Hyper Parametric Timed CTL

Masaki Waga and Étienne André

***Abstract*—Hyperproperties enable simultaneous reasoning about multiple execution traces of a system and are useful to reason about noninterference, opacity, robustness, fairness, observational determinism, etc. We introduce hyper parametric timed computation tree logic (HyperPTCTL), extending hyperlogics with timing reasoning and, notably, parameters to express unknown values. We mainly consider its nest-free fragment, where the temporal operators cannot be nested. However, we allow extensions that enable counting actions and comparing the duration since the most recent occurrence of specific actions. We show that our nest-free fragment with this extension is sufficiently expressive to encode the properties, e.g., opacity, (un)fairness, or robust observational (non)determinism. We propose semi-algorithms for the model checking and synthesis of parametric timed automata (TAs) (an extension of TAs with timing parameters) against this nest-free fragment with the extension via reduction to the PTCTL model checking and synthesis. While the general model checking (and thus synthesis) problem is undecidable, we show that a large part of our extended (yet nest-free) fragment is decidable, provided the parameters only appear in the property, not in the model. We also exhibit additional decidable fragments where the parameters within the model are allowed. We implemented our semi-algorithms on the top of the IMITATOR model checker and performed experiments. Our implementation supports most of the nest-free fragments (beyond the decidable classes). The experimental results highlight our method's practical relevance.**

***Index Terms*—Hyperproperties, model checking, parameter synthesis, parametric timed automata, temporal logic.**

I. Introduction

PARAMETRIC timed automata (PTAs) [1] is an extension of finite-state automata for modeling and verification of the real-time systems, where the timing constraints are not fixed but parameterized. PTAs extend the concept of timed automata (TAs) [2] by introducing the parameters into time bounds, allowing for analysing a system across a range of timing scenarios.

Hyperproperties enable reasoning simultaneously about multiple execution traces of a system and turn useful to reason about noninterference, opacity, fairness, robustness, observational determinism, etc. We introduce hyper parametric timed computation tree logic (HyperPTCTL), extending hyperlogics with not only timing reasoning but also the timing parameters able to express the unknown values. HyperPTCTL can be used typically to reason about multiple traces on PTAs.

After defining the syntax and semantics of general HyperPTCTL, we mainly consider the nest-free fragment, where the temporal operators cannot be nested. However, we extend HyperPTCTL with additional predicates that enable counting actions and comparing the duration since the most recent occurrence of specific actions using the diagonal constraints of the form $\mathrm{LAST}(\sigma_{\pi_1}) - \mathrm{LAST}(\sigma_{\pi_2})$, where σ is a proposition and π_1 π_2 represent two paths. Even without the nesting of temporal operators, we demonstrate that this extension enables encoding the classical properties, such as opacity, (un)fairness, or observational (non)determinism–in a timed and parametric setting. For example, we can use a simple HyperPTCTL formula to encode a *robust observational nondeterminism*: "By giving the same sequence of inputs at the same timing to the system, it is possible to get the same sequence of the outputs but with large time difference." A timing parameter in the formula is used to leave the time difference unspecified, and for example, the feasible values can be synthesized (by our semi-algorithm). We denote this nest-free but extended fragment by Nest-Free Ext-HyperPTCTL.

We consider two problems over parametric formulas and/or models as follows.

1) The *model checking* problem asks whether there exists a valuation for which the model satisfies the formula.
2) The *synthesis* problem asks for the exact valuations set for which the model satisfies the formula. Ideally, this representation should be given symbolically, e.g., in a decidable logical formalism.

We show that Nest-Free Ext-HyperPTCTL model checking (resp, synthesis) of PTAs is reducible to the PTCTL model checking (resp, synthesis) of PTAs. Fig. 1 outlines our reduction. We show a more concrete working example later in Section V-C. First, we reduce Nest-Free Ext-HyperPTCTL model checking (resp, synthesis) to the

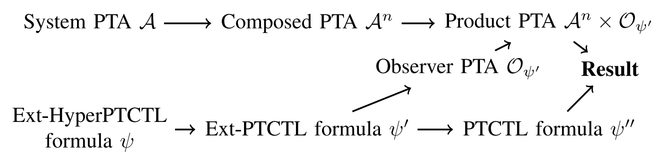

Fig. 1. Our reduction: Nest-Free Ext-HyperPTCTL synthesis (resp, model checking) is reduced to Ext-PTCTL synthesis (resp, model checking) via self-composition; the extended predicates in Ext-PTCTL are evaluated by an observer PTA, which is composed with the PTA $\mathcal{A}^n$.

Manuscript received 9 August 2024; accepted 9 August 2024. This work was supported in part by the Japan Science and Technology (JST) PRESTO under Grant JPMJPR22CA; in part by the Japan Science and Technology (JST) CREST under Grant JPMJCR2012; in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 22K17873; in part by the ANR Provable Mitigation of Side Channel through Parametric Verification under Grant ANR-19-CE25-0015; and in part by the ANR Better Synthesis for Underspecified Quantitative Systems (BisoUS) under Grant ANR-22-CE48-0012. This article was presented at the International Conference on Embedded Software (EMSOFT) 2024 and appeared as part of the ESWEEK-TCAD special issue. This article was recommended by Associate Editor S. Dailey. *(Corresponding author: Masaki Waga.)*

Masaki Waga is with the Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (e-mail: mwaga@fos.kuis.kyoto-u.ac.jp).

Étienne André is with Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, 93430 Villetaneuse, France.

Digital Object Identifier 10.1109/TCAD.2024.3443704

Nest-Free Ext-PTCTL model checking (resp, synthesis) by taking the self-composition $\mathcal{A}^n$ of the system PTA $\mathcal{A}$, where n is the number of quantified path variables (i.e., the number of simultaneously reasoned execution traces) in the Ext-HyperPTCTL formula ψ. Then, we construct an *observer* PTA $\mathcal{O}_{\psi'}$ [3] to evaluate the extended predicates in the given Nest-Free Ext-PTCTL formula ψ'. We show that the result of PTCTL model checking (resp, synthesis) for the product PTA $\mathcal{A}^n \times \mathcal{O}_{\psi'}$ is the same as the result of the original problem. Thus, the original problem is reduced to the PTCTL model checking (resp, synthesis). By integrating this reduction with a semi-algorithm for the PTCTL model checking (resp, synthesis), we derive a semi-algorithm for the Nest-Free Ext-HyperPTCTL model checking (resp, synthesis).

While the Nest-Free Ext-HyperPTCTL model checking of PTAs is trivially undecidable due to the undecidability of reachability-emptiness of PTAs [1], we show that they are decidable for a large part of Nest-Free Ext-HyperPTCTL, provided the parameters only appear in the property, not in the model. We also exhibit additional decidable fragments where the parameters in the model are allowed.

We implemented our approach on the top of the existing IMITATOR parametric timed model checker [4] and performed experiments. Our implementation HyPTCTLchecker supports most of the nest-free fragment (beyond the decidable classes too, in which case at the risk of nontermination or approximated result). The experimental results show that our approach can handle various properties if the PTA has a moderate size.

Our contributions are summarized as follows.

1) We introduce HyperPTCTL and its extension Ext-HyperPTCTL to count the actions and to measure the time since their final occurrence (Section IV).
2) We propose semi-algorithms for the Nest-Free Ext-HyperPTCTL model checking (resp, synthesis) of PTAs (Section V).
3) While the Nest-Free Ext-HyperPTCTL model checking and synthesis are trivially undecidable, we exhibit several decidable subclasses, with the parameters either in the PTA or in the Nest-Free Ext-HyperPTCTL formula (Section VI).
4) We implemented our approach and performed the experiments. The experimental results suggest the practical relevance of our approach (Section VII).

To the best of our knowledge, our work is not only the first one extending TCTL into hyperlogics but also the first one to allow for timing parameters in such a TCTL hyper-extension.

II. Related Work

A. Model Checking Parametric Timed Formalisms

First, model checking PTAs against the nonparametric nest-free fragment (without nested operators) of PTCTL is already undecidable, as reachability-emptiness (also called $\exists\Diamond$-emptiness, i.e., the emptiness over the valuations set for which a given location can be reached) is undecidable for general PTAs over dense or discrete time [1]. Unavoidability($\forall\Diamond$)-emptiness is undecidable too [5].

Reachability-emptiness over discrete time for PTAs with two parametric clocks,[1] arbitrarily many nonparametric clocks and one parameter is EXPSPACE-complete [6].

In [7], model checking nonparametric TAs against parametric TCTL (with integer-valued parameters) is considered over both the discrete and dense time. $\tilde{\exists}$PTCTL is defined as the existential fragment (over parameters) of PTCTL.[2] Both the discrete time [7, Corollary 7.3] and dense time [7, Th. 7.5] model checking the problems are in 5EXPTIME in the product of the model and the formula, and are in 3EXPTIME for the $\tilde{\exists}$PTCTL fragment [7, Propositions 7.4 and 7.6].

Model-checking subclasses of PTAs against TCTL (beyond reachability) is notably considered in [8]: on the one hand, even for the severely restricted class of U-PTAs (a subclass of PTAs in which the parameters can only be compared to a clock as an upper bound [9]), and even without invariants, the emptiness is undecidable for the nested TCTL. On the other hand, it is then shown in [8] that the nest-free TCTL is decidable for L/U-PTAs (a subclass of PTAs in which the parameters are partitioned between the lower-bound and upper-bound parameters [10]) without the invariants.

B. Hyperproperties

Hyperproperties drew the recent attention, and various hyperlogics have been introduced by extending the conventional temporal logics (e.g., [11], [12], [13], and [14]).

One of the closest works to our timed hyperlogics (without the parameters) is HyperMITL [12], a timed extension of HyperLTL [11]. In general, the model checking problem is undecidable, even with very restricting timing constraints; it becomes decidable under certain conditions, notably absence of alternation. For decidable subcases, they use a construction based on the *self-composition*, which we also use in Section V-A. However, their construction is primarily for the untimed models, while our reduction is for PTAs.

