) = \mathsf{iG}(S)$ is the unique initial location,
- $\mathsf{acts}(A)$ is the smallest set of tuples (σ, m) where $\sigma \in \mathsf{rules}(S)$ and m is a match such that $\mathsf{edges}(A)$ below is well-defined,
- $\mathsf{clocks}(A) = Y$ is the set of global clocks,
- $\mathsf{invs}(A)(G) = \psi$ such that $(G, v) \in \mathsf{Confs}(S)$ iff $v \models \psi$,
- $(G, (\sigma, m), \psi, (\mu_1, \mu_2)) \in \mathsf{edges}(A)$, if
  - the guard ψ is a clause of the disjunctive normal form[1] of the constraint $m(\mathsf{guard}(\sigma)) \wedge \psi_{inv} \wedge \neg\psi_{higher}$ where (a) ψ_{inv} is the conjunction of invariants $\mathsf{invs}(A)(G)[R := 0]$ of the target locations adapted to the source location for all $(R, G) \in \mathsf{supp}((\mu_1, \mu_2))$ and (b) ψ_{higher} is the conjunction of the

[1]Note that negation and disjunction are not allowed in guards.

guards ψ'' of all IPTA edges $(G, (\sigma'', m''), \psi'', (\mu_1'', \mu_2'')) \in \mathsf{edges}(A)$ satisfying $\mathsf{prio}(\sigma'') > \mathsf{prio}(\sigma)$[2] and
  - $\mu_i(Y', G') = \sum \{\!|\mu_i'(\rho) \mid \rho \in \mathsf{rules}(\sigma) \wedge Y' = \mathsf{reset}(\sigma)(\rho) \wedge (G, \rho, m, G') \in E \wedge \mathsf{dipd}(\sigma) = (\mu_1', \mu_2')|\!\}$ for $i \in \{1, 2\}$ is the DIPD on the target locations,
- $\mathsf{aps}(A) = \mathsf{aps}(S)$ is the same set of APs, and
- $\mathsf{lab}(A)(G) = \{\phi \in \mathsf{aps}(A) \mid G \models \phi\}$ labels locations in A.

The operation IPTGTStoIPTA preserves the semantics in terms of the induced PTSs, i.e., the IPTGTS and the resulting IPTA induce the same PTS.

Note that we may adapt this translation to handle APs *ap* with nontrivial clock constraints by labelling a fresh state s_{ap} and by adding additional steps from a state s to s_{ap} whenever s is a state that would be labelled by *ap* structurally and by using the clock constraint as a guard for that additional step. In particular, in the view of the available tool support for PTA and IPTA analysis using Prism only allowing for reachability analysis (apart from the digital clocks engine, which uses a different PTA semantics), this adaptation would improve the usability of IPTGTS. However, such an adaptation would also require to adapt the definition of the induced PTS of an IPTGTS. Nevertheless, users of the IPTGTS formalism may just employ the same idea by defining a high priority IPTGT rule with the suitable guard and by adjusting the AP accordingly to match a state once this rule has been just performed.

4.2 Translation of IPTA into PTA

In [4], we have implemented the IPTA model checking algorithm from [21] in Prism. This algorithm takes an IPTA A and an analysis problem of the form $\mathcal{P}_{op=?}(\mathsf{F}_{\leq\infty}\, ap)$ from Definition 4 and computes the resulting probability value.[3] The algorithm operates on a zone-based state space where states of the form $(\ell, \psi) \in \mathsf{locs}(A) \times \mathsf{CC}(\mathsf{clocks}(A))$ represent all states (ℓ, v) of the PTS induced by A satisfying $v \models \psi$. The fixed-point computation performed by the algorithm modifies a probability vector p_i mapping states (ℓ, ψ) to probabilities. Initially, p_0 maps all *target states* (ℓ, ψ) containing locations ℓ that are labeled by the AP *ap* of the considered property to 1 and all other states to 0. In the fixed-point, p_i maps each state (ℓ, ψ) to the probability with which one of the target states is reached. To obtain $p_{i+1}(\ell, \psi)$ in an iteration (a) each IPTA edge used in a step from (ℓ, ψ) with DIPD (μ_1, μ_2) is considered, (b) a PMF $\mu \in \langle(\mu_1, \mu_2)\rangle$ is obtained such that the probability given by $\sum \{\!|\mu(s) \times p_i(s) \mid s \in \mathsf{locs}(A) \times \mathsf{CC}(\mathsf{clocks}(A))|\!\}$ for reaching a target state from the state (ℓ, ψ) using a path where the considered IPTA edge is taken in the first step is maximal/minimal, and (c) $p_{i+1}(\ell, \psi)$ is set to the maximal/minimal value across all IPTA edges considered for (ℓ, ψ). For the IPTA A_1 from Figure 2.3 and the

[2] The dependency between the guards ψ and each ψ'' requires that IPTA edges of A are constructed in descending order of the priorities of the involved IPTGT rules.

[3] As usual, time bounds $\sim c$ (as in Definition 4) are encoded using an additional clock to force a step to a sink location as soon as the time bound is violated.

property $\mathcal{P}_{\mathsf{max=?}}(\mathsf{F}_{\leq\infty}\ \mathit{done})$, we obtain $p_0 = \{(\ell_0, c \leq 5) \mapsto 0, (\ell_1, \top) \mapsto 1\}$ and $p_{i+1} = \{(\ell_0, c \leq 5) \mapsto 0.8 \times p_i(\ell_1, \top) + 0.2 \times p_i(\ell_0, c \leq 5), (\ell_1, \top) \mapsto 1\}$ for $i \in \mathbf{N}$ resulting in $p_i(\ell_0, c \leq 5) = 1$ in the limit $i \to \infty$.

Unfortunately, the described algorithm has not been integrated in the official Prism branch and is only available for the stochastic games engine of Prism. As an alternative approach to obtain model checking support for IPTA, we translate the given IPTA into a PTA (as exemplified in Figure 2.3 where the IPTA A_1 is translated into the PTA A_2) and then apply Prism to the resulting PTA. Intuitively, the PTA is obtained by replacing each IPTA edge e_1 of the IPTA by a set of PTA edges. Thereby, a replacement edge e_2 is obtained from e_1 by replacing its DIPD (μ_1, μ_2) by the DIPD (μ, μ) where $\mu \in \langle(\mu_1, \mu_2)\rangle$ is a PMF that may be obtained for some p_i using the described algorithm. In fact, instead of considering all such possible p_i, it suffices to consider all permutations of the target locations of e_1. The intervals from the DIPD (μ_1, μ_2) are then resolved in the order of the permutation by choosing the maximal[4] probability from the ith interval such that the sum of the first i chosen probabilities plus the sum of the minimal probabilities of the remaining intervals does not exceed 1. For the translation in Figure 2.3, (a) the permutation (ℓ_0, ℓ_1) results in the upper PTA edge where the interval [0.2,0.3] for the first location is resolved by choosing the maximal value 0.3 and by then choosing 0.7 analogously for the second location. (b) the permutation (ℓ_1, ℓ_0) results in the lower PTA edge where the interval [0.7,0.8] for the first location is resolved by choosing the maximal value 0.8 and by then choosing 0.2 analogously for the second location. The described translation, defined by the operation IPTAtoPTA, is correct since the same probabilities are computed for the input IPTA and the output PTA for the analysis problems from Definition 4.

Constructing a PMF for all permutations of intervals for an IPTA edge may result in an exponentially larger PTA. This means that IPTA are exponentially more concise compared to PTA w.r.t. the considered analysis problems (which correspondingly holds for IPTGTSs and PTGTSs). However, when (a) IPTGT rules contain only few GT rules implying a small set of permutations, (b) the permutations result in a small set of PMFs (since different permutations may result in the same PMF), or (c) the model checking efficiency of the PTA or IPTA at hand is dominated by the number of clocks but not by the number of PTA edges, employing the translation via the operation IPTAtoPTA can be as efficient as the IPTA model checking algorithm from [4, 21].

4.3 Analysis of Timed Reachability

As a plug-in procedure, we now discuss our on-the-fly adaptation of the operation IPTGTStoIPTA presented in section 4.1. The goal of this adaptation is to allow for the generation of (finite) IPTA when the intermediate GT state space is infinite while the timed GT state space is finite. To achieve this goal, we employ Uppaal to check

[4] As all permutations are considered, we can also choose the minimal probability here.

whether steps constructed in the GT state space are enabled when considering the timed behavior specified by guards, invariants, and resets in the timed GT state space. For this purpose, we construct a sequence of fragments of the priority-free GT state space where all GT steps are timed reachable (i.e., each GT step of the fragment occurs in some timed path when considering all timing constraints of the IPTGTS). We start the procedure with the GT state space fragment only containing the initial graph of the IPTGTS and then, as a first step, we construct either at most n further GT steps overall or at most n further GT steps from each unfinished state. As a second step, if GT steps have been added, we construct using the operation IPTGTStoTA[5] a TA for the current GT state space fragment. As a third step, we determine those most recently added GT steps that are not timed reachable using Uppaal on the constructed TA removing them from the current GT state space fragment before repeating the described three steps.

Note that a GT step is not guaranteed to be timed reachable due to e.g. the guard that is created for it in the corresponding TA edge even when its source and target graphs are timed reachable. Hence, in the operation IPTGTStoTA, we split each most recently added GT step $G_1 \overset{\rho,m}{\Longrightarrow} G_2$ into two edges in the resulting TA. The first edge implements the GT step but has a fresh target state G' from which the second edge is taken urgently (using an additional single fresh clock employed globally in that translation) leading to G_2.

For our running example, this improvement is essential as the priority-free GT state space for the IPTGTS would be infinite since an unbounded number of errors could be created by an unbounded number of applications of the GT rule $\rho_{\mathsf{send:failure}}$ without ever applying the GT rule $\rho_{\mathsf{process:done}}$. The described procedure solves this problem since the priorities of the IPTGT rules are encoded in the constructed TA similarly as in operation IPTGTStoIPTA ruling out further applications of the GT rule $\rho_{\mathsf{send:failure}}$.

4.4 Analysis of Timed and Structural Properties

The presented analysis approach is also applicable to simpler analysis problems compared to those in Definition 4, which often provide valuable insights when e.g. unexpected results are obtained for more complex properties.

On the one hand, properties not referring to probabilities but time can be analyzed based on the TA constructed in section 4.3. For our running example, Uppaal can be used to verify the satisfaction of e.g. the timed CTL property $\mathsf{E}\,\mathsf{F}_{\leq 6}\ \phi_{\mathsf{fin}}$ (where E and F are the *exists* and *eventually* operators, respectively), stating that the AP ϕ_{fin} can be satisfied within 6 time units.

On the other hand, properties referring to neither probabilities nor time can be analyzed based on the GT state space (Q, E) constructed in section 4.1. For our

[5]The operation IPTGTStoTA is defined similarly to the operation IPTGTStoIPTA and deviates primarily by not aggregating GT steps belonging to a common IPTGT rule.

running example, the CTL property $\mathsf{A}\,\mathsf{G}\ \phi_{\mathsf{con}}$ (where A and G are the *always* and *globally* operators, respectively), stating that all errors are always connected to an observer, can be verified based on the GT state space (Q, E) since each graph in Q would be labeled by the AP ϕ_{con}. Similarly, we can also verify the CTL property $\mathsf{A}\,\mathsf{G}\ \phi_{\mathsf{2err}}$ stating that there can never be two unprocessed errors. This property is satisfied since in the timed GT state space each graph would be labeled by the AP ϕ_{2err} due to the higher priority of the IPTGT rule $\sigma_{\mathsf{process}}$. In both cases, we would use the procedure from section 4.3 where GT step generation is interleaved with the timed reachability analysis to ensure that the state space (Q, E) is finite.

