Consistency in Parametric Interval Probabilistic Timed Automata

Étienne André
[1] *Université Paris 13, Sorbonne Paris Cité, LIPN*
CNRS, UMR 7030, F-93430, Villetaneuse, France
[2] *École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France*

Benoît Delahaye
Université de Nantes
LINA UMR CNRS 6241, Nantes, France

***Abstract*—We propose a new abstract formalism for probabilistic timed systems, Parametric Interval Probabilistic Timed Automata, based on an extension of Parametric Timed Automata and Interval Markov Chains. In this context, we consider the consistency problem that amounts to deciding whether a given specification admits at least one implementation. In the context of Interval Probabilistic Timed Automata (with no timing parameters), we show that this problem is decidable and propose a constructive algorithm for its resolution. We show that the existence of parameter valuations ensuring consistency is undecidable in the general context, but still propose a semi-algorithm that resolves it whenever it terminates.**

***Keywords*-parametric verification; timed probabilistic systems; parametric probabilistic timed automata;**

I. Introduction

Motivation: Nowadays, automata-based modeling and verification methods are mainly used in two different ways: for designing digital systems based on (mostly informal) specifications expressed by the end-users of these systems or from the knowledge designers have of their environment; and in order to abstract existing (not necessarily software) systems that are too complex to comprehend in their entirety. In both cases the complexity of the systems being designed calls for increasingly expressive abstraction artifacts such as time and probabilities. Timed automata [AD94] are a widely recognized modeling formalism for reasoning about real-time systems. This modeling formalism, based on finite control automata equipped with clocks, which are real-valued variables which increase uniformly at the same rate, has been extended to the probabilistic framework in [GJ95], [KNSS02]. In this context, discrete actions are replaced with probabilistic discrete distributions over discrete actions, allowing to model uncertainties in the system's behavior. This formalism has been applied to a number of case studies [KNPS06].

Unfortunately, building a system model based either on imprecise specifications or on imprecise observations often requires to fix arbitrarily a number of constants in the model, which are then calibrated by a fastidious comparison of the model behavior and the expected behavior. This is the case for instance for timing constants or transition probability values. In order to incorporate these uncertainties in the model and to develop automatic calibration, more abstract formalisms have been introduced separately in the timed setting and in the probabilistic setting.

In the timed setting, *parametric timed automata* [AHV93] allow using parameter variables in the guards of timed transitions in order to account for the uncertainty on their values. Parametric probabilistic timed automata were proposed in [AFS13] to answer the following question: given a parameter valuation, what are other valuations preserving the same minimum and maximum probabilities for reachability properties as the reference valuation? Parametric probabilistic timed automata were then given a symbolic semantics in [JK14]; a method has been proposed in that same work to synthesize optimal parameter valuations to maximize or minimize the probability of reaching a discrete location.

In the pure probabilistic setting, Interval Markov Chains (IMCs for short) have been introduced [JL91] to take into account imprecision in the transition probabilities. IMCs extend Markov Chains by allowing to specify intervals of possible probabilities on transitions instead of exact values. Methods have then been developed to decide whether there exists Markov Chains with concrete probability values that match the intervals specified in a given IMC [DLL+12].

Contribution: In this paper, we propose to combine both abstraction approaches into a single specification theory: Parametric Interval Probabilistic Timed Automata (PI$\mathbb{P}$TAs for short). In this setting, parameters can be used in order to abstract timed constants on transition guards while intervals can be used to abstract imprecise transition probabilities. As for IMCs, it is important to be able to decide whether the probability intervals that are specified in a model allow defining coherent probability distributions (i. e., can be matched in a real-life implementation). This is called the consistency problem. In the context of Interval Probabilistic Timed Automata with no timing parameters (I$\mathbb{P}$TAs for short), we propose an algorithm that resolves this problem. In our setting, since the behavior of the system is conditioned by the calibration of parameter values, it is therefore necessary to decide whether there exist parameter

This work is partially supported by the ANR national research program PACS (ANR-14-CE28-0002). This is the author version of the paper of the same name published in the proceedings of the Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning (TIME 2016). The final version is available at dx.doi.org/10.1109/TIME.2016.19.

values that ensure consistency of the resulting model (and synthesize these values when this is possible). We show that the existence of such parameter valuations is undecidable in the general context of PI$\mathbb{P}$TAs. Still, we propose a semi-algorithm that synthesizes, whenever it terminates, the set of parameter values that ensure consistency of the resulting I$\mathbb{P}$TA.

Outline: We start Section II with preliminary definitions and then introduce the concepts of I$\mathbb{P}$TAs and PI$\mathbb{P}$TAs. In Section III, we study the consistency problem for I$\mathbb{P}$TAs and propose a constructive algorithm based on the zone-graph construction that decides whether an I$\mathbb{P}$TA is consistent and produces an implementation if one exists. In Section IV, we move to the general problem of consistency of PI$\mathbb{P}$TAs. We first show that this problem is undecidable in general and then propose a semi-algorithm that synthesizes, whenever it terminates, the set of parameter values ensuring consistency of the resulting I$\mathbb{P}$TA. Finally, Section V concludes the paper.

II. Preliminaries

A. Clocks, Parameters and Constraints

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote the sets of non-negative integers, integers, non-negative rational numbers and non-negative real numbers respectively. Given an arbitrary set S, we write $\mathsf{Dist}(S)$ for the set of probabilistic distributions over S.

Throughout this paper, let $X = \{x_1, \dots, x_H\}$ be a set of *clocks*, i. e., real-valued variables that evolve at the same rate, and $\Gamma = \{\gamma_1, \dots, \gamma_M\}$ be a set of *parameters*, i. e., unknown constants.

A clock valuation is a function $w : X \to \mathbb{R}_+$. We identify a clock valuation w with the *point* $(w(x_1), \dots, w(x_H))$. We write $\vec{0}$ for the valuation that assigns 0 to each clock. Given $d \in \mathbb{R}_+$, $w+d$ denotes the valuation such that $(w+d)(x) = w(x) + d$, for all $x \in X$. Given $\rho \subseteq X$, we define $[w]_\rho$ as the clock valuation obtained by resetting the clocks in ρ and keeping other clocks unchanged.

A parameter *valuation* v is a function $v : \Gamma \to \mathbb{Q}_+$. We identify a parameter valuation v with the *point* $(v(\gamma_1), \dots, v(\gamma_M))$.

In the following, we assume $\prec \in \{<, \leq\}$ and $\sim \in \{<, \leq, \geq, >\}$. *lt* denotes a linear term over $X \cup \Gamma$ of the form $\sum_{1 \leq i \leq H} \alpha_i x_i + \sum_{1 \leq j \leq M} \beta_j \gamma_j + d$, with $x_i \in X$, $\gamma_j \in \Gamma$, and $\alpha_i, \beta_j, d \in \mathbb{Z}$. *plt* denotes a parametric linear term over Γ, that is a linear term without clocks ($\alpha_i = 0$ for all i). A *constraint* C over $X \cup \Gamma$ is a conjunction of inequalities of the form $lt \sim 0$ (i. e., a convex polyhedron). Given a parameter valuation v, $v(C)$ denotes the constraint over X obtained by replacing each parameter γ in C with $v(\gamma)$. Likewise, given a clock valuation w, $w(v(C))$ denotes the expression obtained by replacing each clock x in $v(C)$ with $w(x)$. We say that v *satisfies* C, denoted by $v \models C$, if the set of clock valuations satisfying $v(C)$ is nonempty. Given a parameter valuation v and a clock valuation w, we denote by $w|v$ the valuation over $X \cup \Gamma$ such that for all clocks x, $w|v(x) = w(x)$ and for all parameters γ, $w|v(\gamma) = v(\gamma)$. We use the notation $w|v \models C$ to indicate that $w(v(C))$ evaluates to true. We say that C is *satisfiable* if $\exists w, v$ s. t.$w|v \models C$. We define the *time elapsing* of C, denoted by $C^{\nearrow}$, as the constraint over X and Γ obtained from C by delaying all clocks by an arbitrary amount of time. Given $\rho \subseteq X$, we define the *reset* of C, written $[C]_\rho$, as the constraint obtained from C by resetting the clocks in ρ, and keeping the other clocks unchanged. We denote by $C{\downarrow_\Gamma}$ the projection of C onto Γ, i. e., obtained by eliminating the clock variables (e. g., using the Fourier-Motzkin algorithm).