Another closely related work is HyperMTL [14], another timed extension of HyperLTL. If the time domain is discrete, i.e., the timestamps are integers, HyperMTL model checking is decidable even with quantifier alternation. Although their algorithm covers many interesting properties, it is limited to the discrete-time and nonparametric settings.

Both the amplitude and timing parameters are considered in [13] for HyperSTL, but the goal is requirement mining from the traces rather than the model checking. Quantifier alternation is allowed.

III. Preliminaries

For a set X, we denote its powerset by $\mathcal{P}(X)$. For sets X and Y, we denote a partial function f from X to Y by $f\colon X \nrightarrow Y$ and denote its domain by $\mathbf{dom}(f) \subseteq X$.

We let $\mathbb{T}$ be the domain of the time, which will be either non-negative reals $\mathbb{R}_{\geq 0}$ or naturals $\mathbb{N}$. Let $\mathbb{C} = \{c_1, \dots, c_H\}$ be a set of *clocks*, i.e., variables that evolve at the same rate. A

[1] A parametric clock is a clock compared to a parameter in at least one guard or invariant.

[2] Of the form $\tilde{\exists} p_1, \dots, p_n : \varphi$ with φ without quantifiers over the parameters. Note that, in this article, we use $\tilde{\exists}$ to distinguish between the existential quantification over the parameters ($\tilde{\exists}$) and over paths ($\exists$).

clock valuation is a function $\nu : \mathbb{C} \to \mathbb{T}$. We write $\vec{0}_{\mathbb{C}}$ for the clock valuation assigning 0 to all the clocks. Given $d \in \mathbb{T}$, $\nu + d$ denotes the valuation s.t. $(\nu + d)(c) = \nu(c) + d$, for all $c \in \mathbb{C}$. Given $R \subseteq \mathbb{C}$, we define the *reset* of a valuation ν, denoted by $[\nu]_R$, as follows: $[\nu]_R(c) = 0$ if $c \in R$, and $[\nu]_R(c) = \nu(c)$ otherwise.

We assume a set $\mathbb{P} = \{p_1, \dots, p_M\}$ of *parameters*, i.e., unknown constants. A parameter *valuation* v is a function $v \colon \mathbb{P} \to \mathbb{Q}_{\geq 0}$.[3] We assume $\bowtie \in \{<, \leq, =, \geq, >\}$. A *(clock) guard* g is a constraint over $\mathbb{C} \cup \mathbb{P}$ defined by a conjunction of the inequalities of the form $c \bowtie \gamma$ with $\gamma \in \mathbb{P} \cup \mathbb{N}$. For simplicity, we often use intervals instead of a conjunction of the inequalities. Given g, we write $\nu \models v(g)$ if the expression obtained by replacing each c with $\nu(c)$ and each p with $v(p)$ in g evaluates to true. For a finite set $X = \{x_1, x_2, \dots, x_N\}$ of the size $N \in \mathbb{N}$, a *linear term lt* (resp, *non-negative linear term* $lt_{\geq 0}$) is of the form $\sum_{1 \leq i \leq N} \alpha_i x_i + d$, with $\alpha_i, d \in \mathbb{Z}$ (resp, $\alpha_i, d \in \mathbb{N}$).

PTAs [1] extend TAs [2] with the parameters within guards and invariants in the place of the integer constants.

Definition 1 (PTA): A PTA $\mathcal{A}$ is an eight-tuple $\mathcal{A} = (\Sigma, L, L_0, \mathbb{C}, \mathbb{P}, I, E, \Lambda)$, where as follows:

1) Σ is a finite set of atomic propositions;
2) L is a finite set of locations;
3) $L_0 \subseteq L$ is the set of initial locations,
4) $\mathbb{C}$ is a finite set of clocks;
5) $\mathbb{P}$ is a finite set of parameters;
6) I is the invariant, assigning to every $\ell \in L$ a clock guard $I(\ell)$;
7) E is a finite set of edges $e = (\ell, g, R, \ell')$, where $\ell, \ell' \in L$ are the source and target locations, $R \subseteq \mathbb{C}$ is a set of clocks to be reset, and g is the transition guard;
8) $\Lambda \colon L \to \mathcal{P}(\Sigma)$ is the labeling function assigning the atomic propositions satisfied at each location.

Given a parameter valuation v, we denote by $v(\mathcal{A})$ the nonparametric structure where all the occurrences of a parameter p_i have been replaced by $v(p_i)$. We refer as a *timed automaton (TA)* to any structure $v(\mathcal{A})$, by assuming a rescaling of the constants: by multiplying all the constants in $v(\mathcal{A})$ by the least common multiple of their denominators, we obtain an equivalent (integer-valued) TA as defined in [2].

Example 1: The PTA in Fig. 2 contains one clock c and one parameter p_1. The invariant of ℓ_0 is "$c \leq p_1$" and the transition to ℓ_1 is guarded by "$p_1 - 1 < c < p_1$," and resets c. Atomic propositions H and L are associated with ℓ_0 and ℓ_1, respectively. This PTA models a clock generator with drift: the digital signal switches between the high (H) and low (L) states in a near periodic manner but with some timing deviation, depending on the value of the parameter p_1.

Let us now recall the concrete semantics of TAs.

Definition 2 (Semantics of a TA): For a PTA $\mathcal{A} = (\Sigma, L, L_0, \mathbb{C}, \mathbb{P}, I, E, \Lambda)$ and a parameter valuation v, the semantics of the TA $v(\mathcal{A})$ is given by the timed transition system (TTS) $T_{\mathcal{A}} = (S, S_0, \to)$ with as follows.

1) $S = \{(\ell, \nu) \in L \times \mathbb{T}^H \mid \nu \models I(\ell)\}$.
2) $S_0 = \{(\ell_0, \vec{0}_{\mathbb{C}}) \mid \ell_0 \in L_0\}$.
3) $\to$ consists of the discrete and (continuous) delay transition relations.
   a) *Discrete transitions:* $(\ell, \nu) \overset{e}{\mapsto} (\ell', \nu')$, if $(\ell, \nu), (\ell', \nu') \in S$, and there exists $e = (\ell, g, R, \ell') \in E$, such that $\nu' = [\nu]_R$, and $\nu \models g$.
   b) *Delay transitions:* $(\ell, \nu) \overset{d}{\mapsto} (\ell, \nu + d)$, with $d \in \mathbb{T}$, if $\forall d' \in [0, d], (\ell, \nu + d') \in S$.

Moreover, we write $(\ell, \nu) \xrightarrow{(d,e)} (\ell', \nu')$ for a combination of the delay and discrete transitions if $\exists \nu'' : (\ell, \nu) \overset{d}{\mapsto} (\ell, \nu'') \overset{e}{\mapsto} (\ell', \nu')$. We let $\Lambda((\ell, \nu)) = \Lambda(\ell)$.

Given a TA $v(\mathcal{A})$ with concrete semantics $(S, S_0, \to)$, we refer to the states S as the *concrete states* of $v(\mathcal{A})$. For $s = (\ell, \nu) \in S$ and $d \in \mathbb{T}$, we let $s + d = (\ell, \nu + d)$. A *path* of $v(\mathcal{A})$ from a concrete state s is an alternating *infinite* sequence of concrete states of $v(\mathcal{A})$ and pairs of the edges and delays starting from s of the form $s_0 (= s), (d_0, e_0), s_1, \dots$ with $\sum_{i=0}^{\infty} d_i = +\infty$, for each $i = 0, 1, \dots$, $d_i \in \mathbb{T}$, $e_i \in E$, and $s_i \xrightarrow{(d_i, e_i)} s_{i+1}$. We denote the set of paths of $v(\mathcal{A})$ from s by $\mathrm{Paths}(v(\mathcal{A}), s)$. We let $\mathrm{Paths}(v(\mathcal{A})) = \bigcup_{s \in S} \mathrm{Paths}(v(\mathcal{A}), s)$. For a path $s_0, (d_0, e_0), s_1, \dots$ of $v(\mathcal{A})$, a *position* is a concrete state s satisfying $s = s_i + d$ for some $i \in \mathbb{N}$ and $d \leq d_i$. For a path ρ, we denote its initial position s_0 by $Init(\rho)$. For a position $s = s_i + d$ of a path $\rho = s_0, (d_0, e_0), s_1, \dots$, the *duration* $Dur_\rho(s)$ is $Dur_\rho(s) = d + \sum_{j=0}^{i-1} d_j$. If the path is clear from the context, we just write $Dur(s)$. For positions $s = s_i + d$ and $s' = s_j + d'$ of a path $s_0, (d_0, e_0), s_1, \dots$, we let $s < s'$, if we have $i < j$ or $Dur(s) < Dur(s')$. We let $s \leq s'$, if we have $s < s'$ or $s = s'$. For paths ρ, ρ', we write $\rho \succeq \rho'$, if ρ' is a suffix of ρ, i.e., for $\rho = s_0, (d_0, e_0), s_1, \dots$, ρ' is such that $s_i + d, (d_i - d, e_i), s_{i+1}, \dots$ for some $i \in \mathbb{N}$ and $d \in [0, d_i]$. We let $\mathcal{D}^{\rho}_{\rho'}$ be such i. We let $\rho \succ \rho'$ if we have $\rho \succeq \rho'$ and $\rho \neq \rho'$. For paths ρ, ρ' satisfying $\rho \succeq \rho'$, we let $Dur(\rho - \rho')$ be the duration of the initial position of ρ' in ρ.

For PTAs $\mathcal{A}^1$ and $\mathcal{A}^2$, we define both the parallel composition $\mathcal{A}^1 \mid\mid \mathcal{A}^2$ and synchronized product $\mathcal{A}^1 \times \mathcal{A}^2$. Intuitively, the parallel composition is to juxtapose two PTAs without synchronization, whereas the synchronized product is to compose two PTAs synchronizing the edges with the propositions. The parallel composition will be used when taking the self-composition of the systems to handle multiple paths simultaneously, while the synchronized product will be used when composing the systems with observers encoding the extended predicates.