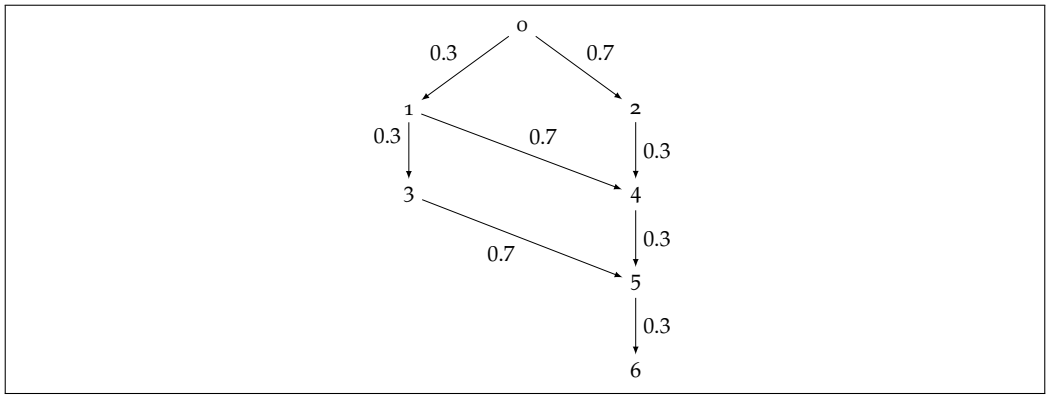
5 Evaluation

To exemplify our model checking approach for IPTGTSs and to strengthen the importance of using IPTGTSs with probability intervals instead of PTGTSs with precise probabilities, we now consider the IPTGTS of our running example for which the components have been discussed in chapter 3 and visualized in Figure 2.2. Note that the IPTGT rule $\sigma_{\mathsf{send}:I_1,I_2}$ contains two underlying GT rules with the assigned probability intervals I_1 and I_2. In our evaluation, we apply our implementation of the presented model checking approach in the tool AutoGraph for multiple instantiations of $\sigma_{\mathsf{send}:I_1,I_2}$ by considering concrete probability intervals and different analysis problems relying on the AP ϕ_{fin}.

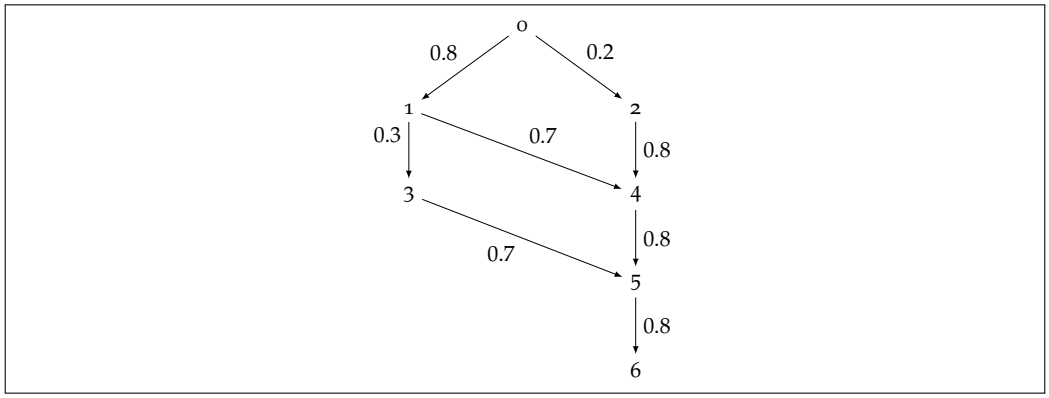
See Table 5.1 for an overview of the obtained results. The first column shows the five considered IPTGTSs instantiations where the chosen singleton probability intervals result in PTGTSs in the last three lines. The other columns show the results for computing the minimal/maximal probability to reach a graph labeled with ϕ_{fin} within 6 or 15 time units in the instantiated IPTGTS. The property $\mathcal{P}_{\mathsf{min}=?}(\mathsf{F}_{\leq 6}\ \phi_{\mathsf{fin}})$ is omitted in the table because its model checking results in the probability of 0 for each of the IPTGTSs instantiations as some adversary can delay sending for up to 5 time units preventing that any state labeled with ϕ_{fin} can be reached within 6 time units. Note that the probability values in the first two columns are identical since 6 time units used in the first column are already sufficient to reach a state labeled with ϕ_{fin} with maximal probability. Moreover, the results obtained for the PTGTSs differ in each case from the results obtained for the IPTGTSs, because, on every path to a graph labeled with ϕ_{fin} in the IPTGTS, the adversary chooses two different PMFs from the DIPD of the IPTGT rule $\sigma_{\mathsf{send}:I_1,I_2}$. For example, for the IPTGTS instantiation $S_{[0.7,0.8],[0.2,0.3]}$, the adversary defines three paths such that the resulting probability of $0.4056 = 0.3 \times 0.8 \times 0.8 \times 0.8 + 0.7 \times 0.3 \times 0.8 \times 0.8 + 0.7 \times 0.7 \times 0.3 \times 0.8$ is obtained in the first line of the first two columns. This resulting probability is the sum of the probabilities of the three paths each containing three successful sending steps with probabilities 0.7 or 0.8 and one unsuccessful sending step with probability 0.3 occurring as the first, second, or third step. See also Figure 5.1 for two further more detailed computations of resulting probabilities for the property $\mathcal{P}_{\mathsf{max}=?}(\mathsf{F}_{\leq 6}\ \phi_{\mathsf{fin}})$ and two further IPTGTSs listed in Table 5.1 where the deadline of 6 time units is again not relevant. From the different resulting probabilities, we conclude that both of the considered IPTGTSs cannot be approximated by any such PTGTSs instantiation appropriately in the sense of obtaining the same probabilities since using singleton intervals precludes adversaries that would choose different interval borders in some of their generated paths.

Table 5.1: Model checking results for the running example

Instantiation	$\mathcal{P}_{\max=?}(\mathsf{F}_{\leq 6}\ \phi_{\mathsf{fin}})$	$\mathcal{P}_{\max=?}(\mathsf{F}_{\leq 15}\ \phi_{\mathsf{fin}})$	$\mathcal{P}_{\min=?}(\mathsf{F}_{\leq 15}\ \phi_{\mathsf{fin}})$
$S_{[0.7,0.8],[0.2,0.3]}$	0.4056	0.4056	0.2366
$S_{[0.3,0.8],[0.2,0.7]}$	0.5952	0.5952	0.0387
$S_{[0.8,0.8],[0.2,0.2]}$	0.3072	0.3072	0.3072
$S_{[0.7,0.7],[0.3,0.3]}$	0.3087	0.3087	0.3087
$S_{[0.3,0.3],[0.7,0.7]}$	0.0567	0.0567	0.0567



(**a**) We consider the example IPTGTS $S_{[0.3,0.3],[0.7,0.7]}$. In this IPTGTS, each DIPD has a unique PMF resolution. We have three paths depending on when the failure occurs. The overall probability to reach the target state is $0.7 \times 0.3 \times 0.3 \times 0.3 + 0.3 \times 0.7 \times 0.3 \times 0.3 + 0.3 \times 0.3 \times 0.7 \times 0.3 = 0.3 \times 0.0189 = 0.0567$.



(**b**) We consider the example IPTGTS $S_{[0.3,0.8],[0.2,0.7]}$. We have three paths depending on when the failure occurs. The overall maximal probability to reach the target state is $0.2 \times 0.8 \times 0.8 \times 0.8 + 0.8 \times 0.7 \times 0.8 \times 0.8 + 0.8 \times 0.3 \times 0.7 \times 0.8 = 0.1024 + 0.3584 + 0.1344 = 0.5952$. Note that we can easily apply here the algorithm MinMaxProbReach from Definition 21 to obtain the given probabilities. The target state 6 has probability 1, the state 5 is maximized to 0.8 by taking $0.8 \in [0.3, 0.8]$; the state 3 is maximized to 0.7×0.8 by taking $0.7 \in [0.2, 0.7]$, the state 4 is maximized to 0.8×0.8 by taking $0.8 \in [0.3, 0.8]$; the state 1 is maximized to $0.7 \times 0.8 \times 0.8 + 0.3 \times 0.7 \times 0.8$ by taking $0.7 \in [0.2, 0.7]$ and $0.3 \in [0.3, 0.8]$ since $0.8 \times 0.8 \geq 0.7 \times 0.8$ implying that the probability to reach state 4 must be maximized before the probability to reach state 3, the state 2 is maximized to $0.8 \times 0.8 \times 0.8$ by taking $0.8 \in [0.3, 0.8]$; the state 0 is maximized to $0.8 \times (0.7 \times 0.8 \times 0.8 + 0.3 \times 0.7 \times 0.8) + 0.2 \times (0.8 \times 0.8 \times 0.8)$ by taking $0.8 \in [0.3, 0.8]$ and $0.2 \in [0.2, 0.7]$ since $0.7 \times 0.8 \times 0.8 + 0.3 \times 0.7 \times 0.8 \geq 0.8 \times 0.8 \times 0.8$ implying that the probability to reach state 1 must be maximized before the probability to reach state 2.

Figure 5.1: Computation of the probabilities for our running example

6 Conclusion and Future Work

We introduced the formalism of IPTGTSs as a high-level description language for the modeling and analysis of complex distributed embedded probabilistic real-time systems. IPTGTSs support, in addition to a nondeterministic passage of time (specified using clocks), a nondeterministic description of the probabilistic rule-based behavior (specified using probability intervals in rules). Moreover, we presented a model checking approach for IPTGTSs w.r.t. worst-case/best-case probabilistic timed reachability properties. This model checking approach is implemented in our tool AutoGraph and is based on a translation of IPTGTSs into PTA via IPTA. The PTA resulting from this translation can then be analyzed using the Prism model checker.

As future work, we will extend Metric Temporal Graph Logic (MTGL) to IPTGTSs to be able to specify more complex properties on the structure dynamics, timed behavior, and probabilistic behavior of the given IPTGTSs. Such an extension is then to be included into the mapping of IPTGTSs to PTA to allow for the automated verification using Prism.

Bibliography

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

[2] 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.

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

[4] C. Krause and H. Giese. "Model Checking Probabilistic Real-Time Properties for Service-Oriented Systems with Service Level Agreements". In: *Proceedings 13th International Workshop on Verification of Infinite-State Systems, INFINITY 2011, Taipei, Taiwan, 10th October 2011*. Edited by F. Yu and C. Wang. Volume 73. EPTCS. 2011, pages 64–78. DOI: 10.4204/EPTCS.73.8.

[5] 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.-J. 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.

[6] C. Krause and H. Giese. *Quantitative modeling and analysis of service-oriented real-time systems using interval probabilistic timed automata*. 2012. ISBN: 978-3-86956-171-4.

[7] 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.

[8] M. Z. Kwiatkowska, G. Norman, and D. Parker. "Stochastic Games for Verification of Probabilistic Timed Automata". In: *Formal Modeling and Analysis of Timed Systems, 7th International Conference, FORMATS 2009, Budapest, Hungary, September 14-16, 2009. Proceedings*. Edited by J. Ouaknine and F. W. Vaandrager. Volume 5813. Lecture Notes in Computer Science. Springer, 2009, pages 212–227. DOI: 10.1007/978-3-642-04368-0_17.

[9] M. Z. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. "Performance analysis of probabilistic timed automata using digital clocks". In: *Formal Methods Syst. Des.* 29.1 (2006), pages 33–78. DOI: 10.1007/s10703-006-0005-2.

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

[11] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. "Symbolic model checking for probabilistic timed automata". In: *Inf. Comput.* 205.7 (2007), pages 1027–1077. DOI: 10.1016/j.ic.2007.01.004.

[12] M. Maximova, H. Giese, and C. Krause. "Probabilistic Timed Graph Transformation Systems". In: *Graph Transformation - 10th International Conference, ICGT 2017, Held as Part of STAF 2017, Marburg, Germany, July 18-19, 2017, Proceedings*. Edited by J. de Lara and D. Plump. Volume 10373. Lecture Notes in Computer Science. Springer, 2017, pages 159–175. ISBN: 978-3-319-61469-4. DOI: 10.1007/978-3-319-61470-0_10.