A *guard* g is a constraint over $X \cup \Gamma$ defined by inequalities of the form $x \sim z$, where z is either a parameter or a constant in $\mathbb{Z}$.

A *zone* is a polyhedron over a set of clocks in which all constraints on variables are of the form $x \sim k$ (rectangular constraints) or $x_i - x_j \sim k$ (diagonal constraints), where $x_i \in X$, $x_j \in X$ and k is an integer. Operations on zones are well-documented (see e. g., [BY03]).

A *parametric zone* is a convex polyhedron over $X \cup \Gamma$ in which all constraints on variables are of the form $x \sim plt$ (parametric rectangular constraints) or $x_i - x_j \sim plt$ (parametric diagonal constraints), where $x_i \in X$, $x_j \in X$ and *plt* is a parametric linear term over Γ. We denote the set of all parametric zones by $\mathcal{Z}$.

B. Timed Probabilistic Systems

We review the definition of timed probabilistic systems, as defined in [KNSS02]. A *timed probabilistic system (TPS)* is a tuple $\mathcal{T} = (S, s_0, \Sigma, \Rightarrow)$ where S is a set of *states*, $s_0 \in S$ is the *initial state*, Σ is a finite set of *actions*, and $\Rightarrow \subseteq S \times \mathbb{R}_+ \times \Sigma \times \mathsf{Dist}(S)$ is a *probabilistic transition relation*.

C. Probabilistic Timed Automata

Probabilistic timed automata [GJ95], [KNSS02] are an extension of classical timed automata [AD94] with discrete probability distributions

1) Syntax:

Definition 1. *A Probabilistic Timed Automaton ($\mathbb{P}$TA) $\mathcal{P}$ is a tuple $(\Sigma, L, l_0, X, prob)$, where: i) Σ is a finite set of actions, ii) L is a finite set of locations, iii) $l_0 \in L$ is the initial location, iv) X is a finite set of clocks, v) $prob$ is a* probabilistic edge relation *consisting of elements of the form (l, g, a, μ), where $l \in L$, g is a constraint on the clocks X, $a \in \Sigma$, and $\mu \in \mathsf{Dist}(2^X \times L)$.*

Note that we use no invariant; this is an important condition for the correctness of our techniques. However, invariants can be eliminated (moved to the guards prior to the transition), following classical techniques defined for (probabilistic) timed automata.

We use the following conventions for the graphical representation of probabilistic timed automata: locations are represented by nodes, within which name of the location is written; probabilistic edges are represented by arcs from locations, labeled by the associated guard and action, and which split into multiple arcs, each of which leads to a location and which is labeled by a set of clocks to be reset to 0 and a probability (probabilistic edges which correspond to probability 1 are illustrated by a single arc from location to location).

Example 1. *1a presents an example of a $\mathbb{P}$TA with two clocks x and y. For example, l_0 can be exited whenever $y < 2$; then, with probability 0.4 the target location becomes l_2, resetting x; or with probability 0.6 the target location is l_1, resetting y. The transition from l_2 can be explained similarly.*

2) Semantics of $\mathbb{P}$TAss: A $\mathbb{P}$TA can be interpreted as an infinite TPS. Due to the continuous nature of clocks, the underlying TPS has uncountably many states, and is uncountably branching.

Definition 2 (Concrete semantics of a $\mathbb{P}$TA)**.** *Given a $\mathbb{P}$TA $\mathcal{P} = (\Sigma, L, l_0, X, prob)$, the concrete semantics of $\mathcal{P}$ is given by the timed probabilistic system $\mathcal{T}_{\mathcal{P}} = (S, s_0, \Sigma, \Rightarrow)$, with*

- $S = \{(l, w) \in L \times \mathbb{R}_+^H\}$, $s_0 = (l_0, \vec{0})$
- $((l, w), d, a, \mu) \in \Rightarrow$ *if both of the following conditions hold:*
  - *time elapse: $\forall d' \in [0, d], (l, w + d') \in S$, and*
  - *edge traversal: there exists a probabilistic edge $e = (l, g, a, \eta) \in prob$ such that $w + d \models g$ and, for each $l' \in L$ and $\rho \subseteq X$, $\eta(\rho, l') = \mu(l', [w + d]_\rho)$.*

Note that, due to the fact that we have no invariants, the first condition (time elapse) is always trivially true.

D. Parametric Interval Probabilistic Timed Automata

In this section, we introduce basic definitions for *(parametric) interval probabilistic timed automata*, that extend (parametric) probabilistic timed automata by providing *intervals* for transition probabilities instead of exact probability values. In the spirit of (parametric) Interval Markov Chains [Del15], [DLP16], (parametric) interval probabilistic timed automata are used for specifying potentially infinite families (sets) of probabilistic timed automata – those whose exact probability values match the specified intervals – with a finite structure of similar form.

1) Syntax: Given an arbitrary set S, we call an *interval distribution* over S a function Υ that assigns to each element of S an interval of probabilities $[a, b] \subseteq [0, 1]$. Intuitively, an interval distribution Υ over S represents the set of all distributions $\mu \in \mathsf{Dist}(S)$ that assign to each element $s \in S$ a probability $\mu(s)$ such that $\mu(s) \in \Upsilon(s)$. Formally, we define the implementation of an interval distribution as follows.

Definition 3 (Implementation of an interval distribution)**.** *Let S be an arbitrary set. Given an interval distribution $\Upsilon \in \mathsf{Int}_{[0,1]}(S)$, $\mu \in \mathsf{Dist}(S)$ is an* implementation *of Υ, written $\mu \in \Upsilon$ iff, for all $s \in S$, we have $\mu(s) \in \Upsilon(s)$.*

In the rest of the paper, we write $\mathsf{Int}_{[0,1]}(S)$ for the set of interval distributions over S. We now move to the definition of (parametric) interval probabilistic timed automata.