For PTAs $\mathcal{A}^1 = (\Sigma^1, L^1, L^1_0, \mathbb{C}^1, \mathbb{P}^1, I^1, E^1, \Lambda^1)$ and $\mathcal{A}^2 = (\Sigma^2, L^2, L^2_0, \mathbb{C}^2, \mathbb{P}^2, I^2, E^2, \Lambda^2)$, their *parallel composition* is $\mathcal{A}^1 || \mathcal{A}^2 = (\Sigma^1 \sqcup \Sigma^2, L^1 \times L^2, L^1_0 \times L^2_0, \mathbb{C}^1 \sqcup \mathbb{C}^2, \mathbb{P}^1 \cup \mathbb{P}^2, I, E, \Lambda)$,

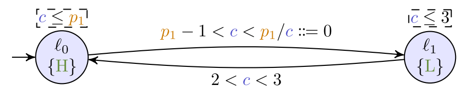

Fig. 2. Drifted clock generator example: PTA $\mathcal{A}$.

[3] We choose $\mathbb{Q}_{\geq 0}$ by consistency with most of the PTA literature, but also because, for the classical PTAs, choosing $\mathbb{R}_{\geq 0}$ leads to undecidability [15].

with $\sqcup$ denoting disjoint union, $I((\ell^1, \ell^2)) = I^1(\ell^1) \wedge I^2(\ell^2)$, $E = \{((\ell^1, \ell^2), g, R, (\ell^{1'}, \ell^2)) \mid (\ell^1, g, R, \ell^{1'}) \in E^1, \ell^2 \in L^2\} \cup \{((\ell^1, \ell^2), g, R, (\ell^2, \ell^{2'})) \mid (\ell^2, g, R, \ell^{2'}) \in E^2, \ell^1 \in L^1\} \cup \{((\ell^1, \ell^2), g^1 \wedge g^2, R^1 \cup R^2, (\ell^{1'}, \ell^{2'})) \mid (\ell^1, g^1, R^1, \ell^{1'}) \in E^1, (\ell^2, g^2, R^2, \ell^{2'}) \in E^2\}$, and $\Lambda((\ell^1, \ell^2)) = \Lambda^1(\ell^1) \sqcup \Lambda^2(\ell^2)$. For TAs $v_1(\mathcal{A}_1)$ and $v_2(\mathcal{A}_2)$ satisfying $v_1(p) = v_2(p)$ for any $p \in \mathbb{P}_1 \cap \mathbb{P}_2$, and paths ρ_1 and ρ_2 of $v_1(\mathcal{A}_1)$ and $v_2(\mathcal{A}_2)$, respectively, we let $\rho_1 \,||\, \rho_2$ be the path of $(v_1 \cup v_2)(\mathcal{A}_1 \,||\, \mathcal{A}_2)$ obtained by parallel composition of ρ_1 and ρ_2, where $v_1 \cup v_2$ is the parameter valuation, such that $(v_1 \cup v_2)(p) = v_1(p)$ if $p \in v_1$ and otherwise $(v_1 \cup v_2)(p) = v_2(p)$. Conversely, for a path ρ of $(v_1 \cup v_2)(\mathcal{A}_1 \,||\, \mathcal{A}_2)$ and $i \in \{1, 2\}$, we let $\rho|_i$ be the path of $v_i(\mathcal{A}_i)$ obtained by removing the locations, clock valuations, and edges from $\mathcal{A}_{3-i}$.

For PTAs $\mathcal{A}^1 = (\Sigma^1, L^1, L_0^1, \mathbb{C}^1, \mathbb{P}^1, I^1, E^1, \Lambda^1)$ and $\mathcal{A}^2 = (\Sigma^2, L^2, L_0^2, \mathbb{C}^2, \mathbb{P}^2, I^2, E^2, \Lambda^2)$, their *synchronized product* is $\mathcal{A}^1 \times \mathcal{A}^2 = (\Sigma^1 \cup \Sigma^2, L^1 \times L^2, L_0, \mathbb{C}^1 \sqcup \mathbb{C}^2, \mathbb{P}^1 \cup \mathbb{P}^2, I, E, \Lambda)$, with $L_0 = \{(\ell_0^1, \ell_0^2) \in L_0^1 \times L_0^2 \mid \Lambda^1(\ell_0^1) \cap \Sigma^2 = \Lambda^2(\ell_0^2) \cap \Sigma^1\}$, $I((\ell^1, \ell^2)) = I^1(\ell^1) \wedge I^2(\ell^2)$, $E = \{((\ell^1, \ell^2), g, R, (\ell^{1'}, \ell^2)) \mid (\ell^1, g, R, \ell^{1'}) \in E^1, \ell^2 \in L^2, \Lambda^1(\ell^{1'}) \cap \Sigma^2 = \Lambda^2(\ell^2) \cap \Sigma^1\} \cup \{((\ell^1, \ell^2), g, R, (\ell^1, \ell^{2'})) \mid (\ell^2, g, R, \ell^{2'}) \in E^2, \ell^1 \in L^1, \Lambda^1(\ell^1) \cap \Sigma^2 = \Lambda^2(\ell^{2'}) \cap \Sigma^1\} \cup \{((\ell^1, \ell^2), g^1 \wedge g^2, R^1 \sqcup R^2, (\ell^{1'}, \ell^{2'})) \mid (\ell^1, g^1, R^1, \ell^{1'}) \in E^1, (\ell^2, g^2, R^2, \ell^{2'}) \in E^2, \Lambda^1(\ell^{1'}) \cap \Sigma^2 = \Lambda^2(\ell^{2'}) \cap \Sigma^1\}$, and $\Lambda((\ell^1, \ell^2)) = \Lambda^1(\ell^1) \cup \Lambda^2(\ell^2)$. For a finite set $I = \{1, 2, \ldots, n\}$ of indices, we let $\times_{i \in I} \mathcal{A}^i = \mathcal{A}^1 \times \mathcal{A}^2 \times \cdots \times \mathcal{A}^n$, where each $\mathcal{A}^i$ is a PTA.

IV. Hyper Parametric Timed CTL

Here, we introduce HyperPTCTL. HyperPTCTL is a generalization of PTCTL [7] with quantifiers over paths to represent hyperproperties.

A. Syntax of (Ext-)HyperPTCTL

Definition 3 (Syntax of HyperPTCTL): For atomic propositions Σ and parameters $\mathbb{P}$, the syntax of HyperPTCTL formulas of the temporal level φ and the top level ψ are defined as follows, where $\sigma \in \Sigma$, $\mathcal{V}$ is the set of path variables, $\pi, \pi_1, \pi_2, \ldots, \pi_n \in \mathcal{V}$, $\gamma \in \mathbb{P} \cup \mathbb{N}$, $p \in \mathbb{P}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, and $lt_{\geq 0}$ is a non-negative linear term over $\mathbb{P}$

$$\begin{aligned} \varphi &::= \top \mid \sigma_\pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists \pi_1, \pi_2, \ldots, \pi_n.\, \varphi\, \mathcal{U}_{\bowtie\gamma}\, \varphi \\ &\quad\ \mid \forall \pi_1, \pi_2, \ldots, \pi_n.\, \varphi\, \mathcal{U}_{\bowtie\gamma}\, \varphi \\ \psi &::= \quad \varphi \mid p \bowtie lt_{\geq 0} \mid \neg\psi \mid \psi \vee \psi \mid \tilde{\exists} p\ \psi. \end{aligned}$$

As the syntax sugar, we utilize the following formulas:

$$\bot \equiv \neg\top \quad \varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2) \quad \varphi_1 \implies \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$$

$$\varphi_1 = \varphi_2 \equiv (\varphi_1 \wedge \varphi_2) \vee ((\neg\varphi_1) \wedge (\neg\varphi_2)) \quad \varphi_1 \neq \varphi_2 \equiv \neg(\varphi_1 = \varphi_2)$$

$$\exists \pi_1, \pi_2, \ldots, \pi_n.\, \varphi_1\, \mathcal{R}_{\bowtie\gamma}\, \varphi_2 \equiv \neg\forall \pi_1, \pi_2, \ldots, \pi_n.\, \neg\varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \neg\varphi_2$$

$$\forall \pi_1, \pi_2, \ldots, \pi_n.\, \varphi_1\, \mathcal{R}_{\bowtie\gamma}\, \varphi_2 \equiv \neg\exists \pi_1, \pi_2, \ldots, \pi_n.\, \neg\varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \neg\varphi_2$$

$$\exists \pi_1, \pi_2, \ldots, \pi_n.\, \Diamond_{\bowtie\gamma}\varphi \equiv \exists \pi_1, \pi_2, \ldots, \pi_n.\, \top\, \mathcal{U}_{\bowtie\gamma}\, \varphi$$

$\tilde{\exists}$*HyperPTCTL* is a subclass of HyperPTCTL such that the (top-level) formulas are of the form $\tilde{\exists} p_1 \tilde{\exists} p_2 \ldots \tilde{\exists} p_n\, \psi$, where ψ has no quantifiers over the parameters. *PTCTL* [7] is a subclass of HyperPTCTL with only one path variable π. $\tilde{\exists}$*PTCTL* [7] is defined analogously.

We extend HyperPTCTL with counters and clocks to constrain the number of occurrences of atomic propositions and to compare the time difference between the occurrences of two atomic propositions, respectively. Intuitively, for each atomic proposition σ and path π, $\mathrm{LAST}(\sigma_\pi)$ indicates the time elapsed since the final switching of σ from false to true in π, and $\mathrm{COUNT}(\sigma_\pi)$ indicates the total number of switches of σ from false to true in π. To ensure decidability, we only allow specific forms of constraints. We denote the extended logic as *Ext-HyperPTCTL*.