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

[14] M. Maximova, S. Schneider, and H. Giese. "Interval Probabilistic Timed Graph Transformation Systems". In: *Graph Transformation - 14th International Conference, ICGT 2021, Held as Part of STAF 2021, Virtual Event, June 24-25, 2021, Proceedings*. Edited by F. Gadducci and T. Kehrer. Volume 12741. Lecture Notes in Computer Science. Springer, 2021, pages 221–239. DOI: 10.1007/978-3-030-78946-6_12.

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

[16] G. Norman, D. Parker, and J. Sproston. "Model checking for probabilistic timed automata". In: *Formal Methods Syst. Des.* 43.2 (2013), pages 164–190. DOI: 10.1007/s10703-012-0177-x.

[17] 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.

[18] 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.

[19] R. Segala. "Modeling and verification of randomized distributed real-time systems". PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[20] *UPPAAL*. Department of Information Technology at Uppsala University, Sweden and Department of Computer Science at Aalborg University, Denmark. 2021. URL: https://uppaal.org/.

[21] J. Zhang, J. Zhao, Z. Huang, and Z. Cao. "Model Checking Interval Probabilistic Timed Automata". In: *2009 First International Conference on Information Science and Engineering*. 2009, pages 4936–4940. DOI: 10.1109/ICISE.2009.749.

Glossary

AP Atomic Proposition.

DIPD Discrete Interval Probability Distribution.

GTS Graph Transformation System.

IPA Interval Probabilistic Automaton.

IPTA Interval Probabilistic Timed Automaton.

IPTGTS Interval Probabilistic Timed Graph Transformation System.

IPTS Interval Probabilistic Timed System.

PA Probabilistic Automaton.

PGTS Probabilistic Graph Transformation System.

PMF Probability Mass Function.

PTA Probabilistic Timed Automaton.

PTCTL Probabilistic Timed Computation Tree Logic.

PTGTS Probabilistic Timed Graph Transformation System.

PTS Probabilistic Timed System.

TA Timed Automaton.

TGTS Timed Graph Transformation System.

A From IPTA to PTS via IPTS

We now present the semantics of IPTA by (a) translating an IPTA into an IPTS (cf. [6]) and (b) translating an IPTS into an PTS. This complements the direct step from Definition 3 where an IPTA induces a PTS directly. Based on IPTSs as an intermediate model, it is easier to state equivalence (e.g. between IPTA and PTA obtained from them) w.r.t. probabilistic real-time reachability properties and to verify such equivalences. The first translation of IPTA into IPTSs removes clocks as used in guards and invariants from the model. The second translation of IPTSs into PTSs then removes probability intervals from the model. Note that already the IPTS representation but also the PTS representation will usually[1] have an infinite number of states because of the precise representation of all real-time delays and the corresponding steps.

We now introduce the syntax of IPTSs.

Definition 11 (Interval Probabilistic Timed System (IPTS))**.** An *interval probabilistic timed system* (*IPTS*) I is a tuple with the following components.

- $\mathsf{states}(I)$ is a (possibly infinite) set of states,
- $\mathsf{istate}(I)$ is the unique initial state from $\mathsf{states}(I)$,
- $\mathsf{acts}(I)$ is a finite set of actions disjoint from $\mathbf{R}_0^+$,
- $\mathsf{steps}(I) \subseteq \mathsf{states}(I) \times (\mathsf{acts}(I) \cup \mathbf{R}_0^+) \times \mathsf{DIPD}(\mathsf{states}(I))$ is a (possibly infinite) set of IPTS steps of the form $(s_1, a, (\mu_1, \mu_2))$ where s_1 is the source state, a is an action or a duration, and where (μ_1, μ_2) is a DIPD on the set $\mathsf{states}(I)$ containing target states,
- $\mathsf{aps}(I)$ is a finite set of APs, and
- $\mathsf{lab}(I) : \mathsf{states}(I) \rightarrow 2^{\mathsf{aps}(I)}$ maps each state to a set of APs that are true in that state.

We now define a translation from IPTA to IPTSs by determining solutions for valuations w.r.t. invariants, guards, and resets. States of the induced IPTS are then given by a pair of a locations and corresponding valuation. See Figure A.1 for a visualization regarding the case of discrete transitions.

Definition 12 (IPTS induced by IPTA)**.** Every IPTA A induces a unique *interval probabilistic timed system* (*IPTS*) $\mathsf{IPTAtoIPTS}(A) = I$ consisting of the following components.

- $\mathsf{states}(I) = \{(\ell, v) \in \mathsf{locs}(A) \times \mathsf{CV}(\mathsf{clocks}(A)) \mid v \models \mathsf{invs}(A)(\ell)\}$,
- $\mathsf{istate}(I) = (\mathsf{iloc}(A), \mathsf{ICV}(\mathsf{clocks}(A)))$,
- $\mathsf{acts}(I) = \mathsf{acts}(A)$,
- $((\ell, v), a, (\mu_1, \mu_2)) \in \mathsf{steps}(I)$ iff one of the two following cases applies.

[1] Unless invariants are chosen to prevent the passage of time entirely, which would be an undesirable modeling error.

- ∘ Timed step:
  - ▷ $a \in \mathbf{R}_0^+$,
  - ▷ $(\ell, v + t') \in \mathsf{states}(I)$ (for all $0 \leq t' \leq a$), and
  - ▷ $(\mu_1, \mu_2) = \mathsf{DIPD}_1(\mathsf{states}(I), (\ell, v + a))$.
- ∘ Discrete step:[2]
  - ▷ $a \in \mathsf{acts}(A)$,
  - ▷ $(\ell, a, \psi, (\mu_1', \mu_2')) \in \mathsf{edges}(A)$,
  - ▷ $v \models \psi$, and
  - ▷ $\mu_i(\ell', v') = \sum \{\!|\mu_i'(X, \ell') \mid X \subseteq \mathsf{clocks}(A), v' = v[X := 0]|\!\}$ (for all $(\ell', v') \in \mathsf{states}(I)$ and $i \in \{1, 2\}$).
- $\mathsf{aps}(I) = \mathsf{aps}(A)$, and
- $\mathsf{lab}(I)(\ell, v) = \mathsf{lab}(A)(\ell)$.

We now define the translation from IPTSs to PTSs by determining solutions for each DIPD used in some IPTS step. Thereby, the remaining nondeterminism given by the DIPDs is resolved into all possible choices of PMFs explicitly.

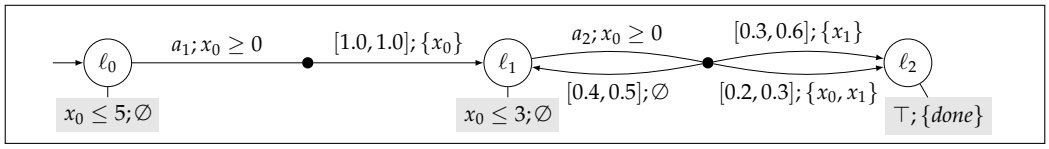
Definition 13 (PTS induced by IPTS)**.** Every IPTS I induces a unique *probabilistic timed system* (*PTS*) $\mathsf{IPTStoPTS}(I) = P$ consisting of the following components.

- $\mathsf{states}(P) = \mathsf{states}(I)$,
- $\mathsf{istate}(P) = \mathsf{istate}(I)$,
- $\mathsf{acts}(P) = \mathsf{acts}(I)$,
- $(s, a, \mu) \in \mathsf{steps}(P)$ iff there is some IPTS step $(s, a, (\mu_1, \mu_2)) \in \mathsf{steps}(I)$ such that $\mu \in \langle (\mu_1, \mu_2) \rangle$.
- $\mathsf{aps}(P) = \mathsf{aps}(I)$, and
- $\mathsf{lab}(P) = \mathsf{lab}(I)$.

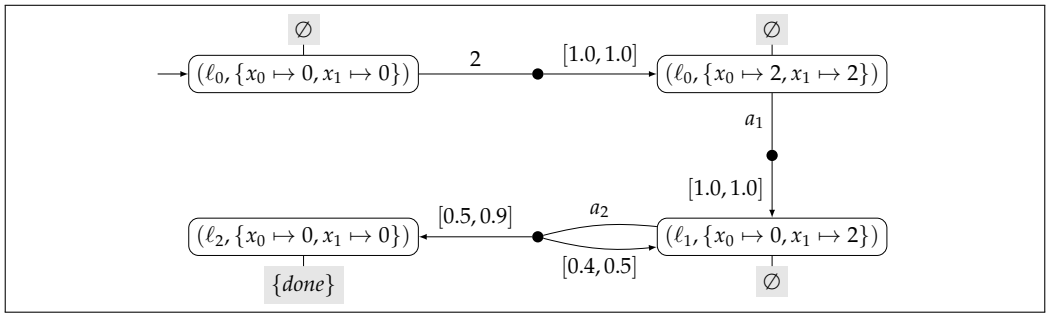
We now state that the operations IPTAtoIPTS and IPTStoPTS presented here are equivalent to the direct translation using IPTAtoPTS.

Lemma 1 (Equivalent Translation from IPTA to PTS)**.** If A is an IPTA, then the two PTSs $\mathsf{IPTStoPTS}(\mathsf{IPTAtoIPTS}(A))$ and $\mathsf{IPTAtoPTS}(A)$ coincide.

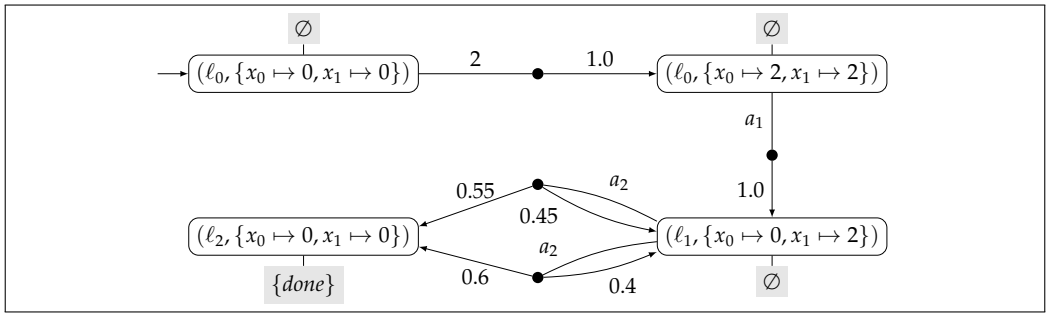
[2]Note that $\langle (\mu_1', \mu_2') \rangle$ may be empty but that the discrete step can then not be used in a path of the semantics of the resulting IPTS.



(**a**) An IPTA to be translated into the IPTS from Figure A.1b



(**b**) A *small fragment of the* IPTS obtained from the IPTA from Figure A.1a using IPTAtoIPTS where only one timed step has been included here. Note that one of the intervals for the a_2 step is $[0.5, 0.9]$ because the resulting valuation can be obtained by resetting only x_1 or by resetting x_0 and x_1 since x_0 has already the value of 0 exemplifying the need for using summation over multisets in various definitions in this paper



(**c**) A *small fragment of the* PTS obtained from the IPTS from Figure A.1b using IPTStoPTS. Note that here, not all PMFs are included for a given DIPD. Especially for the last step, it is important that there are actually uncountably many steps with probabilities $0.6 - x$ and $0.4 + x$ for each $x \in [0.0, 0.1]$

Figure A.1: Visualization for translation from IPTA via IPTS to PTS

B Specification of Probabilistic Timed Structures

We now discuss the formal specification of PTSs as they can be obtained from IPTA, PTA, PTGTSs, and IPTGTSs using Probabilistic Timed Computation Tree Logic (PTCTL), which has been introduced in [10] and for which decidability has been shown for PTSs induced by PTA [10, Section 6, pp. 116] in terms of a model checking algorithm. However, this proposed algorithm exhibits a runtime that is exponential in the number of clocks and the largest constant appearing in clock constraints of the PTA or PTCTL condition. Subsequently, simpler inputs have been considered in terms of (a) subclasses of PTA and (b) either subclasses of PTCTL or similar specifications such as so called probabilistic real-time queries. For such simpler inputs, model checking procedures have been devised such as in [10, Section 7, pp. 128] and [8, 9, 11]. See also [16] for a later survey. Furthermore, the probabilistic model checking tool Prism has been extended to cover multiple of these approaches using so called *engines* but the full support in terms of the decision procedure for PTCTL has not been implemented.