Definition 4. *A Parametric Interval Probabilistic Timed Automaton (PI$\mathbb{P}$TA) $\mathcal{PIP}$ is a tuple $(\Sigma, L, l_0, X, \Gamma, \mathbb{I})$, where: i) Σ is a finite set of actions, ii) L is a finite set of locations, iii) $l_0 \in L$ is the initial location, iv) X is a finite set of clocks, v) Γ is a finite set of parameters, vi) $\mathbb{I}$ is an* interval-valued probabilistic edge relation *consisting of elements of the form (l, g, a, Υ), where $l \in L$, g is a guard, $a \in \Sigma$, and $\Upsilon \in \mathsf{Int}_{[0,1]}(2^X \times L)$ is an interval distribution.*

Given a PI$\mathbb{P}$TA $\mathcal{PIP} = (\Sigma, L, l_0, X, \Gamma, \mathbb{I})$ and a parameter valuation v, the *valuation* of $\mathcal{PIP}$ with v, written $v(\mathcal{PIP})$, is an Interval Probabilistic Timed Automaton (I$\mathbb{P}$TA) $\mathcal{IP} = (\Sigma, L, l_0, X, \mathbb{I}')$, where $\mathbb{I}'$ is obtained by replacing within $\mathbb{I}$ any occurrence of a parameter γ with $v(\gamma)$ and removing all transitions (l, g, a, Υ) such that $v(g) \equiv \bot$ (technically, this latter part is not strictly speaking necessary, but it syntactically reduces a bit the model).

Remark that I$\mathbb{P}$TAs are very similar to $\mathbb{P}$TAs: the only difference is that probabilistic edges are labeled with intervals instead of exact probability values.

Once a parameter valuation is fixed, the resulting I$\mathbb{P}$TA represents a potentially infinite set of $\mathbb{P}$TAs. In order to relate a given I$\mathbb{P}$TA with the $\mathbb{P}$TA it represents, we use the notion of *implementation* defined hereafter. This notion is similar to the one defined in the context of (parametric) Interval Markov Chains [Del15], [DLP16]. Remark that a $\mathbb{P}$TA implementing an I$\mathbb{P}$TA needs to conserve the exact same clocks, guards and resets.

Definition 5 (Implementation of an I$\mathbb{P}$TA)**.** *Let $\mathcal{P} = (\Sigma, L, l_0, X, prob)$ be a $\mathbb{P}$TA and $\mathcal{IP} = (\Sigma, L', l'_0, X, \mathbb{I})$ be an I$\mathbb{P}$TA.*

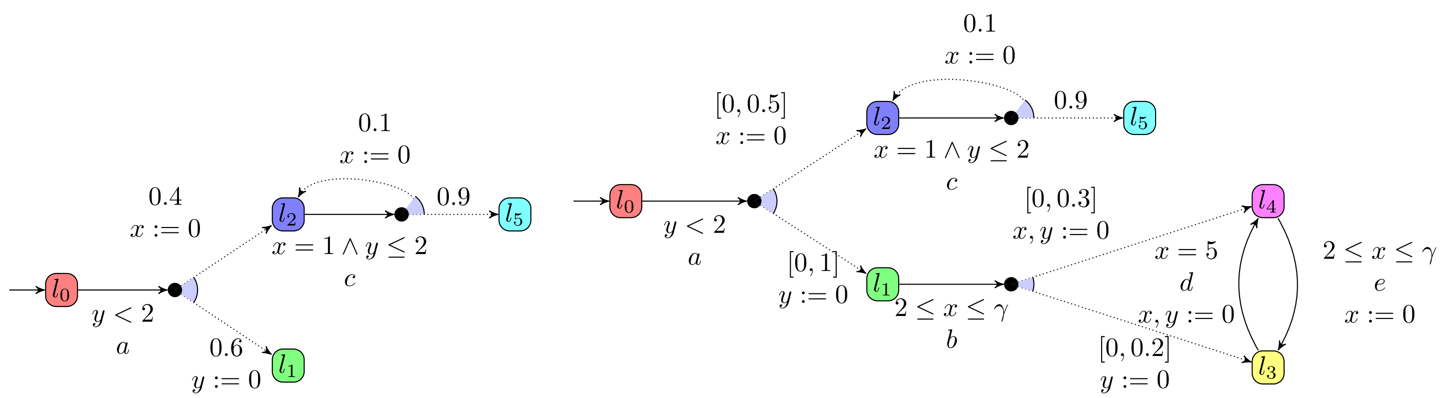
We say that $\mathcal{P}$ is an implementation of $\mathcal{IP}$, written $\mathcal{P} \models \mathcal{IP}$, iff there exists a relation $\mathcal{R} \subseteq L' \times L$, called an implementation relation *s. t. $(l'_0, l_0) \in \mathcal{R}$ and, whenever $(l', l) \in \mathcal{R}$, we have*

- $\forall (l', g', a, \mu) \in prob, \exists (l, g', a, \Upsilon) \in \mathbb{I}$ *s. t.* $\mu \preceq_{\mathcal{R}} \Upsilon$, *and*
- $\forall (l, g, a, \Upsilon) \in \mathbb{I}, \exists (l', g, a, \mu) \in prob$ *s. t.* $\mu \preceq_{\mathcal{R}} \Upsilon$.

where $\mu \preceq_{\mathcal{R}} \Upsilon$ iff $\exists \delta \in \mathsf{Dist}(L' \times L)$ s. t.

- $\forall (\rho', l') \in 2^X \times L', \mu(\rho', l') > 0 \Rightarrow \sum_{l \in L} (\delta(l', l)) = 1$,
- $\forall (\rho, l) \in 2^X \times L, \sum_{l' \in L'} (\mu(\rho, l') \cdot \delta(l', l)) \in \Upsilon(\rho, l)$, *and*
- $\delta(l', l) > 0 \Rightarrow (l', l) \in \mathcal{R}$.

Given an I$\mathbb{P}$TA, deciding whether the family it represents is nonempty is a nontrivial problem. Indeed, the inter-



(a) A $\mathbb{P}$TA (b) A PI$\mathbb{P}$TA

Figure 1: Examples

val distributions used throughout its structure could represent contradictory constraints on the transition probabilities, therefore preventing any $\mathbb{P}$TA from implementing it.

Definition 6 (Consistency of an I$\mathbb{P}$TA)**.** *An I$\mathbb{P}$TA is consistent if it admits at least one implementation.*

Example 2. *Consider the PI$\mathbb{P}$TA $\mathcal{PIP}$ given in Figure 1b, and containing a single parameter γ. When the interval associated with a distribution is reduced to a point (e. g., $[0.9, 0.9]$ from l_2 to l_5), we simply represent it using its punctual value (i. e., 0.9). When a distribution is made of a single target location with probability 1, we simply omit the distribution (e. g., between l_3 to l_4).*

Let v_1 be the parameter valuation such that $v_1(\gamma) = 1$. In the I$\mathbb{P}$TA $v_1(\mathcal{PIP})$, the transition outgoing from l_1 can never be taken, as its guard becomes $2 \leq x \leq 1$, which is unsatisfiable. Then, it is clear that the $\mathbb{P}$TA $\mathcal{P}$ given in Figure 1a is an implementation of $v_1(\mathcal{PIP})$. As a consequence, $v_1(\mathcal{PIP})$ is a consistent I$\mathbb{P}$TA.

An important problem is therefore to decide whether a given I$\mathbb{P}$TA is consistent, which we address in the next section.

III. The Consistency Problem for I$\mathbb{P}$TAs

In this section, we address the problem of deciding whether a given I$\mathbb{P}$TA is consistent. Unlike in the context of IMCs, where it is proven that a given IMC is consistent iff it admits an implementation with the same structure, a given I$\mathbb{P}$TA can be consistent but still not admit any implementation that respects its structure. Since transitions can be removed because their guard becomes unsatisfiable due to parameter valuations, the structure of implementations can differ from the one of the specification. Algorithms such as those proposed for deciding consistency of (p)IMCs in [DLP16] therefore cannot be directly adapted to the I$\mathbb{P}$TAs setting as they are dependent on this property.