Definition 4 (Syntax of Ext-HyperPTCTL): We extend the syntax of HyperPTCTL formulas of the temporal level as follows, where $\sigma \in \Sigma$, $\mathcal{V}$ is the set of path variables, $\pi \in \mathcal{V}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, $d, N \in \mathbb{N}$, lt is a linear term over $\mathbb{P}$, $\Sigma_{\mathrm{COUNT}} = \{\mathrm{COUNT}(\sigma_\pi) \mid \sigma \in \Sigma, \pi \in \mathcal{V}\}$, $cnt_{\geq 0}$ is a *non-negative* linear term over Σ_{COUNT} (i.e., $cnt_{\geq 0} = \sum_{\sigma \in \Sigma, \pi \in \mathcal{V}} \alpha_{\sigma,\pi} \mathrm{COUNT}(\sigma_\pi)$ for some $\alpha_{\sigma,\pi} \in \mathbb{N}$),[4] and cnt is a linear term over Σ_{COUNT} (i.e., $cnt = \sum_{\sigma \in \Sigma, \pi \in \mathcal{V}} \alpha_{\sigma,\pi} \mathrm{COUNT}(\sigma_\pi)$ for some $\alpha_{\sigma,\pi} \in \mathbb{Z}$)

$$\begin{aligned} \varphi ::= \top \mid \sigma_\pi \mid \mathrm{LAST}(\sigma_\pi) - \mathrm{LAST}(\sigma_\pi) \bowtie lt \\ \mid cnt_{\geq 0} \bowtie d \mid (cnt \bmod N)\ \bowtie d \mid \cdots. \end{aligned}$$

We call $\mathrm{LAST}(\sigma_\pi) - \mathrm{LAST}(\sigma_\pi) \bowtie lt$, $cnt_{\geq 0} \bowtie d$, and $(cnt \bmod N)\ \bowtie d$ as LastExpr, $\textsc{CountExpr}_{\geq 0}$, and $\textsc{CountExpr}_{\mathrm{mod}}$, respectively. We let *Ext-PTCTL* be the subclass of Ext-HyperPTCTL with only one path variable π.

Example 2 (Drift of Clock): Let H be the proposition showing the "high" value of a digital clock. For $p \in \mathbb{P}$, AtMostOneDiff $\equiv (\mathrm{COUNT}(\mathrm{H}_{\pi_1}) - \mathrm{COUNT}(\mathrm{H}_{\pi_2})) \bmod 4 \in \{0, 1, 3\}$ denotes that the deviation of $\mathrm{COUNT}(\mathrm{H}_{\pi_1})$ and $\mathrm{COUNT}(\mathrm{H}_{\pi_2})$ is at most by one, if we keep having AtMostOneDiff in the past, SameCount $\equiv (\mathrm{COUNT}(\mathrm{H}_{\pi_1}) - \mathrm{COUNT}(\mathrm{H}_{\pi_2})) \bmod 4 = 0$ denotes that the number of times the clock became high is identical (mod 4) over the two paths; and LargeDeviation $\equiv \mathrm{LAST}(\mathrm{H}_{\pi_1}) - \mathrm{LAST}(\mathrm{H}_{\pi_2}) \notin [-p, p]$ denotes that the final time the clock became high differs by at least p time units over two paths, i.e., consists in a "large deviation." Then, the following Ext-HyperPTCTL formula shows the drift of near periodic clocks of duration at least p time units, globally assuming AtMostOneDiff: $\exists \pi_1, \pi_2.\, (\mathrm{AtMostOneDiff})\, \mathcal{U}_{\geq 0}\, (\mathrm{SameCount} \wedge \mathrm{LargeDeviation})$.

Example 3 (Execution-Time Opacity): Let Private be the proposition showing the private locations of a (P)TA and Goal be the proposition showing the goal locations. The following Ext-HyperPTCTL formula shows a formulation of opacity focusing on the execution time [16], i.e., there are executions of duration p with and without visiting any private locations: $\exists \pi_1, \pi_2.\, (\neg\mathrm{Goal}_{\pi_1} \wedge \neg\mathrm{Goal}_{\pi_2})\, \mathcal{U}_{=p}\, (\mathrm{Goal}_{\pi_1} \wedge \mathrm{Goal}_{\pi_2} \wedge \mathrm{COUNT}(\mathrm{Private}_{\pi_1}) = 0 \wedge \mathrm{COUNT}(\mathrm{Private}_{\pi_2}) > 0)$. That is, the goal is not reached until, after exactly p time units (which encodes the unknown execution time), the goal is reached for both the paths, and one of them did not visit any private

[4] The restriction of $cnt_{\geq 0}$ is to construct finite observers in Section V-B. Our implementation accepts any linear term, which is encoded by variables in IMITATOR [4].

location (“COUNT($\mathrm{Private}_{\pi_1}$) $= 0$”) while the other one did. Note that, this differs from another style of opacity for TAs in [17].

Example 4 (Side-Channel Timing Attack [14]): Let Inv and Idle be the propositions denoting the invocation of a process and the idle state. For $i \in \{1, 2\}$, we let $\mathrm{SyncInv} \equiv \mathrm{Inv}_{\pi_1} = \mathrm{Inv}_{\pi_2}$, $\mathrm{ImmediateExec}_i \equiv \mathrm{Inv}_{\pi_i} \implies \neg\mathrm{Idle}_{\pi_i}$, $\mathrm{ExecBound}_i \equiv \mathrm{Idle}_{\pi_i} \implies (\mathrm{LAST}(\mathrm{Idle}_{\pi_i}) - \mathrm{LAST}(\mathrm{Inv}_{\pi_i}) < p_1)$, and $\mathrm{NearFinish} \equiv (\mathrm{Idle}_{\pi_1} = \mathrm{Idle}_{\pi_2}) \implies (\mathrm{LAST}(\mathrm{Idle}_{\pi_1}) - \mathrm{LAST}(\mathrm{Idle}_{\pi_2}) \in (-p_2, p_2))$. The following Ext-HyperPTCTL formula shows that the execution time of each process must be similar, which is necessary to prevent side-channel timing attacks: $\forall \pi_1, \pi_2. (\neg\mathrm{SyncInv})\ \mathcal{R}_{\geq 0}\ (\mathrm{ImmediateExec}_1 \land \mathrm{ImmediateExec}_2 \land \mathrm{ExecBound}_1 \land \mathrm{ExecBound}_2 \land \mathrm{NearFinish})$. More precisely, for any paths corresponding to two different sequences of process executions, while each pair of the processes has been invoked simultaneously, their execution must be within p_1, and the execution time of each pair of processes must not differ more than p_2 time units.

Example 5 (Unfairness of Schedulers): Let Sub^i and Run^i be the proposition showing the submission and execution of the job $i \in \{1, 2\}$. For $\mathrm{SyncSub} \equiv \mathrm{Sub}^1_{\pi_1} = \mathrm{Sub}^2_{\pi_2}$, $\mathrm{SameCount} \equiv (\mathrm{COUNT}(\mathrm{Run}^1_{\pi_1}) - \mathrm{COUNT}(\mathrm{Run}^2_{\pi_2})) \bmod 4 = 0$, and $\mathrm{LargeDeviation} \equiv \mathrm{LAST}(\mathrm{Run}^1_{\pi_1}) - \mathrm{LAST}(\mathrm{Run}^2_{\pi_2}) \notin (-5, 5)$, the following Ext-HyperPTCTL formula shows unfair scheduling of the jobs 1 and 2: $\exists \pi_1, \pi_2. (\mathrm{SyncSub})\ \mathcal{U}_{\geq 0}\ (\mathrm{SameCount} \land \mathrm{LargeDeviation})$. That is, given two paths corresponding to two different jobs, if they submit the job at the same time and eventually run, but with a time difference of ≥ 5 time units, then the scheduler is unfair.

Example 6 (Robust Observational Nondeterminism): Let $\{\mathrm{In}^i \mid i \in \{1, 2, \dots, m\}\}$ and $\{\mathrm{Out}^i \mid i \in \{1, 2, \dots, n\}\}$ be the set of input and output propositions, respectively. $\mathrm{SyncIn} \equiv \bigwedge_{i \in \{1,2,\dots,m\}} (\mathrm{In}^i_{\pi_1} = \mathrm{In}^i_{\pi_2})$ denotes that the inputs in two runs π_1 and π_2 are synchronized, $\mathrm{AtMostOneDiff} \equiv \bigwedge_{i \in \{1,2,\dots,n\}} (\mathrm{COUNT}(\mathrm{Out}^i_{\pi_1}) - \mathrm{COUNT}(\mathrm{Out}^i_{\pi_2})) \bmod 4 \in \{0, 1, 3\}$ denotes that the deviation of the $\mathrm{COUNT}(\mathrm{Out}^i_{\pi_1})$ and $\mathrm{COUNT}(\mathrm{Out}^i_{\pi_2})$ is at most one, if we keep having AtMostOneDiff in the past, and $\mathrm{LargeDeviation} \equiv \bigvee_{i \in \{1,2,\dots,n\}} ((\mathrm{COUNT}(\mathrm{Out}^i_{\pi_1}) - \mathrm{COUNT}(\mathrm{Out}^i_{\pi_2})) \bmod 4 = 0 \land \mathrm{LAST}(\mathrm{Out}^i_{\pi_1}) - \mathrm{LAST}(\mathrm{Out}^i_{\pi_2}) \notin [-p, p])$ denotes that there is an output proposition that the number of times of the proposition became true is identical (mod 4) over the two paths but the timing differs at least p time units over the two paths. The following Ext-HyperPTCTL formula shows the robust observational nondeterminism assuming AtMostOneDiff, i.e., even if the inputs are given at the same timing, the output timing may deviate more than p: $\exists \pi_1, \pi_2. (\mathrm{SyncIn} \land \mathrm{AtMostOneDiff})\ \mathcal{U}_{\geq 0}\ (\mathrm{LargeDeviation})$.

B. Semantics of (Ext-)HyperPTCTL

Before defining the semantics of HyperPTCTL, we formalize the assignments of the paths. In addition to the partial function assigning the paths, we use a total preorder to fix the order of the discrete transitions at the same time-point.