We now introduce PTCTL from [10] in our notation with its syntax and semantics on PTSs before focussing on probabilistic real-time queries for which Prism has implementation support.

Definition 14 (Syntax of PTCTL)**.** If AP is a finite set of APs and X, Z are disjoint finite sets of system and condition clocks, then ψ is a *PTCTL condition*, written $\psi \in \mathsf{PTCTL}(AP, X, Z)$, if it is generated using the following grammar.

$$\psi ::= \top \mid ap \mid \neg\psi \mid \psi \wedge \psi \mid \neg\psi \mid z.\,\psi \mid \mathcal{P}_{\sqsupseteq\lambda}(\psi\ \exists\mathcal{U}\ \psi) \mid \mathcal{P}_{\sqsupseteq\lambda}(\psi\ \forall\mathcal{U}\ \psi)$$

where $z \in Z$ is a condition clock, $ap \in AP$ is an AP, $\psi \in \mathsf{CC}(X \cup Z)$ is a clock constraint over the system and condition clocks, $\lambda \in [0,1]$, and $\sqsupseteq \in \{>, \geq\}$.

The semantics of PTCTL is then given by a satisfaction relation for a state (ℓ, vs) of a PTS containing a location ℓ and a valuation vs of system clocks X (as obtained from e.g. a PTA) and a valuation vc of the condition clocks Z.

Definition 15 (Semantics of PTCTL)**.** If P is a PTS, X is a finite set of system clocks, $\mathsf{states}(P) \subseteq L \times \mathsf{CV}(X)$, Z is a finite set of condition clocks, $\psi \in \mathsf{PTCTL}(\mathsf{aps}(P), X, Z)$, $s = (\ell, vs) \in \mathsf{states}(P)$ is a state, and $vc : Z \to \mathbf{R}_0^+$ is a clock valuation for the condition clocks, then (s, vc) satisfies ψ, written $(s, vc) \models \psi$, if an item applies.

- $\psi = \top$.
- $\psi = ap$ and $ap \in \mathsf{lab}(P)(s)$.
- $\psi = \psi$ and $v \models \psi$.

- $\psi = \psi_1 \wedge \psi_2$, $(s, vc) \models \psi_1$, and $(s, vc) \models \psi_2$.
- $\psi = \neg\psi'$ and not $(s, vc) \models \psi'$.
- $\psi = z.\ \psi'$ and $(s, vc[z := 0]) \models \psi'$.
- $\psi = \mathcal{P}_{\sqsupseteq\lambda}(\psi_1\ \exists\mathcal{U}\ \psi_2)$ and there is some $A \in \mathsf{AdvD}(P)$ such that $\mathsf{Prob}(P, A, s)(\{\pi \in \mathsf{Paths}(P, A, s) \mid (\pi, vc) \models \psi_1\ \mathcal{U}\ \psi_2\}) \sqsupseteq \lambda$.
- $\psi = \mathcal{P}_{\sqsupseteq\lambda}(\psi_1\ \forall\mathcal{U}\ \psi_2)$ and for all $A \in \mathsf{AdvD}(P)$ it holds that $\mathsf{Prob}(P, A, s)(\{\pi \in \mathsf{Paths}(P, A, s) \mid (\pi, vc) \models \psi_1\ \mathcal{U}\ \psi_2\}) \sqsupseteq \lambda$.

where

- $(\pi, vc) \models \psi_1\ \mathcal{U}\ \psi_2$ holds if $P_{\mathit{UntilSat}}(\pi, vc, <\infty, \psi_1, \psi_2)$ and
- $P_{\mathit{UntilSat}}(\pi, vc, \sim t, \psi_1, \psi_2)$ holds if there is some $((\ell_i, vs_i), i, \delta) \in \mathsf{pos}(\pi)$ such that $\mathsf{dur}(\pi, i) + \delta \sim t$, $((\ell_i, vs_i + \delta), vc + (\mathsf{dur}(\pi, i) + \delta)) \models \psi_2$, and for all $((\ell_j, vs_j), j, \delta') \in \mathsf{pos}(\pi)$ such that $((\ell_j, vs_j), j, \delta') < ((\ell_i, vs_i), i, \delta)$ it holds that $((\ell_j, vs_j + \delta'), vc + (\mathsf{dur}(\pi, j) + \delta')) \models \psi_1 \vee \psi_2$.
  Here, $((\ell_i, vs_i), i, \delta)$ is the position where the right PTCTL condition is satisfied, the state (ℓ_i, vs_i) and the valuation vc are adjusted for this purpose by shifting it by δ and $\mathrm{dur}(\pi) + \delta$, and for, all intermediate states, the left or right condition is satisfied.[1]

We now provide an example of PTCTL conditions used for the specification of a PTS induced by an IPTA.

Example 4 (Specification using PTCTL)**.** We may state the following PTCTL conditions for the IPTA from Figure 2.4a.

$$z.\ \mathcal{P}_{\geq 0.95}(\neg \mathit{error}\ \exists\mathcal{U}\ \mathit{success} \wedge z \leq 20) \tag{B.1}$$

$$z.\ \mathcal{P}_{\geq 0.95}(\neg \mathit{error}\ \forall\mathcal{U}\ \mathit{success} \wedge z \leq 20) \tag{B.2}$$

The IPTA from Figure 2.4a only satisfies (B.1) but not (B.2) because an adversary may let the IPTA stay in ℓ_0 for 20 time units.

In addition to PTCTL there are further operators $\mathcal{P}_{\mathsf{max}=?}(\cdot)$ and $\mathcal{P}_{\mathsf{min}=?}(\cdot)$ for obtaining the maximum and minimum probability (over all adversaries) that the subcondition is satisfied.

Definition 16 (Syntax of Probabilistic Real-time Queries)**.** If AP is a finite set of APs and X, Z are disjoint finite sets of system and condition clocks, then ψ is a *probabilistic real-time query*, written $\psi \in \mathsf{PTQ}(AP, X, Z)$, if it is generated using the following grammar.

$$\psi ::= \mathcal{P}_{\mathsf{max}=?}(\psi) \mid \mathcal{P}_{\mathsf{min}=?}(\psi)$$
$$\psi ::= \top \mid ap \mid \psi \mid \psi \wedge \psi \mid \neg\psi \mid z.\ \psi \mid \psi\ \mathsf{U}\ \psi \mid \psi\ \mathsf{U}_{\sqsubseteq c}\ \psi$$

[1] It is not only required that the left condition ψ_1 is satisfied because for a PTA with the initial state s_0 labeled with the AP *ap0* and with invariant $x \leq 2$, with the state s_1 labeled with the AP *ap1* and with invariant $\top$, and one PTA edge from s_0 with guard $x \leq 2$ that reaches s_1 with probability 1, then $\mathcal{P}_{>2\lambda}(\mathit{ap0}\ \exists\mathcal{U}\ \mathit{ap1})$ needs to determine some $\delta > 2$ such that the PTA was in state s_0 until then. This would not be the case since any $\delta > 2$ allows for some $2 < t < \delta$ where the PTA is not in state s_0 anymore. With the presented semantics, the PTA would satisfy the condition as desired.

where $z \in Z$ is a condition clock, $ap \in AP$ is an AP, $\psi \in \mathsf{CC}(X \cup Z)$ is a clock constraint over the system and condition clocks, $c \in \mathbf{R}_0^+$ is an upper bound duration, and $\sqsubset \in \{<, \leq\}$.

Definition 17 (Semantics of Probabilistic Real-time Queries)**.** If P is a PTS, X is a finite set of system clocks, $\mathsf{states}(P) \subseteq L \times \mathsf{CV}(X)$, Z is a finite set of condition clocks, $v_0 = \mathsf{ICV}(Z) : Z \rightarrow \mathbf{R}_0^+$ is the initial valuation of the condition clocks, and $\psi \in \mathsf{PTQ}(\mathsf{aps}(P), X, Z)$, then ψ has a result from $[0, 1]$ as follows.

$$\begin{aligned}
\langle\!\langle \mathcal{P}_{\mathsf{max}=?}(\psi), P \rangle\!\rangle &= \sup(R) \\
\langle\!\langle \mathcal{P}_{\mathsf{min}=?}(\psi), P \rangle\!\rangle &= \inf(R) \\
R = \{&\mathsf{Prob}(P, A, \mathsf{istate}(P))(\\
&\{\pi \in \mathsf{Paths}(P, A, \mathsf{istate}(P)) \mid \\
&(\pi, v_0) \models \psi\}) \mid A \in \mathsf{AdvD}(P)\}
\end{aligned}$$

and where $(\pi, vc) \models \psi$ is satisfied, if an item applies.

- $\psi = \top$.
- $\psi = ap$ and $ap \in \mathsf{lab}(P)(\mathsf{first}(\pi))$.
- $\psi = \psi$ and $vc \models \psi$.
- $\psi = \psi_1 \wedge \psi_2$, $(\pi, vc) \models \psi_1$, and $(\pi, vc) \models \psi_2$.
- $\psi = \neg\psi'$ and not $(\pi, vc) \models \psi'$.
- $\psi = z.\ \psi'$ and $(\pi, vc[z := 0]) \models \psi'$
- $\psi = \psi_1\ \mathsf{U}\ \psi_2$ and[2] $P_{\mathit{UntilSat}}(\pi, vc, < \infty, \psi_1, \psi_2)$.
- $\psi = \psi_1\ \mathsf{U}_{\sqsubset c}\ \psi_2$ and[2] $P_{\mathit{UntilSat}}(\pi, vc, \sqsubset c, \psi_1, \psi_2)$.

However, Prism does not support conditions of PTCTL and all subconditions for the two mentioned probabilistic real-time queries. We now provide overview of the properties that can be analyzed using Prism.[3]

The *Digital Clocks Engine* supports the conditions ϕ generated by the following grammar.

$$\begin{aligned}
\phi &::= \mathcal{P}_{\mathsf{max}=?}(\psi) \mid \mathcal{P}_{\mathsf{min}=?}(\psi) \mid \mathcal{P}_{\geq\kappa}(\psi) \mid \mathcal{P}_{>\kappa}(\psi) \mid \mathcal{P}_{\leq\kappa}(\psi) \mid \mathcal{P}_{<\kappa}(\psi) \\
\psi &::= \neg\psi \mid \mathsf{F}\ \chi \mid \mathsf{F}_{\leq c}\ \chi \mid \mathsf{F}_{<c}\ \chi \mid \chi\ \mathsf{U}\ \chi \\
\chi &::= \neg\chi \mid \chi \wedge \chi \mid \top \mid ap \mid x = c \mid x \geq c \mid x \leq c
\end{aligned}$$

Here, $\kappa \in [0, 1]$ and $c \in \mathbf{N}$. As an additional restriction, the digital clocks engine requires that clock constraints in the condition as well as in the PTA do not use strict clock comparison (i.e., $x < c$ and $x > c$ are forbidden) and diagonal free (i.e., $x + c \sim x + c$ is forbidden for any $\sim$). For this purpose, clock constraints of the form $x \geq c$ may only occur in an even number of negations (for simplicity we keep this out of the grammar above).

[2] See Definition 15.

[3] See also `https://www.prismmodelchecker.org/manual/ThePRISMLanguage/PTAs` and `https://www.prismmodelchecker.org/manual/PropertySpecification/PTAProperties`.

It has to be noted that $\mathcal{P}_{\mathsf{max}=?}(x \leq 1 \,\mathsf{U}\, x \geq 2)$ will (for suitable PTA) result in 1.0 because in the digital clocks analysis, clocks advance not by $t \in \mathbf{R}_0^+$ but by $t \in \mathbf{N}$. However, the digital clocks engine provides correct results (w.r.t. PTCTL and PTA) for certain conditions such as $\mathcal{P}_{\mathsf{max}=?}(ap \,\mathsf{U}\, x \geq 2)$.