Fortunately, the operational semantics of I$\mathbb{P}$TAs can be expressed in terms of Interval Markov Decision Processes (IMDPs), which are similar to IMCs and satisfy the same structural properties regarding consistency. We therefore propose an algorithm for deciding consistency of I$\mathbb{P}$TAs based on the consistency of their symbolic IMDP semantics. We start with preliminary definitions on IMDPs, then formally define the symbolic semantics of I$\mathbb{P}$TAs and finally propose an algorithm for deciding whether a given I$\mathbb{P}$TA is consistent.

A. Preliminary Definitions

An IMDP is a tuple (S, s_0, Σ, T) where S is a set of states, $s_0 \in S$ is the initial state, Σ is a finite set of actions and $T \subseteq S \times \Sigma \times \mathsf{Int}_{[0,1]}(S)$ is a probabilistic (interval) transition relation.

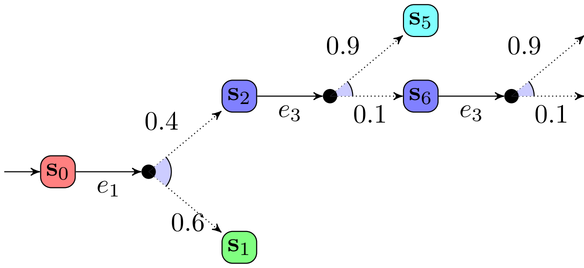
Example 3. *Figure 2b depicts an example of an IMDP. Just as for I$\mathbb{P}$TAs, when the interval associated with a distribution is reduced to a point (e. g., $[0.9, 0.9]$ from $\mathbf{s}_2$ to $\mathbf{s}_5$), we simply represent it using its punctual value (i. e., 0.9). When a distribution is made of a single target location with probability 1, we simply omit the distribution (e. g., between $\mathbf{s}_3$ to $\mathbf{s}_4$).*

An MDP is an IMDP such that for each $(s, a, [m, n]) \in T$, we have $m = n$, and for each $s \in S$, $\sum_{(s,a,s',[m,n]) \in T} m = 1$.

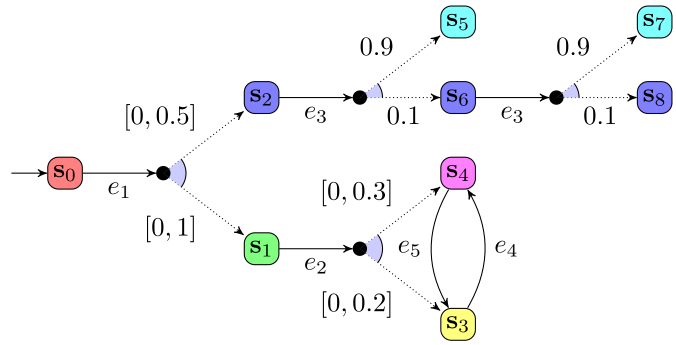
Example 4. *Figure 2a depicts an example of an MDP.*

Definition 7 (implementation of an IMDP)**.** *Let $\mathcal{IM} = (S, s_0, \Sigma, T)$ be an IMDP. Let $\mathcal{M} = (S', s'_0, \Sigma, T')$ be an MDP. We say that $\mathcal{M}$ is an implementation of $\mathcal{IM}$, written $\mathcal{M} \models \mathcal{IM}$, if $\exists \mathcal{R} \subseteq S' \times S$ s. t. $(s'_0, s_0) \in \mathcal{R}$ and $(s', s) \in \mathcal{R}$ if*

- *$\forall (s', a, \mu) \in T', \exists (s, a, I) \in T$ s. t. $\mu \preceq_{\mathcal{R}} I$, and*
- *$\forall (s, a, I) \in T, \exists (s', a, \mu) \in T'$ s. t. $\mu \preceq_{\mathcal{R}} I$.*



(a) An example of an MDP



(b) An example of an IMDP

Figure 2: Examples

As for IPTAs, we say that an IMDP is *consistent* iff it admits at least one implementation.

Example 5. *The IMDP given in Figure 2b admits no implementation: indeed, on the (single) transition labeled with* e_2*, no valuation of the two intervals* $[0, 0.3]$ *and* $[0, 0.2]$ *is such that the sum of both valuations is equal to 1. In addition, the transition from* s_0 *to* s_1 *cannot be eliminated by assigning a 0-probability to that target state; although this would be compatible with the interval (*$0 \in [0, 1]$*), the second interval (to* s_2*) does not accept a 1-probability since its probability must be within* $[0, 0.5]$.

As said above IMDPs satisfy the same structural property as IMCs concerning implementations: they are consistent iff they admit at least one implementation that respects their structure. This result is formalized in the following lemma.

Lemma 1 (structure of an implementation)**.** *An IMDP* $\mathcal{IM}$ *is consistent iff there exists an MDP* $\mathcal{M}$ *with the same structure s. t.* $\mathcal{M} \models \mathcal{IM}$.

Proof:
Let $\mathcal{IM} = (S, s_0, \Sigma, T)$ be an IMDP.

One direction of this result is trivial: if there exists an MDP $\mathcal{M}$ with the same structure as $\mathcal{IM}$ s. t. $\mathcal{M} \models \mathcal{IM}$, then $\mathcal{IM}$ is clearly consistent.

The reverse implication is more involved. Assume that $\mathcal{IM}$ is consistent, i. e., there exists an MDP $\mathcal{M} = (S', s'_0, \Sigma, T')$, with no assumption on its structure, such that $\mathcal{M} \models \mathcal{IM}$. We then have to build an MDP $\mathcal{M}^* = (S, s_0, \Sigma, T^*)$ such that $\mathcal{M}^*$ and $\mathcal{M}$ have the same structure. Observe that S and s_0 must be identical to that of $\mathcal{IM}$ because they have the same structure.

Let $\mathcal{R}$ be the relation witnessing that $\mathcal{M} \models \mathcal{IM}$ and let $f : S \to S'$ be a partial function that associates to each state in $\mathcal{IM}$ one of the states from $\mathcal{M}$ that contributes to its implementation, if any. Formally, for all $s \in S$, if $f(s)$ is defined then $(f(s), s) \in \mathcal{R}$.

The transition relation T^* of $\mathcal{M}^*$ is constructed as follows: For each state s that is implemented, i. e., such that $f(s)$ is defined, and probabilistic interval transition $(s, a, I) \in T$ in $\mathcal{IM}$, we build a corresponding transition (s, a, μ^I) in $\mathcal{M}^*$ from the transitions in $\mathcal{M}$ that implement (s, a, I). States that are not implemented do not serve for consistency and are therefore not considered.

Formally, let $(s_1, a, I) \in T$ be a probabilistic interval transition in $\mathcal{IM}$. From Definition 7, we know that there exists $(f(s_1), a, \mu) \in T'$ s. t. $\mu \preceq_{\mathcal{R}} I$. Let δ be the function given by $\mu \preceq_{\mathcal{R}} I$. The distribution μ^I is then constructed as follows: for all $s_2 \in S$, let $\mu^I(s_2) = \sum_{s' \in S'} \mu(s') \cdot \delta(s', s_2)$.

By definition of δ, observe that $\mu^I(s_2) \in I(s_2)$ for all $s_2 \in S$ and that, whenever $\mu^I(s_2) > 0$, $f(s_2)$ is defined.