Definition 5 (Path Assignments): For the path variables $\mathcal{V}$ and a TA $v(\mathcal{A})$, a *path assignment* $(\Pi, \trianglelefteq_\Pi)$ is a pair of a partial function $\Pi\colon \mathcal{V} \nrightarrow \mathrm{Paths}(v(\mathcal{A}))$ from path variables $\mathcal{V}$ to the paths $\mathrm{Paths}(v(\mathcal{A}))$ of $v(\mathcal{A})$ and a total preorder $\trianglelefteq_\Pi$ on $\mathbf{dom}(\Pi) \times \mathbb{N}$, such that for any $\pi, \pi' \in \mathbf{dom}(\Pi)$ and $i, j \in \mathbb{N}$, $i < j$ implies $(\pi, i) \trianglelefteq_\Pi (\pi, j)$ and $(\pi, j) \ntrianglelefteq_\Pi (\pi, i)$, $(\pi, i) \trianglelefteq_\Pi (\pi', j)$ implies $\sum_{k=0}^{i} d_k^\pi \leq \sum_{k=0}^{j} d_k^{\pi'}$, and $\sum_{k=0}^{i} d_k^\pi < \sum_{k=0}^{j} d_k^{\pi'}$ implies $(\pi, i) \trianglelefteq_\Pi (\pi', j)$, where d_k^π and $d_k^{\pi'}$ are the kth delay in $\Pi(\pi)$ and $\Pi(\pi')$, respectively.

We let $(\Pi_\emptyset, \trianglelefteq_{\Pi_\emptyset})$ be the empty path assignment, i.e., the path assignment satisfying $\mathbf{dom}(\Pi_\emptyset) = \emptyset$. For the path assignments $(\Pi, \trianglelefteq_\Pi)$ and $(\Pi', \trianglelefteq_{\Pi'})$, $(\Pi', \trianglelefteq_{\Pi'})$ is an *extension* of $(\Pi, \trianglelefteq_\Pi)$ if we have $\mathbf{dom}(\Pi) \subseteq \mathbf{dom}(\Pi')$ and for any $\pi, \pi' \in \mathbf{dom}(\Pi)$ and $i, j \in \mathbb{N}$, we have $(\pi, i) \trianglelefteq_\Pi (\pi', j) \iff (\pi, i) \trianglelefteq_{\Pi'} (\pi', j)$ and $\Pi(\pi) = \Pi'(\pi)$. For the path assignments $(\Pi, \trianglelefteq_\Pi)$ and $(\Pi', \trianglelefteq_{\Pi'})$, we let $(\Pi, \trianglelefteq_\Pi) \succeq (\Pi', \trianglelefteq_{\Pi'})$ (resp. $(\Pi, \trianglelefteq_\Pi) \succ (\Pi', \trianglelefteq_{\Pi'})$) if $\mathbf{dom}(\Pi) = \mathbf{dom}(\Pi')$ and there is $d \in \mathbb{R}_{\geq 0}$, such that for any $\pi \in \mathbf{dom}(\Pi)$, we have $\Pi(\pi) \succeq \Pi'(\pi)$ (resp. $\Pi(\pi) \succ \Pi'(\pi)$) and $\mathit{Dur}(\Pi(\pi) - \Pi'(\pi)) = d$, and for any $\pi, \pi' \in \mathbf{dom}(\Pi)$ and $i, j \in \mathbb{N}$, we have $(\pi, i) \trianglelefteq_{\Pi'} (\pi', j) \iff (\pi, i + \mathcal{D}_{\Pi(\pi)}^{\Pi'(\pi)}) \trianglelefteq_\Pi (\pi', j + \mathcal{D}_{\Pi(\pi')}^{\Pi'(\pi')})$, and if $\mathcal{D}_{\Pi(\pi)}^{\Pi'(\pi)} \geq 1$ holds, $(\pi, \mathcal{D}_{\Pi(\pi)}^{\Pi'(\pi)} - 1) \trianglelefteq_\Pi (\pi', \mathcal{D}_{\Pi(\pi')}^{\Pi'(\pi')})$ and $(\pi', \mathcal{D}_{\Pi(\pi')}^{\Pi'(\pi')}) \ntrianglelefteq_\Pi (\pi, \mathcal{D}_{\Pi(\pi)}^{\Pi'(\pi)} - 1)$ also hold. We let $\mathit{Dur}(\Pi - \Pi') = d$ for such Π, Π', and d.

Definition 6 (Semantics of HyperPTCTL): Let $\mathcal{A}$ be a PTA over parameters $\mathbb{P}_1$; given a HyperPTCTL formula over the parameters $\mathbb{P}_2$, let $\mathbb{P} = \mathbb{P}_1 \cup \mathbb{P}_2$. For a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, a path assignment $(\Pi, \trianglelefteq_\Pi)$, and a concrete state s of $v(\mathcal{A})$, the satisfaction relation of the temporal level HyperPTCTL formulas is defined as follows.

1) $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \sigma_\pi$ if $\pi \in \mathbf{dom}(\Pi)$ and $\sigma \in \Lambda(\mathit{Init}(\Pi(\pi)))$.
2) $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \neg\varphi$ if we have $(\Pi, \trianglelefteq_\Pi), s \not\models_{v,\mathcal{A}} \varphi$.
3) $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \varphi_1 \lor \varphi_2$ if $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \varphi_1$ or $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \varphi_2$ holds.
4) $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \exists \pi_1, \pi_2, \dots, \pi_n. \varphi_1 \mathcal{U}_{\bowtie\gamma} \varphi_2$ if for some extension $(\Pi^1, \trianglelefteq_{\Pi^1})$ of $(\Pi, \trianglelefteq_\Pi)$ satisfying $\mathbf{dom}(\Pi^1) = \mathbf{dom}(\Pi) \sqcup \{\pi_1, \pi_2, \dots, \pi_n\}$ and $\Pi^1(\pi_i) \in \mathrm{Paths}(v(\mathcal{A}), s)$ for each $i \in \{1, 2, \dots, n\}$, there is $(\Pi^2, \trianglelefteq_{\Pi^2})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^2, \trianglelefteq_{\Pi^2})$, $\mathit{Dur}(\Pi^1 - \Pi^2) \bowtie v(\gamma)$, $(\Pi^2, \trianglelefteq_{\Pi^2}), \mathit{Init}(\Pi^2(\pi_n)) \models_{v,\mathcal{A}} \varphi_2$, and for any $(\Pi^3, \trianglelefteq_{\Pi^3})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^3, \trianglelefteq_{\Pi^3}) \succ (\Pi^2, \trianglelefteq_{\Pi^2})$, $(\Pi^3, \trianglelefteq_{\Pi^3}), \mathit{Init}(\Pi^3(\pi_n)) \models_{v,\mathcal{A}} \varphi_1$ holds.
5) $(\Pi, \trianglelefteq_\Pi), s \models_{v,\mathcal{A}} \forall \pi_1, \pi_2, \dots, \pi_n. \varphi_1 \mathcal{U}_{\bowtie\gamma} \varphi_2$ if for any extension $(\Pi^1, \trianglelefteq_{\Pi^1})$ of $(\Pi, \trianglelefteq_\Pi)$ satisfying $\mathbf{dom}(\Pi^1) = \mathbf{dom}(\Pi) \sqcup \{\pi_1, \pi_2, \dots, \pi_n\}$ and $\Pi^1(\pi_i) \in \mathrm{Paths}(v(\mathcal{A}), s)$ for each $i \in \{1, 2, \dots, n\}$, there is $(\Pi^2, \trianglelefteq_{\Pi^2})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^2, \trianglelefteq_{\Pi^2})$, $\mathit{Dur}(\Pi^1 - \Pi^2) \bowtie v(\gamma)$, $(\Pi^2, \trianglelefteq_{\Pi^2}), \mathit{Init}(\Pi^2(\pi_n)) \models_{v,\mathcal{A}} \varphi_2$, and for any $(\Pi^3, \trianglelefteq_{\Pi^3})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^3, \trianglelefteq_{\Pi^3}) \succ (\Pi^2, \trianglelefteq_{\Pi^2})$, $(\Pi^3, \trianglelefteq_{\Pi^3}), \mathit{Init}(\Pi^3(\pi_n)) \models_{v,\mathcal{A}} \varphi_1$ holds.

For a PTA $\mathcal{A}$, a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, and a temporal-level HyperPTCTL formula φ, we write $\mathcal{A} \models_v \varphi$ if we have $(\Pi_\emptyset, \trianglelefteq_{\Pi_\emptyset}), s_0 \models_{v,\mathcal{A}} \varphi$, where s_0 is an initial state of $v(\mathcal{A})$.

For a PTA $\mathcal{A}$ and a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, the satisfaction relation of the top-level HyperPTCTL formulas is defined as follows.

1) $\mathcal{A} \models_v p \bowtie lt_{\geq 0}$ if we have $v(p) \bowtie v(lt_{\geq 0})$, where $v(lt_{\geq 0})$ denotes the expression obtained by replacing each p with $v(p)$ in $lt_{\geq 0}$.
2) $\mathcal{A} \models_v \neg\psi$ if we have $\mathcal{A} \not\models_v \psi$.
3) $\mathcal{A} \models_v \psi_1 \vee \psi_2$ if we have $\mathcal{A} \models_v \psi_1$ or $\mathcal{A} \models_v \psi_2$.
4) $\mathcal{A} \models_v \tilde{\exists} p\, \psi$ if there is $v' \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$ satisfying $v(p') = v'(p')$ for any $p' \in \mathbb{P} \setminus \{p\}$ and $\mathcal{A} \models_{v'} \psi$.