The *Stochastic Games Engine* supports the conditions ϕ generated by the following grammar.

$$\begin{aligned}\phi &::= \mathcal{P}_{\mathsf{max}=?}(\psi) \mid \mathcal{P}_{\mathsf{min}=?}(\psi)\\ \psi &::= \mathsf{F}\,\chi \mid \mathsf{F}_{\leq c}\,\chi \mid \mathsf{F}_{<c}\,\chi\\ \chi &::= \neg\chi \mid \chi \wedge \chi \mid \top \mid ap\end{aligned}$$

The *Backwards Reachability Engine* supports the conditions ϕ generated by the following grammar.

$$\begin{aligned}\phi &::= \mathcal{P}_{\mathsf{max}=?}(\psi)\\ \psi &::= \mathsf{F}\,\chi \mid \mathsf{F}_{\leq c}\,\chi \mid \mathsf{F}_{<c}\,\chi\\ \chi &::= \neg\chi \mid \chi \wedge \chi \mid \top \mid ap\end{aligned}$$

Note that the backwards reachability engine may be more efficient than the stochastic games engine for certain inputs.

C Analysis of Interval Probabilistic Timed Automata

An approach to model check IPTA against PTCTL was presented in [21] but no implementation is available as of now and no formal semantics of PTCTL for the case of IPTA has been provided. Later on, a model checking procedure extending the "stochastic games engine" of Prism was presented in [6] inheriting the same restrictions of the original Prism stochastic games engine. However, the adaptation was also not integrated into the official version of Prism as of now and the other engines do not support IPTA even in this adaptation as of now. Besides this adaptation of Prism, [6] also discusses a reduction of IPTA to PTA that is sound w.r.t. certain probabilistic real-time queries. Both of these approaches are discussed subsequently in more detail. Note, in parts, this chapter can be understood to provide additional details for section 4.2.

We now recall the IPTA analysis algorithm MinMaxProbReach (compare [6, Section 5.2.2, pp. 30]) for computing the minimal and maximal probability (w.r.t. the choice among the adversaries) to reach one state in a set of target states T from the initial state in the PTS obtained from the IPTS obtained from an IPTA.

Basically, it is important to understand that $\sum \{\!|a_i \times b_i \mid i \in I|\!\}$ is maximized (minimized) for a fixed probability vector b and an ordering of a probability vector a with $\sum \{\!|a_i \mid i \in I|\!\} = 1$ (essentially a is a PMF later on) when the nth highest value of b is paired with the nth highest (lowest) value of a.[1] Hence, to maximize the probability to reach a certain state, the nondeterminism of a DIPD (μ_1, μ_2) must be resolved for a given probability vector b for reaching the target states (μ_1, μ_2) such that the probabilities in b (associated with a target state) are paired with the highest probability permitted by the DIPD for that target state. Certainly, we must resolve nondeterminism of a DIPD to obtain a *well-formed* PMF from the semantics of the DIPD: that is, each choice from an interval for some target state depends on the previous selections (the total probability of 1 may not be exceeded) and on the choices that must be made for the subsequent intervals (the remaining probability must suffice for these choices as well). To decouple this handling from the subsequently presented algorithm, we now introduce the operation corner to obtain the so called corners of a DIPD w.r.t. a given probability vector.

As a first step, we provide two operations for converting between orderings and probability vectors. The operation orderedPV forgets about the precise probability

[1] For example, $b = (0.4, 0.8)$ and $a = (0.7, 0.3)$ results in the maximal value $0.8 \times 0.7 + 0.4 \times 0.3 = 0.64 + 0.12 = 0.76$ minimal value $0.4 \times 0.7 + 0.8 \times 0.3 = 0.28 + 0.24 = 0.52$.

and only keeps the ordering induced by the forgotten values, which results in ambiguity (multiple possible orderings) when $p(s_i) = p(s_j)$ for some $s_i \neq s_j$.

Definition 18 (Ordering of Probability Vector)**.** If A is a finite set, $p : A \rightharpoonup [0,1]$ is a probability vector, $s = (s_1, \ldots, s_n)$ is a duplication free ordering of A, $p(s_i) \geq p(s_{i+1})$ for each $1 \leq i \leq n-1$, then s is some ordering of p, written $s \in \mathsf{orderedPV}(p)$.

The operation pvector creates a precise probability vector based on the given ordering.

Definition 19 (Probability Vector of Ordering)**.** If A is a finite set, $s = (s_1, \ldots, s_n)$ is a duplication free ordering of A, $p : A \rightharpoonup [0,1]$ is a probability vector, $p(s_i) = 1/i$ for each $1 \leq i \leq n$, then p is the probability vector induced by s, written $\mathsf{pvector}(s) = p$.

The connection between these two support operations is obvious: information forgotten is not reestablished in general but orderings are preserved.

Lemma 2 (Connection between orderedPV and pvector)**.** If A is a finite set, $p : A \rightharpoonup [0,1]$ is a probability vector, and s is a duplication free ordering of A, then the following items are satisfied.

- $\mathsf{orderedPV}(\mathsf{pvector}(s)) = \{s\}$.
- If $s \in \mathsf{orderedPV}(p)$, then $\mathsf{pvector}(s)$ may be different from p.

We now define the corners (given by PMFs) of a DIPD w.r.t. a given ordering.

Definition 20 (Corner of DIPD w.r.t. Ordering)**.** If A is a finite set, $p : A \rightharpoonup [0,1]$ is a probability vector, $s = (s_1, \ldots, s_n) \in \mathsf{orderedPV}(p)$, $(\mu_1, \mu_2) \in \mathsf{DIPD}(A)$ is a DIPD on A, and $op \in \{\mathsf{min}, \mathsf{max}\}$ is an operation, then $(\mu, \mu) \in \mathsf{DIPD}(A)$ *is the corner of* (μ_1, μ_2) *w.r.t.* p, written $\mathsf{corner}(op, (\mu_1, \mu_2), p) = (\mu, \mu)$, if the following items are satisfied.

- $op' = \mathsf{min}$ and $(\mu_1', \mu_2') = (\mu_1, \mu_2)$ if $op = \mathsf{max}$,
- $op' = \mathsf{max}$ and $(\mu_1', \mu_2') = (\mu_2, \mu_1)$ if $op = \mathsf{min}$,
- $\mathsf{taken}(i) = \sum \{\!|\mu(s_j) \mid 1 \leq j \leq i-1|\!\}$ (for each $1 \leq i \leq n$) is the accumulated probability used for the states s_1–s_{i-1},
- $\mathsf{reserved}(i) = \sum \{\!|\mu_1'(s_j) \mid i+1 \leq j \leq n|\!\}$ (for each $1 \leq i \leq n$) is the accumulated probability that has to be reserved for the s_{i+1}–s_n, and
- $\mu(s_i) = op'(\mu_2'(s_i), 1 - \mathsf{taken}(i) - \mathsf{reserved}(i))$ (for each $1 \leq i \leq n$) is the probability selected.

We now consider an example of two applications of the operation corner.

Example 5 (Corner of DIPD w.r.t. Ordering)**.** Let $A = \{s_1, s_2, s_3\}$, $p : A \rightharpoonup [0,1]$ is a probability vector, $p = \{s_1 \mapsto 1, s_2 \mapsto 0.5, s_3 \mapsto 0.2\}$, $s = (s_1, \ldots, s_n) \in \mathsf{orderedPV}(p)$, and $(\mu_1, \mu_2) \in \mathsf{DIPD}(A)$ is a DIPD on A with

$$\mu_1 = \{s_1 \mapsto 0.3, s_2 \mapsto 0.4, s_3 \mapsto 0.2\}$$
$$\mu_2 = \{s_1 \mapsto 0.7, s_2 \mapsto 0.6, s_3 \mapsto 0.8\}.$$

We construct $(\mu, \mu) = \mathsf{corner}(\mathsf{max}, (\mu_1, \mu_2), p)$.

$$
\begin{aligned}
\mathsf{taken}(1) &= \sum \{\!|\mu(s_j) \mid 1 \leq j \leq 1-1|\!\} \\
&= 0 \\
\mathsf{reserved}(1) &= \sum \{\!|\mu_1(s_j) \mid 1+1 \leq j \leq 3|\!\} \\
&= \mu_1(s_2) + \mu_1(s_3) \\
&= 0.4 + 0.2 \\
&= 0.6 \\
\mu(s_1) &= \mathsf{min}(\mu_2(s_1), 1 - \mathsf{taken}(1) - \mathsf{reserved}(1)) \\
&= \mathsf{min}(0.7, 1 - 0 - 0.6) \\
&= \mathsf{min}(0.7, 0.4) \\
&= 0.4 \\
\mathsf{taken}(2) &= \sum \{\!|\mu(s_j) \mid 1 \leq j \leq 2-1|\!\} \\
&= \mu(s_1) \\
&= 0.4 \\
\mathsf{reserved}(2) &= \sum \{\!|\mu_1(s_j) \mid 2+1 \leq j \leq 3|\!\} \\
&= \mu_1(s_3) \\
&= 0.2 \\
\mu(s_2) &= \mathsf{min}(\mu_2(s_2), 1 - \mathsf{taken}(2) - \mathsf{reserved}(2)) \\
&= \mathsf{min}(0.6, 1 - 0.4 - 0.2) \\
&= \mathsf{min}(0.6, 0.4) \\
&= 0.4 \\
\mathsf{taken}(3) &= \sum \{\!|\mu(s_j) \mid 1 \leq j \leq 3-1|\!\} \\
&= \mu(s_1) + \mu(s_2) \\
&= 0.4 + 0.4 \\
&= 0.8 \\
\mathsf{reserved}(3) &= \sum \{\!|\mu_1(s_j) \mid 3+1 \leq j \leq 3|\!\} \\
&= 0 \\
\mu(s_3) &= \mathsf{min}(\mu_2(s_3), 1 - \mathsf{taken}(3) - \mathsf{reserved}(3)) \\
&= \mathsf{min}(0.8, 1 - 0.8 - 0) \\
&= \mathsf{min}(0.8, 0.2) \\
&= 0.2
\end{aligned}
$$

Finally, we obtain the PMF (μ, μ) with

$$\mu = \{s_1 \mapsto 0.4, s_2 \mapsto 0.4, s_3 \mapsto 0.2\}$$

As a second application, we construct $(\mu', \mu') = \mathsf{corner}(\mathsf{min}, (\mu_1, \mu_2), p)$.

$$
\begin{aligned}
\mathsf{taken}(1) &= \sum \{\!|\mu'(s_j) \mid 1 \leq j \leq 1-1|\!\} \\
&= 0 \\
\mathsf{reserved}(1) &= \sum \{\!|\mu_2(s_j) \mid 1+1 \leq j \leq 3|\!\} \\
&= \mu_2(s_2) + \mu_2(s_1) \\
&= 0.6 + 0.8 \\
&= 1.4 \\
\mu'(s_1) &= \mathsf{max}(\mu_1(s_1), 1 - \mathsf{taken}(1) - \mathsf{reserved}(1)) \\
&= \mathsf{max}(0.3, 1 - 0 - 1.4) \\
&= \mathsf{max}(0.3, -0.4) \\
&= 0.3 \\
\mathsf{taken}(2) &= \sum \{\!|\mu'(s_j) \mid 1 \leq j \leq 2-1|\!\} \\
&= \mu'(s_1) \\
&= 0.3 \\
\mathsf{reserved}(2) &= \sum \{\!|\mu_2(s_j) \mid 2+1 \leq j \leq 3|\!\} \\
&= \mu_2(s_1) \\
&= 0.8 \\
\mu'(s_2) &= \mathsf{max}(\mu_1(s_2), 1 - \mathsf{taken}(2) - \mathsf{reserved}(2)) \\
&= \mathsf{max}(0.4, 1 - 0.3 - 0.8) \\
&= \mathsf{max}(0.4, -0.1) \\
&= 0.4 \\
\mathsf{taken}(3) &= \sum \{\!|\mu'(s_j) \mid 1 \leq j \leq 3-1|\!\} \\
&= \mu'(s_1) + \mu'(s_2) \\
&= 0.3 + 0.4 \\
&= 0.7 \\
\mathsf{reserved}(3) &= \sum \{\!|\mu_2(s_j) \mid 3+1 \leq j \leq 3|\!\} \\
&= 0 \\
\mu'(s_1) &= \mathsf{max}(\mu_1(s_3), 1 - \mathsf{taken}(3) - \mathsf{reserved}(3)) \\
&= \mathsf{max}(0.2, 1 - 0.7 - 0) \\
&= \mathsf{max}(0.2, 0.3) \\
&= 0.3
\end{aligned}
$$

Finally, we obtain the PMF (μ', μ') with

$$\mu' = \{s_1 \mapsto 0.3, s_2 \mapsto 0.4, s_3 \mapsto 0.3\}$$

We now continue this example by obtaining the minimal and maximal probabilities based on μ' and μ.