Clearly, $\mathcal{M}^*$ is therefore an implementation of $\mathcal{IM}$, with witnessing relation $\mathcal{R}^*$ the identity relation on the set of states $s \in S$ such that $f(s)$ is defined. ∎

B. A Symbolic Semantics for IPTAs

We equip IPTAs with a symbolic semantics, defined below. Basically, it is inline with the symbolic semantics defined for timed automata, with the addition of probabilistic intervals on the edges; as a consequence, the semantics becomes not an LTS, but an IMDP.

Definition 8 (Symbolic semantics of an IPTA)**.** *Given an IPTA* $\mathcal{IP} = (\Sigma, L, l_0, X, \mathbb{I})$*, the symbolic semantics of* $\mathcal{IP}$ *is given by the IMDP* $(\mathbf{S}, \mathbf{s}_0, \Sigma, T)$*, with*

- $\mathbf{S} = \{(l, C) \in L \times \mathcal{Z}\}$, $\mathbf{s}_0 = (l_0, (\bigwedge_{1 \leq i \leq H} x_i = 0)^{\nearrow})$,
- $(\mathbf{s}, e, \Upsilon') \in T$ *if* $e = (l, g, a, \Upsilon) \in \mathbb{I}$ *and for all* $l' \in L$*, for all* $\rho \subseteq X$ *such that* $\Upsilon(l', \rho) > 0$*,* $C' = \big([C \wedge g]_\rho\big)^{\nearrow}$*, and* $\Upsilon'((l', C')) = \Upsilon(l', \rho)$.

Observe that, whenever an IPTA has no probabilistic choice, then the IMDP becomes a labeled transition system, and the symbolic semantics matches that of timed automata given in the form of a zone graph [BY03]. It is well-know that the zone graph of a timed automaton can have an infinite number of states; however, applying the classical k-extrapolation (that basically splits zones between a part where the clock constraints are smaller or equal to k and a part where constraints are larger than k, where k is the

State	Location	C
$\mathbf{s}_0$	l_0	$x = y \wedge x \geq 0$
$\mathbf{s}_1$	l_1	$0 \leq x - y < 2 \wedge y \geq 0$
$\mathbf{s}_2$	l_2	$0 \leq y - x < 2 \wedge x \geq 0$
$\mathbf{s}_5$	l_5	$0 \leq y - x \leq 1 \wedge x \geq 1$
$\mathbf{s}_6$	l_2	$1 \leq y - x \leq 2 \wedge x \geq 0$
$\mathbf{s}_7$	l_5	$y \geq 2 \wedge y = x + 1$
$\mathbf{s}_8$	l_2	$y \geq 2 \wedge y = x + 2$

Table I: Description of the states in Figure 2a

largest integer-constant in the timed automaton) yields termination (see, e. g., [BBLP06]). In the following, we apply the classical k-extrapolation to the symbolic constraints of the semantics of an $\mathbb{IP}$TA $\mathcal{IP}$, and therefore the number of states in the IMDP described in Definition 8 is finite. We refer to the symbolic semantics of $\mathcal{IP}$ as the *probabilistic zone graph* of $\mathcal{IP}$.

Remark that the probabilistic zone graph is defined for $\mathbb{IP}$TAs in the form of an IMDP; a $\mathbb{P}$TA can be understood as an $\mathbb{IP}$TA, and its associated zone graph becomes an MDP.

Example 6. *The probabilistic zone graph of the $\mathbb{P}$TA in Figure 1a is the MDP given in Figure 2a. The symbolic states $\mathbf{s}_i = (l_i, C_i)$ are expanded in Table I.*

C. Reconstructing a $\mathbb{P}$TA from a Probabilistic Zone Graph

It is well-known that, given a timed automata $\mathcal{A}$ and its zone graph, a second timed automaton $\mathcal{A}'$ can be reconstructed from the zone graph, with the same structure as the zone graph, and such that the zone graph of $\mathcal{A}'$ is the same as that of $\mathcal{A}$. We extend this technique here to $\mathbb{P}$TAs.

Let $\mathcal{P}$ be a $\mathbb{P}$TA and let $\mathcal{M}$ be its probabilistic zone graph. Let us build a second $\mathbb{P}$TA $\mathcal{P}'$. Each state of $\mathcal{M}$ is translated into a location of $\mathcal{P}'$. Then, for each transition $(\mathbf{s}, e, \Upsilon') \in T$ in $\mathcal{M}$, where $e = (l, g, a, \Upsilon)$, we create a transition (l, g, a, Υ), where Υ is defined exactly as in $\mathcal{P}$, except that the target location matches the target state in $\mathcal{M}$ (a single location in $\mathcal{P}$ can yield different states in $\mathcal{M}$). Following results for timed automata, the probabilistic zone graph of $\mathcal{P}'$ is $\mathcal{M}$.

Example 7. *The $\mathbb{P}$TA reconstructed from the probabilistic zone graph in Figure 2a is given in Figure 3. Its probabilistic zone graph is again that of Figure 2a.*

D. An Algorithm for the Consistency of $\mathbb{IP}$TAs

We start with the following observation: by construction, the purpose of the symbolic semantics of $\mathbb{IP}$TAs is to represent, at a lower level of abstraction, the same set of objects. Intuitively, the symbolic IMDP semantics of a given $\mathbb{IP}$TA should therefore be consistent iff the original $\mathbb{IP}$TA is itself consistent. This result is formally proven in the following theorem.

Theorem 1. *An $\mathbb{IP}$TA $\mathcal{IP}$ is consistent iff its probabilistic zone graph is consistent.*

Proof:

- $\Rightarrow$ Assume $\mathcal{IP}$ is consistent, and let us show that its probabilistic zone graph is consistent. From the definition of consistency, there exists a $\mathbb{P}$TA $\mathcal{P}$ such that $\mathcal{P} \models \mathcal{IP}$. Let $\mathcal{IM}$ (resp. $\mathcal{M}$) be the probabilistic zone graph of $\mathcal{IP}$ (resp. $\mathcal{P}$). From Definition 5, $\mathcal{P}$ simulates in part the transition relation of $\mathcal{IP}$ while matching its probability intervals. As a consequence, $\mathcal{M}$ will also simulate in part the transition relation of $\mathcal{IM}$ while matching its probability intervals. Hence, from Definition 7, we have $\mathcal{M} \models \mathcal{IM}$.
- $\Leftarrow$ Assume the probabilistic zone graph of $\mathcal{IP}$ is consistent, and let us show that $\mathcal{IP}$ is consistent. Let $\mathcal{IM}$ be the probabilistic zone graph of $\mathcal{IP}$. Since $\mathcal{IM}$ is consistent, from Lemma 1, there exists an implementation of $\mathcal{IM}$ with the same structure. Let $\mathcal{M}$ be that implementation of same structure. Let $\mathcal{P}$ be the $\mathbb{P}$TA reconstructed from the probabilistic zone graph $\mathcal{M}$, following the construction in Section III-C. Observe that, since $\mathcal{M}$ and $\mathcal{IM}$ have the same structure, the probabilistic zone graphs of $\mathcal{P}$ and $\mathcal{IP}$ are equal (except for the value of the probabilities). Now, since $\mathcal{M} \models \mathcal{IM}$, then we also have $\mathcal{P} \models \mathcal{IP}$.

■

Given the results presented in Lemma 1 and Theorem 1, deciding whether a given $\mathbb{IP}$TA $\mathcal{IP}$ is consistent can be done by deciding whether its probabilistic zone graph admits at least one implementation that preserves its structure.