Example 7: Consider the formula $\varphi : \tilde{\exists} p_2 (p_2 > p_1 \wedge \exists \pi_1, \pi_2. (\mathrm{L}_{\pi_1} \implies \mathrm{H}_{\pi_2})\, \mathcal{U}_{=p_2} (\mathrm{H}_{\pi_1} \wedge \mathrm{H}_{\pi_2}))$. Fix $v(p_1) = 1.8$. For the PTA $\mathcal{A}$ in Fig. 2, we have $\mathcal{A} \models_v \varphi$ with $v(p_2) = 2.0$, and $\Pi^1(\pi_1)$ and $\Pi^1(\pi_2)$ are as follows: in π_1, we jump from ℓ_0 to ℓ_1 at 1.5 and jump from ℓ_1 to ℓ_0 at 2.0; in π_2, we jump from ℓ_0 to ℓ_1 at 0.5 and jump from ℓ_1 to ℓ_0 at 1.0.

To define the semantics of Ext-HyperPTCTL, we introduce the valuations of $\mathrm{COUNT}(\sigma_\pi)$ and $\mathrm{LAST}(\sigma_\pi)$. A counter valuation is a function $\eta \colon \Sigma \times \mathcal{V} \to \mathbb{N}$. A recording valuation is a function $\theta \colon \Sigma \times \mathcal{V} \to \mathbb{R}_{\geq 0}$. We write $\vec{0}_{\mathrm{cnt}}$ and $\vec{0}_{\mathrm{rec}}$ for the counter and recording valuations assigning 0 to all $(\sigma, \pi) \in \Sigma \times \mathcal{V}$, respectively. For a linear term cnt over $\{\mathrm{COUNT}(\sigma_\pi) \mid \sigma \in \Sigma, \pi \in \mathcal{V}\}$, we let $\eta(cnt)$ be the inequality obtained by replacing $\mathrm{COUNT}(\sigma_\pi)$ with $\eta(\sigma, \pi)$ and $\mathit{Vars}(cnt) = \{\pi \in \mathcal{V} \mid \exists \sigma \in \Sigma, \alpha_{\sigma,\pi} \neq 0]\}$, with $cnt = \sum_{\sigma \in \Sigma, \pi \in \mathcal{V}} \alpha_{\sigma,\pi} \mathrm{COUNT}(\sigma_\pi)$.

For the paths ρ, ρ' satisfying $\rho \succeq \rho'$, we let $\mathit{Rising}(\sigma, \rho - \rho')$ be the set of positions s in ρ satisfying $\mathit{Init}(\rho) < s \leq \mathit{Init}(\rho')$, $\sigma \in \Lambda(s)$, and there is $\delta \in \mathbb{T}$ such that for any $s' < s$, $\mathit{Dur}(s) - \mathit{Dur}(s') < \delta$ implies $\sigma \notin \Lambda(s)$. Notice that $\mathit{Rising}(\sigma, \rho - \rho')$ is finite because $\mathit{Dur}(\rho - \rho')$ is finite and we have no Zeno behavior. For a counter valuation η and the path assignments Π, Π' satisfying $\Pi \succeq \Pi'$, $[\eta]_{\Pi - \Pi'}$ is the counter valuation such that for any $(\sigma, \pi) \in \Sigma \times \mathcal{V}$, $[\eta]_{\Pi - \Pi'}(\sigma, \pi) = \eta(\sigma, \pi) + |\mathit{Rising}(\sigma, \Pi(\pi) - \Pi'(\pi))|$ if $\pi \in \mathbf{dom}(\Pi)$ and $[\eta]_{\Pi - \Pi'}(\sigma, \pi) = \eta(\sigma, \pi)$ otherwise. For a recording valuation θ and path assignments Π, Π' satisfying $\Pi \succeq \Pi'$, $[\theta]_{\Pi - \Pi'}$ is the recording valuation such that for any $(\sigma, \pi) \in \Sigma \times \mathcal{V}$, $[\theta]_{\Pi - \Pi'}(\sigma, \pi) = \mathit{Dur}(\Pi - \Pi')$ if $\pi \notin \mathbf{dom}(\Pi)$ or $\mathit{Rising}(\sigma, \Pi(\pi) - \Pi'(\pi)) = \emptyset$ and otherwise, $[\theta]_{\Pi - \Pi'}(\sigma, \pi)$ is the duration of $\mathit{Init}(\Pi'(\pi))$ in the suffix of $\Pi(\pi)$ starting from the final position in $\mathit{Rising}(\sigma, \Pi(\pi) - \Pi'(\pi))$.

Definition 7 (Semantics of Ext-HyperPTCTL): Let $\mathcal{A}$ be a PTA over parameters $\mathbb{P}_1$; given an Ext-HyperPTCTL formula over parameters $\mathbb{P}_2$, let $\mathbb{P} = \mathbb{P}_1 \cup \mathbb{P}_2$. For a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, a path assignment $(\Pi, \trianglelefteq_\Pi)$, a concrete state s of $v(\mathcal{A})$, and counter and recording valuations η and θ, the satisfaction relation of the temporal level Ext-HyperPTCTL formulas is defined as follows.

1) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \sigma_\pi$ if $\pi \in \mathbf{dom}(\Pi)$ and $\sigma \in \Lambda(\mathit{Init}(\Pi(\pi)))$.
2) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \mathrm{LAST}(\sigma_\pi) - \mathrm{LAST}(\sigma'_{\pi'}) \bowtie lt$ if we have $\pi, \pi' \in \mathbf{dom}(\Pi)$ and $\theta(\sigma, \pi) - \theta(\sigma', \pi') \bowtie v(lt)$.
3) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} cnt_{\geq 0} \bowtie d$ if we have $\mathit{Vars}(cnt_{\geq 0}) \subseteq \mathbf{dom}(\Pi)$ and $\eta(cnt_{\geq 0}) \bowtie d$.
4) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} (cnt \bmod N) \bowtie d$ if we have $\mathit{Vars}(cnt) \subseteq \mathbf{dom}(\Pi)$ and $(\eta(cnt) \bmod N) \bowtie d$.
5) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \neg\varphi$ if we have $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \not\models_{v,\mathcal{A}} \varphi$;
6) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \varphi_1 \vee \varphi_2$ if we have $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \varphi_1$ or $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \varphi_2$.
7) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \exists \pi_1, \pi_2, \ldots, \pi_n. \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$ if for some extension $(\Pi^1, \trianglelefteq_{\Pi^1})$ of $(\Pi, \trianglelefteq_\Pi)$ satisfying $\mathbf{dom}(\Pi^1) = \mathbf{dom}(\Pi) \sqcup \{\pi_1, \pi_2, \ldots, \pi_n\}$ and $\Pi^1(\pi_i) \in \mathrm{Paths}(v(\mathcal{A}), s)$ for each $i \in \{1, 2, \ldots, n\}$, there is $(\Pi^2, \trianglelefteq_{\Pi^2})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^2, \trianglelefteq_{\Pi^2})$, $\mathit{Dur}(\Pi^1 - \Pi^2) \bowtie v(\gamma)$, $(\Pi^2, \trianglelefteq_{\Pi^2}), \mathit{Init}(\Pi^2(\pi_n)), [\eta]_{\Pi^1 - \Pi^2}, [\theta]_{\Pi^1 - \Pi^2} \models_{v,\mathcal{A}} \varphi_2$, and for any $(\Pi^3, \trianglelefteq_{\Pi^3})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^3, \trianglelefteq_{\Pi^3}) \succ (\Pi^2, \trianglelefteq_{\Pi^2})$, we have $(\Pi^3, \trianglelefteq_{\Pi^3}), \mathit{Init}(\Pi^3(\pi_n)), [\eta]_{\Pi^1 - \Pi^3}, [\theta]_{\Pi^1 - \Pi^3} \models_{v,\mathcal{A}} \varphi_1$.
8) $(\Pi, \trianglelefteq_\Pi), s, \eta, \theta \models_{v,\mathcal{A}} \forall \pi_1, \pi_2, \ldots, \pi_n. \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$ if for any extension $(\Pi^1, \trianglelefteq_{\Pi^1})$ of $(\Pi, \trianglelefteq_\Pi)$ satisfying $\mathbf{dom}(\Pi^1) = \mathbf{dom}(\Pi) \sqcup \{\pi_1, \pi_2, \ldots, \pi_n\}$ and $\Pi^1(\pi_i) \in \mathrm{Paths}(v(\mathcal{A}), s)$ for each $i \in \{1, 2, \ldots, n\}$, there is $(\Pi^2, \trianglelefteq_{\Pi^2})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^2, \trianglelefteq_{\Pi^2})$, $\mathit{Dur}(\Pi^1 - \Pi^2) \bowtie v(\gamma)$, $(\Pi^2, \trianglelefteq_{\Pi^2}), \mathit{Init}(\Pi^2(\pi_n)), [\eta]_{\Pi^1 - \Pi^2}, [\theta]_{\Pi^1 - \Pi^2} \models_{v,\mathcal{A}} \varphi_2$, and for any $(\Pi^3, \trianglelefteq_{\Pi^3})$ satisfying $(\Pi^1, \trianglelefteq_{\Pi^1}) \succeq (\Pi^3, \trianglelefteq_{\Pi^3}) \succ (\Pi^2, \trianglelefteq_{\Pi^2})$, we have $(\Pi^3, \trianglelefteq_{\Pi^3}), \mathit{Init}(\Pi^3(\pi_n)), [\eta]_{\Pi^1 - \Pi^3}, [\theta]_{\Pi^1 - \Pi^3} \models_{v,\mathcal{A}} \varphi_1$.

For a PTA $\mathcal{A}$, a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, and a temporal-level HyperPTCTL formula φ, we write $\mathcal{A} \models_v \varphi$ if we have $(\Pi_\emptyset, \trianglelefteq_{\Pi_\emptyset}), s_0, \vec{0}_{\mathrm{cnt}}, \vec{0}_{\mathrm{rec}} \models_{v,\mathcal{A}} \varphi$, where s_0 is an initial state of $v(\mathcal{A})$. The satisfaction relation of the top-level Ext-HyperPTCTL formulas is the same as that of HyperPTCTL.