Example 6 (Minimal and Maximal Probabilities)**.** We obtain the following lower and upper bounds by pairing largest with smallest and largest with largest values while respecting the constraints of the DIPD.

$$\sum \{\!|(\mu' \times p)(s_i) \mid 1 \leq i \leq 3|\!\} = 0.3 * 1 + 0.4 * 0.5 + 0.3 * 0.2 = 0.53$$
$$\sum \{\!|(\mu \times p)(s_i) \mid 1 \leq i \leq 3|\!\} = 0.4 * 1 + 0.4 * 0.5 + 0.2 * 0.2 = 0.64$$

We now state that the construction of corners has the desired properties in terms of generating minimal/maximal probabilities given as lower and upper bounds.

Lemma 3 (Correctness of Corners)**.** If A is a finite set, $p : A \rightharpoonup [0,1]$ is a probability vector on A, $p' : A \rightharpoonup [0,1]$ with $p'(s) = 1 - p(s)$, $(\mu_1, \mu_2) \in \mathsf{DIPD}(A)$ is a DIPD on A, $(\bar{\mu}, \bar{\mu}) \in \langle (\mu_1, \mu_2) \rangle$, $\mathsf{corner}(\mathsf{max}, (\mu_1, \mu_2), p) = (\mu, \mu)$, and $\mathsf{corner}(\mathsf{min}, (\mu_1, \mu_2), p) = (\mu', \mu')$, then

- Welldefined:
  $(\mu, \mu), (\mu', \mu') \in \langle (\mu_1, \mu_2) \rangle$
- Optimal up to index (1/2):
  $\forall 1 \leq j \leq n.\ \sum \{\!|\bar{\mu}(s_i) \mid 1 \leq i \leq j|\!\} \leq \sum \{\!|\mu(s_i) \mid 1 \leq i \leq j|\!\}$
- Optimal up to index (2/2):
  $\forall 1 \leq j \leq n.\ \sum \{\!|\mu'(s_i) \mid j \leq i \leq n|\!\} \leq \sum \{\!|\bar{\mu}(s_i) \mid j \leq i \leq n|\!\}$
- Maximal versus Minimal Computation (1/2):
  $\mathsf{corner}(\mathsf{min}, (\mu_1, \mu_2), p) = \mathsf{corner}(\mathsf{max}, (\mu_1, \mu_2), p')$
- Maximal versus Minimal Computation (2/2):
  $\mathsf{corner}(\mathsf{max}, (\mu_1, \mu_2), p) = \mathsf{corner}(\mathsf{min}, (\mu_1, \mu_2), p')$
- Correct Lower and Upper Bounds:
  $\sum \{\!|(\mu' \times p)(s_i) \mid s_i \in A|\!\} \leq \sum \{\!|(\bar{\mu} \times p)(s_i) \mid s_i \in A|\!\} \leq \sum \{\!|(\mu \times p)(s_i) \mid s_i \in A|\!\}$

Based on the corners of DIPDs defined above, we now present the algorithm MinMaxProbReach where we include explicitly the case of the minimal probability.

Definition 21 (Algorithm MinMaxProbReach)**.** If I is an IPTS, $F \subseteq \mathsf{states}(I)$ is a set of target states of I, $\bar{F} \subseteq \mathsf{states}(I)$ is the set of states of I from which no state in F can be reached, and $op \in \{\mathsf{min}, \mathsf{max}\}$ is an operation, then $\mathsf{MinMaxProbReach}(op, I, F, \bar{F}) = (\lim_{n \to \infty} \mathsf{pre}(op, I, F, \bar{F})^n(p_0))(\mathsf{istate}(I))$ is the resulting minimal/maximal probability using a fixed-point computation where

$$p_0 = \begin{cases} 0 \\ \quad \text{if } s \in \mathsf{states}(I) - F \\ 1 \\ \quad \text{if } s \in F \end{cases}$$

$$\mathsf{pre}(op, I, F, \bar{F})(p)(s) = \begin{cases} p(s) \\ \quad \text{if } s \in F \cup \bar{F} \\ op\{\mathsf{corner}(op, (\mu_1, \mu_2), p) \times p \\ \qquad \mid (s, a, (\mu_1, \mu_2)) \in \mathsf{steps}(I)\} \\ \quad \text{if } s \in \mathsf{states}(I) - (F \cup \bar{F}) \end{cases}$$

We now state the correctness of this algorithm w.r.t. the semantics of the probabilistic real-time queries $\mathcal{P}_{\mathsf{min}=?}(\cdot)$ and $\mathcal{P}_{\mathsf{max}=?}(\cdot)$.

Theorem 1 (Algorithm MinMaxProbReach)**.** If I is an IPTS, P is the PTS induced by I, $F \subseteq \mathsf{states}(I)$ is a set of target states of I, $ap \in \mathsf{lab}(I)(s)$ iff $s \in F$ (for each $s \in \mathsf{states}(I)$), and $\bar{F} \subseteq \mathsf{states}(I)$ is the set of states of I from which no state in F can be reached, then

- $\mathsf{MinMaxProbReach}(\mathsf{min}, I, F, \bar{F}) = \langle\!\langle \mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ ap), P \rangle\!\rangle$ and
- $\mathsf{MinMaxProbReach}(\mathsf{max}, I, F, \bar{F}) = \langle\!\langle \mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ ap), P \rangle\!\rangle$.

The same also holds for time bounded properties since these time bounds can be encoded beforehand.

For a better understanding of the algorithm, we provide an example.

Example 7 (Algorithm MinMaxProbReach)**.** Consider the IPTA from Figure C.1a. Note that the IPTA does not use clocks and, hence, the IPTS I induced by this IPTA has the same structure. MinMaxProbReach computes a sequence of probability vectors p_i until a fixed-point is reached.

We give names to the DIPDs of I.

$$
\begin{aligned}
(\mu_1^1, \mu_2^1) &= (\{s_1 \mapsto 0.4, s_2 \mapsto 0.3, s_3 \mapsto 0.2\}, \{s_1 \mapsto 0.6, s_2 \mapsto 0.7, s_3 \mapsto 0.8\}) \\
(\mu_1^2, \mu_2^2) &= (\{s_4 \mapsto 0.6, s_5 \mapsto 0.4\}, \{s_4 \mapsto 0.6, s_5 \mapsto 0.4\}) \\
(\mu_1^3, \mu_2^3) &= (\{s_4 \mapsto 0.7, s_5 \mapsto 0.3\}, \{s_4 \mapsto 0.7, s_5 \mapsto 0.2\}) \\
(\mu_1^4, \mu_2^4) &= (\{s_4 \mapsto 0.2, s_5 \mapsto 0.8\}, \{s_4 \mapsto 0.2, s_5 \mapsto 0.8\})
\end{aligned}
$$

We compute $\mathsf{MinMaxProbReach}(\mathsf{max}, I, \{s_4\}, \{s_5\})$.

$$
\begin{aligned}
p_0 &= \{s_0 \mapsto 0, s_1 \mapsto 0, s_2 \mapsto 0, s_3 \mapsto 0, s_4 \mapsto 1, s_5 \mapsto 0\} \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_0) \quad (\text{note that: } p_0(s_1) \geq p_0(s_2) \geq p_0(s_3)) \\
&= \left\{ \begin{array}{l} s_1 \mapsto \mathsf{min}(\mu_2^1(s_1), 1 - (0) - (0.3 + 0.2) = 0.5, \\ s_2 \mapsto \mathsf{min}(\mu_2^1(s_2), 1 - (0.5) - (0.2) = 0.3, \\ s_3 \mapsto \mathsf{min}(\mu_2^1(s_3), 1 - (0.5 + 0.3) - (0) = 0.2 \end{array} \right\} \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_0) \times p_0 = 0.5 \times 0 + 0.3 \times 0 + 0.2 \times 0 = 0 \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_0) \quad (\text{note that: } p_0(s_4) \geq p_0(s_5)) \\
&= \left\{ \begin{array}{l} s_4 \mapsto \mathsf{min}(\mu_2^2(s_4), 1 - (0) - (0.4) = 0.6, \\ s_5 \mapsto \mathsf{min}(\mu_2^2(s_5), 1 - (0.4) - (0) = 0.4 \end{array} \right\} \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_0) \times p_0 = 0.6 \times 1 + 0.4 \times 0 = 0.6 \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_0) \quad (\text{note that: } p_0(s_4) \geq p_0(s_5)) \\
&= \left\{ \begin{array}{l} s_4 \mapsto \mathsf{min}(\mu_2^3(s_4), 1 - (0) - (0.3) = 0.7, \\ s_5 \mapsto \mathsf{min}(\mu_2^3(s_5), 1 - (0.7) - (0) = 0.3 \end{array} \right\} \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_0) \times p_0 = 0.7 \times 1 + 0.3 \times 0 = 0.7 \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_0) \quad (\text{note that: } p_0(s_4) \geq p_0(s_5)) \\
&= \left\{ \begin{array}{l} s_4 \mapsto \mathsf{min}(\mu_2^4(s_4), 1 - (0) - (0.8) = 0.2, \\ s_5 \mapsto \mathsf{min}(\mu_2^4(s_5), 1 - (0.2) - (0) = 0.8 \end{array} \right\} \\
&\mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_0) \times p_0 = 0.2 \times 1 + 0.8 \times 0 = 0.2
\end{aligned}
$$

$$p_1 = \left\{ \begin{array}{l} s_0 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_0) \times p_0) = 0, \\ s_1 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_0) \times p_0) = 0.6, \\ s_2 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_0) \times p_0) = 0.7, \\ s_3 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_0) \times p_0) = 0.2, \\ s_4 \mapsto p_0(s_4) = 1, \\ s_5 \mapsto p_0(s_5) = 0 \end{array} \right\}$$

$\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_1)$ (note that: $p_1(s_2) \geq p_1(s_1) \geq p_1(s_3)$)

$$= \left\{ \begin{array}{l} s_2 \mapsto \mathsf{min}(\mu_2^1(s_2), 1 - (0) - (0.4 + 0.2) = 0.4, \\ s_1 \mapsto \mathsf{min}(\mu_2^1(s_1), 1 - (0.4) - (0.2) = 0.4, \\ s_3 \mapsto \mathsf{min}(\mu_2^1(s_3), 1 - (0.4 + 0.4) - (0) = 0.2 \end{array} \right\}$$

$\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_1) \times p_1 = 0.7 \times 0.4 + 0.6 \times 0.4 + 0.2 \times 0.2 = 0.56$

$\mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_0) \times p_0 = \mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_1) \times p_1$

$\mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_0) \times p_0 = \mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_1) \times p_1$