Such an algorithm was provided in [Del15] in the context of IMCs instead of IMDPs. We show how this algorithm can be adapted to our context. As for IMCs, we say that a state is *locally inconsistent* in a given IMDP iff one of its outgoing probabilistic (interval) transitions cannot be implemented, i. e., if there is no distribution that matches the specified intervals. Let $\mathcal{IM} = (S, s_0, \Sigma, T)$ be the IMDP symbolic semantics of a given $\mathbb{IP}$TA. The algorithm proceeds as follows:

Algorithm 1: Consistency of IMDPs

Let Inc be the set of locally inconsistent states in $\mathcal{IM}$ and Passed $= \emptyset$.
while $s_0 \notin$ Passed *and* Inc $\neq \emptyset$ **do**
    Let $s \in$ Inc and Passed $=$ Passed $\cup \{s\}$.
    Replace all transitions (s', a, I) such that $I(s) \neq [0, 0]$ with (s', a, I') where
    - $I'(s'') = I(s'')$ for all $s'' \neq s$,
    - $I'(s) = [0, 0]$ if $0 \in I(s)$, and
    - $I'(s) = \emptyset$ otherwise.

    Update Inc $\subseteq (S \setminus$ Passed$)$.

The algorithm is based on the following principle: as soon as a locally inconsistent state is detected, it is either made

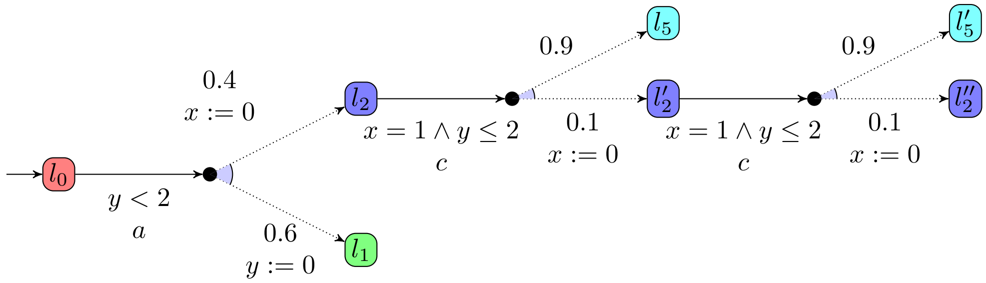


Figure 3: A $\mathbb{P}$TA reconstructed from the probabilistic zone graph in Figure 2a

unreachable by forcing incoming interval probabilities to $[0, 0]$ whenever this is possible (which might create new local inconsistencies in predecessor states) or by enforcing predecessor states to be inconsistent by modifying the interval probabilities to $\emptyset$ when 0 is not an admissible transition probability.

In the context of IMCs, it is proven in [Del15] that this algorithm converges and that the original IMC is consistent iff the initial state is not locally inconsistent in the resulting IMC. The proof from [Del15] can be trivially adapted to the context of IMDPs.

IV. Consistency-emptiness and Synthesis for PI$\mathbb{P}$TAs

We now move to the parametric setting and consider the following two problems:

Consistency-emptiness problem: Given a PI$\mathbb{P}$TA $\mathcal{PIP}$, does there exist a parameter valuation v such that $v(\mathcal{PIP})$ is consistent?

Consistency-synthesis problem: Given a PI$\mathbb{P}$TA $\mathcal{PIP}$, find all parameter valuations v for which $v(\mathcal{PIP})$ is consistent.

In the following, we first address the consistency-emptiness problem and show that this problem is undecidable in the general context of PI$\mathbb{P}$TAs. We then propose a semi-algorithm for the consistency-synthesis problem based on an adaptation of the parametric zone-graph construction for parametric timed automata and the decision algorithm for I$\mathbb{P}$TAs presented in Section III. This semi-algorithm only terminates when the parametric probabilistic zone-graph construction of the original PI$\mathbb{P}$TA is finite. When this is the case, the set of parameter values that are synthesized is exactly those that ensure consistency of the resulting I$\mathbb{P}$TA.

A. Undecidability of the Emptiness Problem

The undecidability of the consistency-emptiness for PI$\mathbb{P}$TAs follows from the undecidability of the reachability emptiness for parametric timed automata.

Theorem 2. *The consistency-emptiness for PI$\mathbb{P}$TAs is undecidable.*

Proof: The reachability emptiness for parametric timed automata (i. e., the existence of at least one parameter valuation for which a given location is reachable) is undecidable. This result comes with various "flavors" in the literature (numbers of clocks or parameters, dense or discrete time, strict or non-strict inequalities in guards, use or not of invariants, etc. – see [And16] for a survey), but all use a reduction from the halting problem of a 2-counter machine, which is undecidable. All these reductions define a matching between a state of the machine and a location of the parametric timed automaton (PTA). Clearly, one can use any of these encodings of a 2-counter machine to conclude that the consistency-emptiness for PI$\mathbb{P}$TAs is undecidable. Let us reuse the proof of undecidability given in [BBLS15] for two reasons. First, it is the best known proof over discrete time, and one best proof over dense time, in terms of number of clocks used in the reduction (three, all compared to parameters). Second, it uses no invariant, which is inline with our setting.

Let us reuse the PTA encoding a 2-counter machine proposed in [BBLS15]. (The reader can refer to [BBLS15] for details.) We modify that encoding as follows. In the PTA location encoding the unique halting state of the 2-counter machine, we add a transition to a new location for which no implementation exists (for example a single transition labeled with $[0.5, 0.5]$). Hence the halting location is reachable iff the underlying I$\mathbb{P}$TA admits no implementation. Hence the 2-counter machine halts iff there exists no parameter valuation for which there exists an implementation. ■

The undecidability of the consistency-emptiness problem rules out the possibility to, in general, compute a solution to the consistency-synthesis problem. In the following, we will still address this computation problem by proposing an algorithm based on the parametric probabilistic zone graph; if this latter graph is finite, then our algorithm is exact.

B. A Symbolic Semantics for PI$\mathbb{P}$TAs

We equip PI$\mathbb{P}$TAs with a symbolic semantics, defined below. Basically, it is inline with the symbolic semantics defined for parametric timed automata (see e. g., [ACEF09], [JLR15]), with the addition of probabilistic intervals on the

State	Location	C	$C{\downarrow_\Gamma}$
$\mathbf{s}_0$	l_0	$x = y \wedge x \geq 0 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_1$	l_1	$0 \leq x - y < 2 \wedge y \geq 0 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_2$	l_2	$0 \leq y - x < 2 \wedge x \geq 0 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_3$	l_3	$2 \leq x - y \leq \gamma \wedge y \geq 0$	$\gamma \geq 2$
$\mathbf{s}_4$	l_4	$x = y \wedge x \geq 0 \wedge \gamma \geq 2$	$\gamma \geq 2$
$\mathbf{s}_5$	l_5	$0 \leq y - x \leq 1 \wedge x \geq 1 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_6$	l_2	$1 \leq y - x \leq 2 \wedge x \geq 0 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_7$	l_5	$y \geq 2 \wedge y = x + 1 \wedge \gamma \geq 0$	$\gamma \geq 0$
$\mathbf{s}_8$	l_2	$y \geq 2 \wedge y = x + 2 \wedge \gamma \geq 0$	$\gamma \geq 0$

Table II: Description of the states in Figure 2b

edges; as a consequence, the semantics becomes not an LTS, but an IMDP.