C. Problems

Here, we formalize the problems we consider in this article. We consider each problem under both continuous-time and discrete-time semantics, i.e., $\mathbb{T}$ is either $\mathbb{R}_{\geq 0}$ or $\mathbb{N}$. We let $\mathbb{P}$ be the set of parameters shared between the PTA and the (Ext-)HyperPTCTL formula.

(Ext-)HyperPTCTL model checking problem:
INPUT: PTA $\mathcal{A}$ and a top-level (Ext-)HyperPTCTL formula ψ
PROBLEM: Decide if there is $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$ satisfying $\mathcal{A} \models_v \psi$

(Ext-)HyperPTCTL parameter synthesis problem:
INPUT: PTA $\mathcal{A}$ and a top-level (Ext-)HyperPTCTL formula ψ
PROBLEM: Return the set $\{v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}} \mid \mathcal{A} \models_v \psi\}$

The solution to the latter problem can be *effectively computed* whenever its representation is symbolic, and can be represented by decidable formalisms, typically a finite union of polyhedra.

Let ψ be a top-level HyperPTCTL formula with no quantifiers over the parameters. The *emptiness* of the parameter valuations to have $\mathcal{A} \models_v \psi$ can be checked by model checking of ψ. The *universality* of the parameter valuations to have $\mathcal{A} \models_v \psi$ can be checked by model checking of $\neg(\tilde{\exists} p_1 \tilde{\exists} p_2 \ldots \tilde{\exists} p_n \neg\psi)$, where $\mathbb{P} = \{p_1, p_2, \ldots, p_n\}$.

In the remainder of this article, we focus on the *nest-free* fragment of HyperPTCTL (e.g., "Nest-Free HyperPTCTL," "Nest-Free Ext-HyperPTCTL," and "Nest-Free $\tilde{\exists}$HyperPTCTL" ...), i.e., fragments with no nesting of the temporal operators. Observe that all our Example 3, 2, 5, and 6 fit into this nest-free fragment. The following is the definition of Nest-Free HyperPTCTL. The other fragments are defined similarly.

Definition 8 (Syntax of Nest-Free HyperPTCTL): For the atomic propositions Σ and the parameters $\mathbb{P}$, the syntax of Nest-Free HyperPTCTL formulas of the Boolean level $\mathcal{B}$, the temporal level φ, and the top level ψ are defined as follows, where $\sigma \in \Sigma$, $\pi, \pi_1, \pi_2, \dots, \pi_n \in \mathcal{V}$, $\gamma \in \mathbb{P} \cup \mathbb{N}$, $p \in \mathbb{P}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, and $lt_{\geq 0}$ is a non-negative linear term over $\mathbb{P}$

$$\mathcal{B} ::= \top \mid \sigma_\pi \mid \neg\mathcal{B} \mid \mathcal{B} \vee \mathcal{B}$$

$$\varphi ::= \exists \pi_1, \pi_2, \dots, \pi_n.\, \mathcal{B}\, \mathcal{U}_{\bowtie\gamma}\, \mathcal{B} \mid \forall \pi_1, \pi_2, \dots, \pi_n.\, \mathcal{B}\, \mathcal{U}_{\bowtie\gamma}\, \mathcal{B}$$

$$\psi ::= \varphi \mid p \bowtie lt_{\geq 0} \mid \neg\psi \mid \psi \vee \psi \mid \tilde{\exists} p\, \psi.$$

V. Reduction of Nest-Free Ext-HyperPTCTL Synthesis to PTCTL Synthesis

A. Reduction of Path Variables via Self-Composition of PTAs

We show that the model checking and parameter synthesis of nest-free (Ext-)HyperPTCTL is reducible to ones of (Ext-)PTCTL by self-composition of PTAs. For a PTA $\mathcal{A} = (\Sigma, L, L_0, \mathbb{C}, \mathbb{P}, I, E, \Lambda)$ and $n \in \mathbb{N}_{>0}$, we let $\mathcal{A}^n = \underbrace{\mathcal{A} \mid\mid \mathcal{A} \mid\mid \dots \mid\mid \mathcal{A}}_{\text{n times}}$, and for each $i \in \{1, 2, \dots, n\}$ and $\sigma \in \Sigma$ (resp, $c \in \mathbb{C}$), we denote the corresponding atomic proposition in Σ^n (resp, clock in $\mathbb{C}^n$) of the ith component as σ^i (resp, c^i), where Σ^n and $\mathbb{C}^n$ are the sets of atomic propositions and clocks in $\mathcal{A}^n$. We generalize the projection of paths to such n-compositions, i.e., for a path ρ of $\mathcal{A}^n$, we let $\rho|_i$ be the projection of ρ to the ith component.

We define an auxiliary function $reduce^n$ to "compose" the atomic propositions in (Ext-)HyperPTCTL formulas.

Definition 9 ($reduce^n$): For $n \in \mathbb{N}_{>0}$, the function $reduce^n$ from nest-free temporal-level Ext-HyperPTCTL formulas with atomic propositions Σ to nest-free temporal-level Ext-PTCTL formulas with atomic propositions Σ^n is inductively defined as follows with $reduce^n(\sum_{\sigma\in\Sigma, i\in\{1,2,\dots,n\}} \alpha_{\sigma,\pi_i} \mathrm{COUNT}(\sigma_{\pi_i})) = \sum_{\sigma^i\in\Sigma^n} \alpha_{\sigma^i,\pi} \mathrm{COUNT}(\sigma^i_\pi)$:

1) $reduce^n(\top) = \top$;
2) $reduce^n(\sigma_{\pi_i}) = \sigma^i_\pi$;
3) $reduce^n(\mathrm{LAST}(\sigma_{\pi_i}) - \mathrm{LAST}(\sigma_{\pi_j}) \bowtie lt) = \mathrm{LAST}(\sigma^i_\pi) - \mathrm{LAST}(\sigma^j_\pi) \bowtie lt$;
4) $reduce^n(cnt_{\geq 0} \bowtie d) = reduce^n(cnt_{\geq 0}) \bowtie d$;
5) $reduce^n((cnt \bmod N) \bowtie d) = (reduce^n(cnt) \bmod N) \bowtie d$;
6) $reduce^n(\neg\varphi) = \neg reduce^n(\varphi)$;
7) $reduce^n(\varphi_1 \vee \varphi_2) = reduce^n(\varphi_1) \vee reduce^n(\varphi_2)$;
8) $reduce^n(\exists \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2) = \exists \pi.\, reduce^n(\varphi_1)\, \mathcal{U}_{\bowtie\gamma}\, reduce^n(\varphi_2)$;
9) $reduce^n(\forall \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2) = \forall \pi.\, reduce^n(\varphi_1)\, \mathcal{U}_{\bowtie\gamma}\, reduce^n(\varphi_2)$.

Algorithm 1: Outline of Our Reduction of Nest-Free Ext-HyperPTCTL Synthesis to Nest-Free Ext-PTCTL Synthesis

Input: A PTA $\mathcal{A}$ and a Nest-Free Ext-HyperPTCTL formula ψ
Output: The set $\{v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}} \mid \mathcal{A} \models_v \psi\}$

1 **def** `reduceSynth(`$\mathcal{A}, \psi$`)`**:**
2   **switch** ψ **do**
3   **case** $\top$ **do return** $(\mathbb{Q}_{\geq 0})^{\mathbb{P}}$
4   **case** σ_{π_i} **or** $\mathrm{LAST}(\sigma_{\pi_i}) - \mathrm{LAST}(\sigma_{\pi_j})$ **or** $cnt_{\geq 0} \bowtie d$ **or** $(cnt \bmod N) \bowtie d$ **do**
      `// `ψ` does not hold for empty path assignments`
5   **return** $\emptyset$
6   **case** $\neg\psi$ **do**
7   **return** $(\mathbb{Q}_{\geq 0})^{\mathbb{P}} \setminus$ `reduceSynth(`$\mathcal{A}, \psi$`)`
8   **case** $\psi_1 \vee \psi_2$ **do**
9   **return** `reduceSynth(`$\mathcal{A}, \psi_1$`)`$\cup$`reduceSynth(`$\mathcal{A}, \psi_2$`)`
10   **case** $\exists \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$ **or** $\forall \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$ **do**
      `// Use nest-free Ext-PTCTL synthesis`
11   **return** `synthesisExtPTCTL(`$\mathcal{A}^n, reduce^n(\psi)$`)`
12   **case** $p \bowtie lt_{\geq 0}$ **do**
13   **return** $\{v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}} \mid v(p) \bowtie v(lt_{\geq 0})\}$
14   **case** $\tilde{\exists} p\, \psi$ **do**
15   pre $\leftarrow$ `reduceSynth(`$\mathcal{A}, \psi$`)`
16   **return** $\{v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}} \mid \exists v' \in \mathrm{pre}. \forall p' \in \mathbb{P} \setminus \{p\}.\, v(p') = v'(p')\}$

We naturally extend $reduce^n$ to top-level Nest-Free Ext-HyperPTCTL formulas.

Algorithm 1 outlines our reduction of the synthesis problem. The reduction of model checking is similar. Our reduction is inductive on the structure of the Ext-HyperPTCTL formula ψ. Since the path assignment $(\Pi, \trianglelefteq_\Pi)$ is empty, for atomic formulas, ψ is satisfied (line 3) or violated (line 4) independent of $\mathcal{A}$ and v. For Boolean cases, we obtain the result from the result of the reduction of the immediate subformula(s) (lines 7 and 9). For the temporal operators, we use the result of the synthesis for the composed PTA $\mathcal{A}^n$ and the reduced formula $reduce^n(\psi)$ (line 11). For the remaining cases, the result is independent of $\mathcal{A}$ (line 13) or obtained by un-constraining the result for p (line 16).

The correctness of Algorithm 1 is immediate from the following theorem.