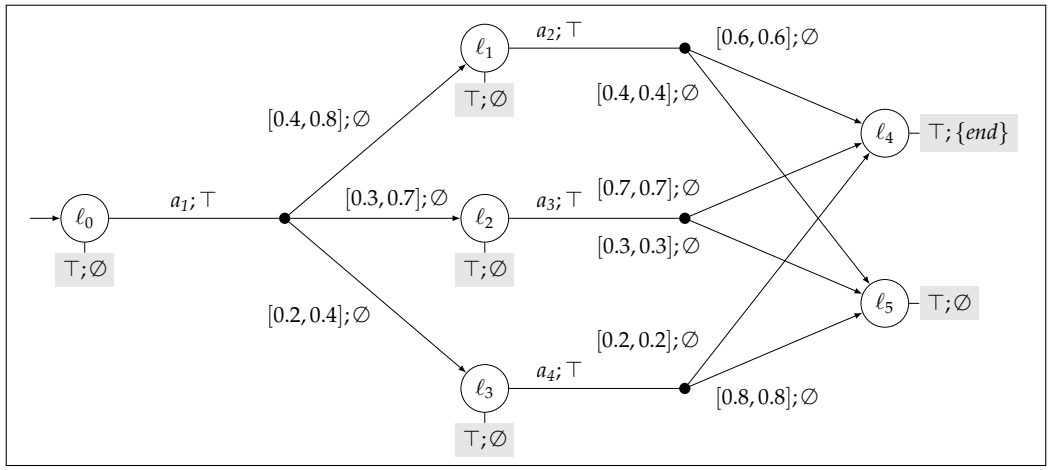
$\mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_0) \times p_0 = \mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_1) \times p_1$

$$p_2 = \left\{ \begin{array}{l} s_0 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^1, \mu_2^1), p_1) \times p_1) = 0.56, \\ s_1 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^2, \mu_2^2), p_1) \times p_1) = 0.6, \\ s_2 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^3, \mu_2^3), p_1) \times p_1) = 0.7, \\ s_3 \mapsto \mathsf{max}(\mathsf{corner}(\mathsf{max}, (\mu_1^4, \mu_2^4), p_1) \times p_1) = 0.2, \\ s_4 \mapsto p_1(s_4) = 1, \\ s_5 \mapsto p_1(s_5) = 0 \end{array} \right\}$$

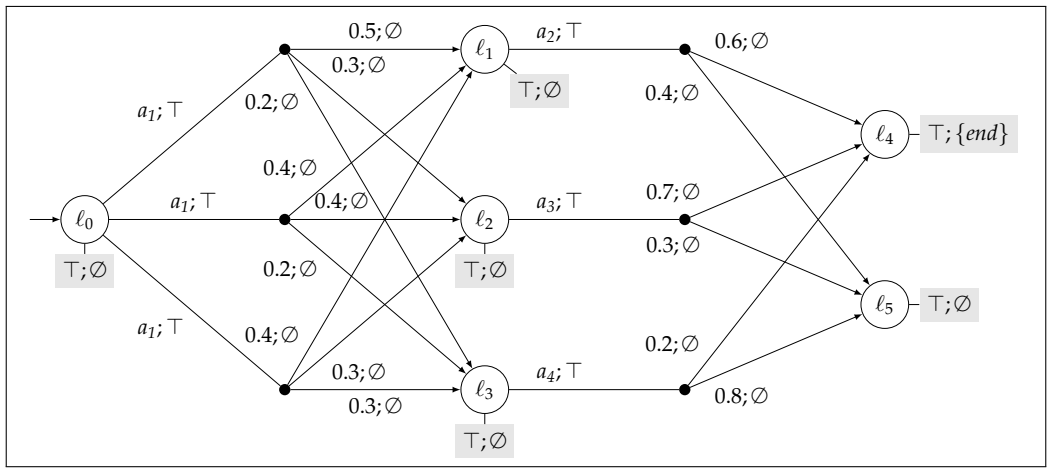
Note that the next probability vector p_3 is equal to p_2 and, hence, we have reached a fixed-point. The maximal probability to reach an *end* labeled state is therefore 0.56.

In the presented algorithm, target locations are sorted according to the current probability vector p_i. This sorting is not unique in general but it suffices to consider a single such ordering as all such orderings result in the same probability vector p_{i+1}. In principle, the algorithm could also consider all orderings generating the probability for each of these orderings and then determine the maximum/minimum probability for each state. Constructing the ordering is done to only consider a single ordering for efficiency.

Based on this observation, another analysis algorithm was presented in [6, Section 3.3.2, pp. 21] for computing again the minimal and maximal probability (w.r.t. the choice among the adversaries) to reach one state in a set of target states T from the initial state in the PTS. The idea of this alternative algorithm is to convert the IPTA into a PTA such that the PTS induced by the PTA can be analyzed using e.g. the stochastic games engine or the digital clocks engine of Prism using a simpler form of the presented algorithm MinMaxProbReach from above for the case without probability intervals. While each DIPD represents (often) an infinite set of PMFs, only the corners from above have to be considered. That is, one edge in the IPTA with DIPD (μ_1, μ_2) with n target states (i.e., n elements in $\mathsf{supp}((\mu_1, \mu_2))$) results in $n!$ edges in the resulting PTA where each PMF is obtained using $\mathsf{corner}(op, (\mu_1, \mu_2), p)$ where p determines a unique ordering by mapping each target location in that ordering to a decreasing probability. Also note that when all orderings are considered, then the



(a) An IPTA for Example 7.



(b) A PTA alternative for maximum/minimum probability for the IPTA from Figure C.1a.

Figure C.1: Visualizations of IPTA for analysis

value of *op* does not matter according to Lemma 3. For an example, see Figure C.1b where the PTA resulting from the IPTA from Figure C.1a is given. Note, in this particular example, (a) 3 PMFs of the $3! = 6$ PMFs are duplicates and (b) the same result is obtained for $op = \mathsf{min}$ and $op = \mathsf{max}$. We now define this construction.

Definition 22 (Operation IPTAtoPTA)**.** Every IPTA A_1 induces a unique *probabilistic timed automaton* (*PTA*) $\mathsf{IPTAtoPTA}(A_1) = A_2$ consisting of the following components.

- $\mathsf{locs}(A_2) = \mathsf{locs}(A_1)$,
- $\mathsf{iloc}(A_2) = \mathsf{iloc}(A_1)$,
- $\mathsf{acts}(A_2) = \mathsf{acts}(A_1)$,
- $\mathsf{clocks}(A_2) = \mathsf{clocks}(A_1)$,
- $\mathsf{invs}(A_2) = \mathsf{invs}(A_1)$,
- $(\ell_1, a, \psi, (\mu', \mu')) \in \mathsf{edges}(A_2)$ if

- $(\ell_1, a, \psi, (\mu_1, \mu_2)) \in \mathsf{edges}(A_1)$,
- $s' = (\ell'_1, \ldots, \ell'_n)$ is an ordering of $\mathsf{supp}((\mu_1, \mu_2))$, and
- $\mathsf{corner}(op, (\mu_1, \mu_2), \mathsf{pvector}(s')) = \mu'$,

- $\mathsf{aps}(A_2) = \mathsf{aps}(A_1)$, and
- $\mathsf{lab}(A_2) = \mathsf{lab}(A_1)$.

The presented construction is correct since the same maximal and minimal probabilities would be computed.

Theorem 2 (Correctness of PTA construction from IPTA)**.** If A_1 is an IPTA, I_1 is the IPTS induced by A_1, P_1 is the PTS induced by I_1, $\mathsf{IPTAtoPTA}(A_1) = A_2$ is the resulting PTA, P_2 is the PTS induced by A_2, and *ap* is an AP of A_1, then

- $\langle\!\langle \mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ ap), P_1 \rangle\!\rangle = \langle\!\langle \mathcal{P}_{\mathsf{min}=?}(\mathsf{F}\ ap), P_2 \rangle\!\rangle$ and
- $\langle\!\langle \mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ ap), P_1 \rangle\!\rangle = \langle\!\langle \mathcal{P}_{\mathsf{max}=?}(\mathsf{F}\ ap), P_2 \rangle\!\rangle$.

The same also holds for time bounded properties since these time bounds can be encoded beforehand.

D Translation from IPTGTS into IPTA

We now consider an example of a translation of an IPTGTS into an IPTA based on Definition 10 especially considering the construction of guards encoding priorities and target state invariants. For this purpose, we now define a custom IPTGTS, which is very simple w.r.t. the other modeling capabilities of IPTGTSs. See Figure D.1 and Figure D.2 for the components of this IPTGTS. The IPTA from Figure D.3 is obtained from the IPTGTS S_{pe} from Figure D.1 and Figure D.2 as follows.

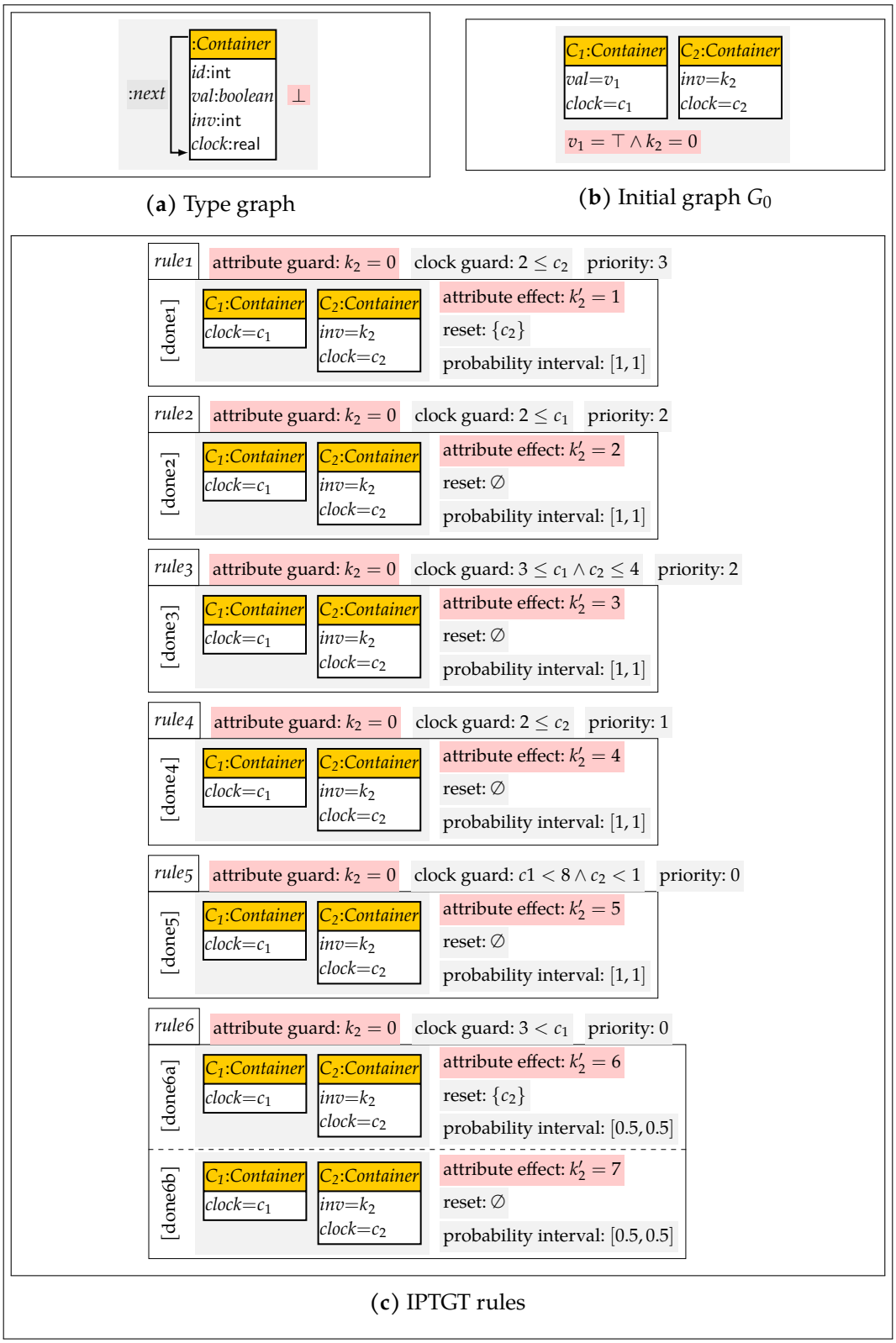
- The structural state space consists of the initial graph G_0 and seven steps to seven further graphs where the variable k_2 has the values 1 through 7.
- The global set of clocks is given by $\{c_1, c_2\}$.
- The two steps to graphs where k_2 has values 6 and 7 belong both to the IPGT rule σ_6 and they are therefore to be grouped together in the resulting IPTA.
- The reset sets of each of the seven steps is taken from the GT rule used.
- The probability intervals of each of the seven steps is taken from the GT rule used.
- Each of the seven additional graphs is labeled with the IPTGT AP $\phi_{\mathsf{pe:ap}}$.
- The invariants of each of the eight states can easily be derived from the IPTGT invariant $\phi_{\mathsf{pe:inv}}$.