Definition 9 (Symbolic semantics of a PI$\mathbb{P}$TA)**.** *Given a PI$\mathbb{P}$TA $\mathcal{PIP} = (\Sigma, L, l_0, X, \Gamma, \mathbb{I})$, the symbolic semantics of $\mathcal{PIP}$ is given by the IMDP $(\mathbf{S}, \mathbf{s}_0, \Sigma, T)$, with*

- $\mathbf{S} = \{(l, C) \in L \times \mathcal{Z}\}$, $\mathbf{s}_0 = (l_0, (\bigwedge_{1 \leq i \leq H} x_i = 0)^{\nearrow})$,
- $(\mathbf{s}, a, \Upsilon') \in T$ *if there exists* $(l, g, a, \Upsilon) \in \mathbb{I}$ *such that for all* $l' \in L$*, for all* $\rho \subseteq X$ *such that* $\Upsilon(l', \rho) > 0$, $C' = \big([C \wedge g]_\rho\big)^{\nearrow}$*, and* $\Upsilon'((l', C')) = \Upsilon(l', \rho)$*.*

Observe that, whenever a PI$\mathbb{P}$TA has no probabilistic choice, then the IMDP becomes a labeled transition system, and the symbolic semantics matches that of parametric timed automata. We refer to the symbolic semantics of $\mathcal{PIP}$ as the *parametric probabilistic zone graph* of $\mathcal{PIP}$.

Just as in parametric timed automata, the number of symbolic states in a PI$\mathbb{P}$TA can be infinite in general.

In parametric timed automata, the *reachability condition* is the projection onto the parameters of a parametric zone [JLR15]. It is well-known that, given a symbolic run of a parametric timed automaton leading to a symbolic state (l, C), there exists an equivalent concrete run iff $\gamma \models C{\downarrow_\Gamma}$ [HRSV02]. Since our definition of zones matches that of [HRSV02], this results extends to PI$\mathbb{P}$TAs in a straightforward manner.

Lemma 2. *Let $\mathcal{PIP}$ be a PI$\mathbb{P}$TA. Consider a run in the parametric probabilistic zone graph of $\mathcal{PIP}$ reaching state (l, C). Let v be a parameter valuation. Then, there exists an equivalent run in $v(\mathcal{PIP})$ iff $v \models C{\downarrow_\Gamma}$.*

By equivalent run, we mean (just as for parametric timed automata) an identical discrete structure (locations and edges).

Example 8. *The parametric probabilistic zone graph of the PI$\mathbb{P}$TA in Figure 1b is the IMDP given in Figure 2b. The symbolic states $\mathbf{s}_i = (l_i, C_i)$ are expanded in Table II. In addition, we also give the reachability condition of each state, i. e., the projection onto the parameters of the zone ($C{\downarrow_\Gamma}$).*

C. A Semi-Algorithm for Consistency-Synthesis for PI$\mathbb{P}$TAs

Unlike for I$\mathbb{P}$TAs / IMDPs where inconsistent states can only be avoided by enforcing their incoming probabilities to 0, there are two ways of avoiding inconsistent states in PI$\mathbb{P}$TAs. Indeed, while imposing a 0 probability to all transitions going to inconsistent states is a safe choice, it is also possible to avoid inconsistent states by cleverly choosing parameter values such that the guards of transitions potentially going to these states are never satisfied.

The algorithm we propose for synthesizing parameter valuations ensuring consistency of a given PI$\mathbb{P}$TA is based on the following observation: Since parameters only occur in transition guards, the choice of parameter values cannot interfere with the choice of probability distributions matching (or not) the specified intervals. That comes from the fact that, given a state $\mathbf{s}$, all successors of this state via a given transition have the same parameter constraint (this would not hold with invariants). As a consequence, states that can be made unreachable through probabilistic choice can be made so regardless of the choice of parameter values.

Algorithm 2 is therefore constituted of two main parts. The first part (while loop – lines 5–14) is similar to Algorithm 1 presented earlier. The main difference is that the loop from Algorithm 2 does not entirely remove inconsistent states. Instead of systematically making locally inconsistent states unreachable whether this is allowed or not according to the specified intervals, this version marks inconsistent states (with marking function λ) but only makes them unreachable when this is allowed, i. e., when replacing incoming transition probability intervals with $[0, 0]$ does not make predecessor states inconsistent. If this is not the case, then the incoming transitions are left untouched but the predecessor states are marked as inconsistent with λ. If they can be made unreachable without creating new inconsistencies, then they will be made so in a later pass. Otherwise, locally inconsistent states will be "removed" using parameter valuations in the second loop.

Once the first loop is processed, the only locally inconsistent states that remain are those that cannot be avoided using probabilities.

The second part (lines 18–22) consists in removing parameter valuations that allow reaching locally inconsistent states in the resulting IMDP. In fact, instead of removing the inconsistent states, we remove their successors responsible for making a state inconsistent (lines 19–21). Also note that, due to the absence of invariants, all successors of a state through a given probability distribution have the same parameter constraint; it is hence sufficient to pick any of them. Recall from Lemma 2 that the parametric zone $C{\downarrow_\Gamma}$ attached to a given symbolic state $\mathbf{s} = (l, C)$ in the IMDP semantics of a given PI$\mathbb{P}$TA $\mathcal{PIP}$ exactly represents the parameter valuations for which the state $\mathbf{s}$ is reachable in the resulting semantics. As a consequence, $\mathbf{s}$ will be reachable in the IMDP semantics of the I$\mathbb{P}$TA resulting from a given parameter valuation iff this parameter valuation is in $C{\downarrow_\Gamma}$. Remark that the order in which locally inconsistent states are processed is not important. In fact, they can be all processed at once by removing all associated parameter zones.

Algorithm 2: Consistency of PI$\mathbb{P}$TAs

Input: IMDP $\mathcal{IM}$ (semantics of a PI$\mathbb{P}$TA $\mathcal{PIP}$)
Output: Constraint K guaranteeing consistency

1 Let Inc be the set of locally inconsistent states in $\mathcal{IM}$ and $\mathsf{Passed} = \emptyset$.
2 $\lambda((l, C)) = \infty$ for all $(l, C) \in \mathbf{S} \setminus \mathsf{Inc}$.
3 $\lambda((l, C)) = 0$ for all $(l, C) \in \mathsf{Inc}$.
4 $n = 0$
5 **while** $\mathsf{Inc} \neq \emptyset$ **do**
6   Pick $(l, C) \in \mathsf{Inc}$ s.t. $\lambda((l, C)) \neq \infty$ is minimal
7     $\mathsf{Passed} = \mathsf{Passed} \cup \{(l, C)\}$.
8     $\mathsf{Inc} = \mathsf{Inc} \setminus \{(l, C)\}$
9   **for** *all transitions* $(\mathbf{s}, a, I)$ *such that* $\mathbf{s} \notin \mathsf{Passed}$ *and* $I((l, C)) \neq [0, 0]$ **do**
10   **if** $0 \in I((l, C))$ *and* $I[(l, C)|_{[0,0]}]$ *is consistent* **then**
11             $I((l, C)) \leftarrow [0, 0]$
12   **else**
13             $\lambda(\mathbf{s}) = \min(\lambda(\mathbf{s}), \lambda((l, C)) + 1)$
14             $\mathsf{Inc} \leftarrow \mathsf{Inc} \cup \{\mathbf{s}\}$
15 **if** $\lambda(\mathbf{s}_0) = \infty$ **then**
16   **return** $\top$
17 Remove all unreachable states from $\mathcal{IM}$
18 $K \leftarrow \top$
19 **for** *all locally inconsistent transitions* (s, a, I) **do**
20   Pick a state (l, C) such that $I((l, C)) \neq [0, 0]$
21     $K \leftarrow K \setminus C{\downarrow_\Gamma}$
22 Remove in $\mathcal{IM}$ and Inc all states (l', C') such that $C'{\downarrow_\Gamma} \cap K = \bot$ (as well as transitions from and to these states)
23 **if** $\mathbf{s}_0$ *has been removed* **then**
24   **return** $\bot$
25 **else**
26   **return** K

Proposition 1 (Termination)**.** *Let $\mathcal{PIP}$ be a PI$\mathbb{P}$TA, and let $\mathcal{IM}$ be its parametric probabilistic zone graph. Assume $\mathcal{IM}$ is finite. Then the application of Algorithm 2 to $\mathcal{IM}$ terminates.*

Proof: The first loop iterates on inconsistent states; at each iteration, one state is removed from Inc, and one or more states are added to Inc. In addition, exactly one state is added to Passed; since a state in Passed can never be added again to Inc, the first loop terminates.

The second loop iterates exactly once on each locally inconsistent transition, of which there is a finite number. ∎

Proposition 2 (Correctness)**.** *Let $\mathcal{PIP}$ be a PI$\mathbb{P}$TA, and let $\mathcal{IM}$ be its parametric probabilistic zone graph. Assume $\mathcal{IM}$ is finite. Let K be the result of the application of Algorithm 2 to $\mathcal{IM}$. Let $v \models K$.*

Then $v(\mathcal{PIP})$ is consistent.

Proof (sketch): From Lemma 2 and the fact that any inconsistent state has been removed, and therefore valuations leading to inconsistent states are absent from K. ∎

Proposition 3 (Completeness)**.** *Let $\mathcal{PIP}$ be a PI$\mathbb{P}$TA, and let $\mathcal{IM}$ be its parametric probabilistic zone graph. Assume $\mathcal{IM}$ is finite. Let v be such that $v(\mathcal{PIP})$ is consistent. Let K be the result of the application of Algorithm 2 to $\mathcal{IM}$.*

Then $v \models K$.

Proof (sketch): From Lemma 2 and the fact that only parameter valuations leading to inconsistent states (and for which no implementation of interval distribution can be set) are removed. ∎

Remark 1. *Algorithm 2 is an algorithm: it always terminates, and its result is sound and complete. However, it takes as input the parametric probabilistic zone graph of the PI$\mathbb{P}$TA, the computation of which may not terminate in general. Hence, our entire procedure (computation of the parametric probabilistic zone graph, and then application of Algorithm 2) can be seen as a semi-algorithm: it may not terminate but, if it terminates, then its result is correct.*

Example 9. *Let us apply Algorithm 2 to the PI$\mathbb{P}$TA $\mathcal{PIP}$ given in Figure 1b. Recall that the parametric probabilistic zone graph of $\mathcal{PIP}$ is given in Figure 2b, with the description of the states given in Table II. Initially, $\mathsf{Inc} = \{\mathbf{s}_1\}$ and $\lambda(\mathbf{s}_1) = 0$ (and ∞ for other states). In the first while loop, setting to 0 the probability on the transition from $\mathbf{s}_0$ to $\mathbf{s}_1$ fails, because this does not satisfy the test "$I[(l, C)|_{[0,0]}]$ is consistent" (line 10); indeed, the second probability leaving $\mathbf{s}_0$ via action e_1 can only be at most 0.5. Hence, $\mathbf{s}_0$ becomes marked, and $\lambda(\mathbf{s}_0) = 1$.*

Then, since the initial state is marked, we cannot conclude yet, and we enter the second phase. We have a single locally inconsistent transition, i. e., the one originating from $\mathbf{s}_1$. We pick arbitrarily $\mathbf{s}_3$ (picking $\mathbf{s}_4$ is identical), project its constraint onto Γ, which yields $\gamma \geq 2$ according to Table II, and perform the difference between K and $\gamma \geq 2$. This yields $K : \gamma < 2$. We then remove states for which the parameter constraint is disjoint from K, i. e., $\mathbf{s}_3, \mathbf{s}_4$. Since $\mathbf{s}_0$ was not removed, we return $K : \gamma < 2$. Hence, for any parameter valuation v such that $\gamma < 2$, $v(\mathcal{PIP})$ is consistent.

V. Conclusion

In this work, we provided abstractions to reason on systems involving real-time constraints and probabilities: first, by allowing probabilities to range in some intervals, and, second, by allowing timing constants to be abstracted in the form of parameters. Without parameters, we proposed an approach to decide whether an interval probabilistic timed automaton is consistent, i. e., admits an implementation based on a simulation relation. When adding parameters, the mere existence of a parameter valuation yielding consistency is undecidable. We proposed however a semi-algorithm to synthesize valuations ensuring consistency.

Future works include the exhibition of subclasses of PI$\mathbb{P}$TAs for which exact synthesis can be achieved. In addition, we are interested in considering higher-level abstractions of probabilities, e. g., in the form of parameters instead of intervals with constant bounds.

References

[ACEF09] Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, 2009.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[AFS13] Étienne André, Laurent Fribourg, and Jeremy Sproston. An extension of the inverse method to probabilistic timed automata. *Formal Methods in System Design*, (2):119–145, 2013.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC'93, pages 592–601, New York, NY, USA, 1993. ACM.

[And16] Étienne André. What's decidable about parametric timed automata? In Cyrille Artho and Peter Csaba Ölveczky, editors, *Proceedings of the 4th International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS'15)*, volume 596 of *Communications in Computer and Information Science*, pages 1–17. Springer, 2016.

[BBLP06] Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *International Journal on Software Tools for Technology Transfer*, 8(3):204–215, 2006.

[BBLS15] Nikola Beneš, Peter Bezděk, Kim G. Larsen, and Jiří Srba. Language emptiness of continuous-time parametric timed automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer, July 2015.

[BY03] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2003.

[Del15] Benoît Delahaye. Consistency for parametric interval markov chains. In *2nd International Workshop on Synthesis of Complex Parameters, SynCoP 2015, April 11, 2015, London, United Kingdom*, volume 44 of *OASICS*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[DLL+12] Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. Consistency and refinement for interval markov chains. *J. Log. Algebr. Program.*, 81(3):209–226, 2012.

[DLP16] Benoît Delahaye, Didier Lime, and Laure Petrucci. Parameter synthesis for parametric interval markov chains. In *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings*, volume 9583 of *Lecture Notes in Computer Science*, pages 372–390. Springer, 2016.

[GJ95] H. Gregersen and H. E. Jensen. Formal design of reliable real time systems. Master's thesis, Department of Mathematics and Computer Science, Aalborg University, 1995.

[HRSV02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.

[JK14] Aleksandra Jovanović and Marta Z. Kwiatkowska. Parameter synthesis for probabilistic timed automata using stochastic game abstractions. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Proceedings of the 8th International Workshop on Reachability Problems (RP 2014)*, volume 8762 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2014.

[JL91] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277. IEEE Computer, 1991.

[JLR15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Integer parameter synthesis for timed automata. *IEEE Transactions on Software Engineering*, 41(5):445–461, 2015.

[KNPS06] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

[KNSS02] Marta Z. Kwiatkowska, Gethin Norman, R. Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282:101–150, 2002.