The remaining problem is that guards need to be determined. Essentially, a step with lower priority should be disabled (via its guard) when a step with a higher priority could be taken. For this to work properly, the target state invariants of the higher priority steps also need to be taken into account. Also, it is to be noted that guards may not contain disjunctions: if a disjunction is obtained for some step subsequently, we duplicate steps such that the disjunction of the guards of these copies equals the computed guard. Due to such step duplications (for IPTGT rule σ_4), unsatisfiable guards (for IPTGT rule σ_1), and step groupings (for IPTGT rule σ_6), we obtain a different number of IPTA edges compared to the steps in the structural state space. For clarity, each location ℓ_i represents the graph where k_2 has the value i.

- Priority 3 steps:
  - IPTGT rule σ_1:
    The guard of the used IPTGT rule is $2 \leq c_2$. The reset set for the used GT rule is $\{c_2\}$. The invariant of the target graph for $k_2 = 1$ is $1 < c_2$. The conjunction of the guards of IPTA edges generated for steps with higher priority is $\top$. The guard of the edge should be $2 \leq c_2 \wedge 1 < 0 \wedge \top$. This guard is unsatisfiable. There is no resulting IPTA edge.
- Priority 2 steps:
  - IPTGT rule σ_2: The guard of the used IPTGT rule is $2 \leq c_1$. The reset set for the used GT rule is $\emptyset$. The invariant of the target graph for $k_2 = 2$ is $6 \leq c_1$. The conjunction of the guards of IPTA edges generated for steps with higher

priority is $\top$. The guard of the edge should be $2 \leq c_1 \wedge 6 \leq c_1 \wedge \top$. This guard can be simplified to $6 \leq c_1$. There is one resulting IPTA edge with guard $6 \leq c_1$.
  - IPTGT rule σ_3: The guard of the used IPTGT rule is $3 \leq c_1 \wedge c_2 \leq 4$. The reset set for the used GT rule is $\emptyset$. The invariant of the target graph for $k_2 = 3$ is $2 < c_2$. The conjunction of the guards of IPTA edges generated for steps with higher priority is $\top$. The guard of the edge should be $3 \leq c_1 \wedge c_2 \leq 4 \wedge 2 < c_2 \wedge \top$. This guard can be simplified to $3 \leq c_1 \wedge 2 < c_2 \leq 4$. There is one resulting IPTA edge with guard $3 \leq c_1 \wedge 2 < c_2 \leq 4$.
- Priority 1 steps:
  - IPTGT rule σ_4: The guard of the used IPTGT rule is $2 \leq c_2$. The reset set for the used GT rule is $\emptyset$. The invariant of the target graph for $k_2 = 4$ is $\top$. The conjunction of the guards of IPTA edges generated for steps with higher priority is $\neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4)$. The guard of the edge should be $2 \leq c_2 \wedge \top \wedge \neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4)$. This guard can be simplified to $c_1 < 3 \wedge 2 \leq c_2 \vee c_1 < 6 \wedge 2 \leq c_2 \leq 2 \vee c_1 < 6 \wedge 4 < c_2$. There are three resulting IPTA edges with guards $c_1 < 3 \wedge 2 \leq c_2$, $c_1 < 6 \wedge 2 \leq c_2 \leq 2$, and $c_1 < 6 \wedge 4 < c_2$.
- Priority 0 steps:
  - IPTGT rule σ_5: The guard of the used IPTGT rule is $c_1 < 8 \wedge c_2 < 1$. The reset set for the used GT rule is $\emptyset$. The invariant of the target graph for $k_2 = 5$ is $\top$. The conjunction of the guards of IPTA edges generated for steps with higher priority is $\neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4) \wedge \neg(c_1 < 3 \wedge 2 \leq c_2) \wedge \neg(c_1 < 6 \wedge 2 \leq c_2 \leq 2) \wedge \neg(c_1 < 6 \wedge 4 < c_2)$. The guard of the edge should be $c_1 < 8 \wedge c_2 < 1 \wedge \top \wedge \neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4) \wedge \neg(c_1 < 3 \wedge 2 \leq c_2) \wedge \neg(c_1 < 6 \wedge 2 \leq c_2 \leq 2) \wedge \neg(c_1 < 6 \wedge 4 < c_2)$. This guard can be simplified to $c_1 < 6 \wedge c_2 < 1$. There is one resulting IPTA edge with guard $c_1 < 6 \wedge c_2 < 1$.
  - IPTGT rule σ_6: The guard of the used IPTGT rule is $3 < c_1$. The reset set for the first used GT rule is $\{c_2\}$. The reset set for the second used GT rule is $\emptyset$. The invariant of the first target graph for $k_2 = 6$ is $c_1 \leq 4 \wedge c_2 < 10$. The invariant of the second target graph for $k_2 = 7$ is $c_1 < 4$. The conjunction of the guards of IPTA edges generated for steps with higher priority is $\neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4) \wedge \neg(c_1 < 3 \wedge 2 \leq c_2) \wedge \neg(c_1 < 6 \wedge 2 \leq c_2 \leq 2) \wedge \neg(c_1 < 6 \wedge 4 < c_2)$. The guard of the edge should be $3 < c_1 \wedge (c_1 \leq 4 \wedge 0 < 10) \wedge c_1 < 4 \neg(6 \leq c_1) \wedge \neg(3 \leq c_1 \wedge 2 < c_2 \leq 4) \wedge \neg(c_1 < 3 \wedge 2 \leq c_2) \wedge \neg(c_1 < 6 \wedge 2 \leq c_2 \leq 2) \wedge \neg(c_1 < 6 \wedge 4 < c_2)$. This guard can be simplified to $3 < c_1 < 4 \wedge c_2 < 2$. There is one resulting IPTA edge with guard $3 < c_1 < 4 \wedge c_2 < 2$.

Note that since $0 \leq c_1 = c_2 \leq 7$ in ℓ_0, the σ_2 step is possible in $[6, 7]$, the σ_3 step is possible in $[3, 4]$, the σ_4 step is possible in $[2, 2] \cup (4, 6)$, the σ_5 step is possible in $[0, 1)$, and the σ_6 step is possible in $\emptyset$.

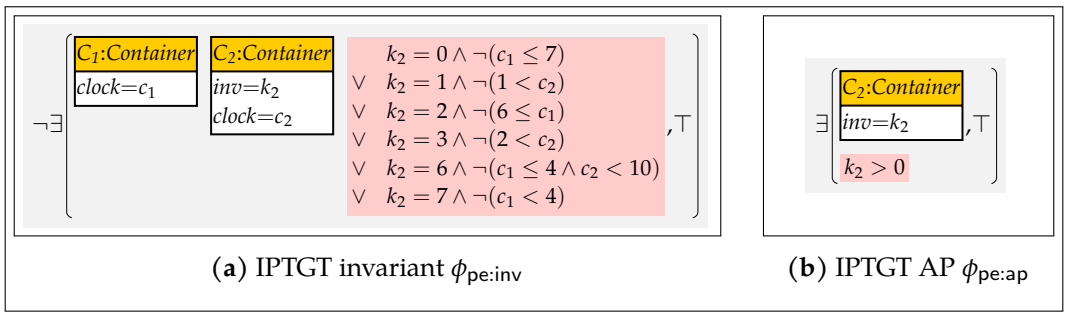


(a) Type graph

(b) Initial graph G_0

(c) IPTGT rules

Figure D.1: IPTGTS to be Translated (1/2)



(**a**) IPTGT invariant $\phi_{\text{pe:inv}}$ (**b**) IPTGT AP $\phi_{\text{pe:ap}}$

Figure D.2: IPTGTS to be Translated (2/2)

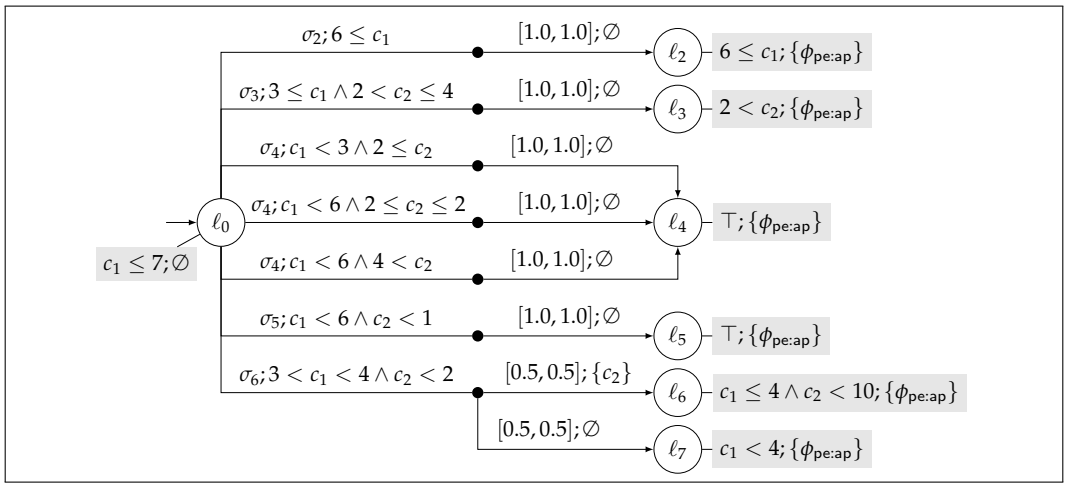


Figure D.3: IPTA resulting from IPTGTS from Figure D.1 and Figure D.2

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
133	978-3-86956-501-9	**Compositional analysis of probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
132	978-3-86956-482-1	**SandBlocks : Integration visueller und textueller Programmelemente in Live-Programmiersysteme**	Leon Bein, Tom Braun, Björn Daase, Elina Emsbach, Leon Matthes, Maximilian Stiede, Marcel Taeumel, Toni Mattis, Stefan Ramson, Patrick Rein, Robert Hirschfeld, Jens Mönig
131	978-3-86956-481-4	**Was macht das Hasso-Plattner-Institut für Digital Engineering zu einer Besonderheit?**	August-Wilhelm Scheer
130	978-3-86956-475-3	**HPI Future SOC Lab : Proceedings 2017**	Christoph Meinel, Andreas Polze, Karsten Beins, Rolf Strotmann, Ulrich Seibold, Kurt Rödszus, Jürgen Müller
129	978-3-86956-465-4	**Technical report : Fall Retreat 2018**	Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch, Tobias Friedrich, Erwin Böttinger, Christoph Lippert
128	978-3-86956-464-7	**The font engineering platform : collaborative font creation in a self-supporting programming environment**	Tom Beckmann, Justus Hildebrand, Corinna Jaschek, Eva Krebs, Alexander Löser, Marcel Taeumel, Tobias Pape, Lasse Fister, Robert Hirschfeld
127	978-3-86956-463-0	**Metric temporal graph logic over typed attributed graphs : extended version**	Holger Giese, Maria Maximova, Lucas Sakizloglou, Sven Schneider
126	978-3-86956-462-3	**A logic-based incremental approach to graph repair**	Sven Schneider, Leen Lambers, Fernando Orejas
125	978-3-86956-453-1	**Die HPI Schul-Cloud : Roll-Out einer Cloud-Architektur für Schulen in Deutschland**	Christoph Meinel, Jan Renz, Matthias Luderich, Vivien Malyska, Konstantin Kaiser, Arne Oberländer
124	978-3-86956-441-8	**Blockchain : hype or innovation**	Christoph Meinel, Tatiana Gayvoronskaya, Maxim Schnjakin

ISBN 978-3-86956-502-6
ISSN 1613-5652