Theorem 1: For a PTA $\mathcal{A}$, a temporal-level Nest-Free Ext-HyperPTCTL formulas $\varphi_\exists = \exists \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$ and $\varphi_\forall = \forall \pi_1, \pi_2, \dots, \pi_n.\, \varphi_1\, \mathcal{U}_{\bowtie\gamma}\, \varphi_2$, and a parameter valuation v, we have $\mathcal{A} \models_v \varphi_\exists$ (resp, $\mathcal{A} \models_v \varphi_\forall$) if and only if we have $\mathcal{A}^n \models_v reduce^n(\varphi_\exists)$ (resp, $\mathcal{A}^n \models_v reduce^n(\varphi_\forall)$).

Proof (Sketch): Since the other cases are similar, we only outline the proof of $\mathcal{A} \models_v \varphi_\exists \implies \mathcal{A}^n \models_v reduce^n(\varphi_\exists)$. Suppose $\mathcal{A} \models_v \varphi_\exists$ holds. By the semantics of Ext-HyperPTCTL, for some extension $(\Pi^1, \trianglelefteq_{\Pi^1})$ of $(\Pi_\emptyset, \trianglelefteq_{\Pi_\emptyset})$ satisfying $\mathbf{dom}(\Pi) = \{\pi_1, \pi_2, \dots, \pi_n\}$ and $\Pi^1(\pi_i) \in \mathrm{Paths}(v(\mathcal{A}))$ for each $i \in \{1, 2, \dots, n\}$, there is a suffix $(\Pi^2, \trianglelefteq_{\Pi^2})$ of $(\Pi^1, \trianglelefteq_{\Pi^1})$ such that we have $Dur(\Pi^1 - \Pi^2) \bowtie v(\gamma)$, φ_2 holds at $(\Pi^2, \trianglelefteq_{\Pi^2})$, and for any $(\Pi^3, \trianglelefteq_{\Pi^3})$

between $(\Pi^1, \trianglelefteq_{\Pi^1})$ and $(\Pi^2, \trianglelefteq_{\Pi^2})$, φ_1 holds at $(\Pi^3, \trianglelefteq_{\Pi^3})$. Since there are paths ρ and ρ' of $v(\mathcal{A}^n)$ such that 1) ρ' is a suffix of ρ and 2) for each $i \in \{1, 2, \ldots, n\}$, we have $\Pi^1(\pi_i) = \rho|_i$ and $\Pi^2(\pi_i) = \rho'|_i$, we can construct path assignments $(\overline{\Pi}^1, \trianglelefteq_{\overline{\Pi}^1})$ and $(\overline{\Pi}^2, \trianglelefteq_{\overline{\Pi}^2})$ mapping π to them. Notice that $\mathit{reduce}^n(\varphi_2)$ holds at such $(\overline{\Pi}^2, \trianglelefteq_{\overline{\Pi}^2})$. Moreover, for any path assignment $(\overline{\Pi}^3, \trianglelefteq_{\overline{\Pi}^3})$ between $(\overline{\Pi}^1, \trianglelefteq_{\overline{\Pi}^1})$ and $(\overline{\Pi}^2, \trianglelefteq_{\overline{\Pi}^2})$, since there is a corresponding path assignment $(\Pi^3, \trianglelefteq_{\Pi^3})$ between $(\Pi^1, \trianglelefteq_{\Pi^1})$ and $(\Pi^2, \trianglelefteq_{\Pi^2})$, $\mathit{reduce}^n(\varphi_1)$ holds at $(\overline{\Pi}^3, \trianglelefteq_{\overline{\Pi}^3})$ Therefore, we have $\mathcal{A}^n \models_v \mathit{reduce}^n(\varphi_\exists)$. ∎

B. Observers for Extended Predicates

We show that the satisfaction of the additional predicates $\mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$, $\mathit{cnt}_{\geq 0} \bowtie d$, and $(\mathit{cnt} \bmod N) \bowtie d$ in Ext-PTCTL are observable by a PTA, and thus, Ext-PTCTL model checking and synthesis are reducible to the PTCTL model checking and synthesis, respectively. Since we consider the Ext-PTCTL formulas, we assume $\mathcal{V} = \{\pi\}$ without loss of generality.

For LASTExpr $\mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$, since the truth value of $\mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$ changes only when the truth value of σ^1_π or σ^2_π changes from $\perp$ to $\top$, we can construct an observer by "re-evaluating" $\mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$ when σ^1 or σ^2 changes from false to true. Additionally, we use invariants so that the initial states depend on the parameter valuations.

For CountExpr$_{\geq 0}$ $\mathit{cnt}_{\geq 0} \bowtie d$, since $\mathrm{COUNT}(\sigma_\pi)$ is monotonically increasing, we can abstract the precise value once its value is sufficiently large. Therefore, we can encode the counted value by finite locations.

For CountExpr$_{\mathrm{mod}}$ $(\mathit{cnt} \bmod N) \bowtie d$, since the value of $\mathrm{COUNT}(\sigma_\pi)$ is cycling back at $N \in \mathbb{N}$, we can also encode the counted value modulo N by finite locations. Observers were studied in [3], and we define them as follows.

Definition 10 (Observers for LASTExpr*):* Let $\sigma^1, \sigma^2 \in \Sigma$, $\bowtie \in \{<, \leq, =, \geq, >\}$, and lt be a linear term over $\mathbb{P}$. The observer for $\varphi = \mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$ is a PTA $\mathcal{O}_\varphi = (L, L, L_0, \{c_{\sigma^1}, c_{\sigma^2}\}, \mathbb{P}, I, E, \Lambda)$, where $L = \mathcal{P}(\{\sigma^1, \sigma^2, \varphi\}) \times \{\top, \perp\}$, L_0 is $L_0 = \{(pr, b) \in L \mid b = \perp\}$, I is such that $I(\ell) = \top$ for any $\ell \notin L_0$, and for $\ell = (pr, \perp) \in L_0$, $I(\ell)$ is $0 \bowtie \mathit{lt}$ if $\varphi \in pr$ and otherwise, $I(\ell)$ is $\neg(0 \bowtie \mathit{lt})$, Λ is the identity function, and $E = \big\{((pr, b), \top, \emptyset, (pr', \top)) \mid pr' \subsetneq pr, \varphi \in pr \cap pr' \text{ or } \varphi \notin pr \cup pr'\big\} \cup \big\{((pr, b), (c_{\sigma^1} - c_{\sigma^2} \bowtie \mathit{lt})[\mathbb{C}_{\Sigma_{\mathrm{rise}}} := 0], \mathbb{C}_{\Sigma_{\mathrm{rise}}}, (pr' \cup \{\varphi\}, \top)) \mid pr' \subseteq \{\sigma^1, \sigma^2\}, \Sigma_{\mathrm{rise}} = pr' \setminus pr, \Sigma_{\mathrm{rise}} \neq \emptyset\big\} \cup \big\{((pr, b), \neg(c_{\sigma^1} - c_{\sigma^2} \bowtie \mathit{lt})[\mathbb{C}_{\Sigma_{\mathrm{rise}}} := 0], \mathbb{C}_{\Sigma_{\mathrm{rise}}}, (pr', \top)) \mid pr' \subseteq \{\sigma^1, \sigma^2\}, \Sigma_{\mathrm{rise}} = pr' \setminus pr, \Sigma_{\mathrm{rise}} \neq \emptyset\big\}$, where $\mathbb{C}_{\Sigma_{\mathrm{rise}}} = \{c_{\sigma^i} \mid \sigma^i \in \Sigma_{\mathrm{rise}}\}$ and $(c_{\sigma^1} - c_{\sigma^2} \bowtie \mathit{lt})[\mathbb{C}_{\Sigma_{\mathrm{rise}}} := 0]$ is $-c_{\sigma^2} \bowtie \mathit{lt}$ if $\mathbb{C}_{\Sigma_{\mathrm{rise}}} = \{c_{\sigma^1}\}$, $c_{\sigma^1} \bowtie \mathit{lt}$ if $\mathbb{C}_{\Sigma_{\mathrm{rise}}} = \{c_{\sigma^2}\}$, and $0 \bowtie \mathit{lt}$ if $\mathbb{C}_{\Sigma_{\mathrm{rise}}} = \{c_{\sigma^1}, c_{\sigma^2}\}$.

Definition 11 (Observers for CountExpr$_{\geq 0}$*):* Let $\mathit{cnt}_{\geq 0} = \sum_{\sigma \in \Sigma} \alpha_{\sigma,\pi} \mathrm{COUNT}(\sigma_\pi)$ be a non-negative linear term, i.e., $\alpha_{\sigma,\pi} \in \mathbb{N}$. The observer for $\varphi = \mathit{cnt}_{\geq 0} \bowtie d$ is a PTA $\mathcal{O}_\varphi = (\mathcal{P}(\Sigma \cup \{\varphi\}), \mathcal{P}(\Sigma) \times \{0, 1, \ldots, d, d+1\}^\Sigma, L_0, \emptyset, \emptyset, I, E, \Lambda)$, where $L_0 = \mathcal{P}(\Sigma) \times \{0\}^\Sigma$, $I(\ell) = \top$ for any $\ell \in L$, Λ is such that $\Lambda((pr, \tilde{\eta})) = pr \cup \{\varphi\}$ if $\sum_{\sigma \in \Sigma} \alpha_{\sigma,\pi} \tilde{\eta}(\sigma) \bowtie d$ holds and $\Lambda((pr, \tilde{\eta})) = pr$ otherwise, and $E = \{((pr, \tilde{\eta}), \top, \emptyset, (pr', \tilde{\eta}[pr' \setminus pr \mathrel{+}= 1])) \mid pr, pr' \subseteq \Sigma, \tilde{\eta} \in \{0, 1, \ldots, d, d+1\}^\Sigma\}$, where $\tilde{\eta}[pr' \setminus pr \mathrel{+}= 1]$ is such that $v[pr' \setminus pr \mathrel{+}= 1](\sigma) = v(\sigma)$ for $\sigma \notin pr' \setminus pr$, $\tilde{\eta}[pr' \setminus pr \mathrel{+}= 1](\sigma) = \tilde{\eta}(\sigma) + 1$ if $\sigma \in pr' \setminus pr$ and $\tilde{\eta}(\sigma) < d$, and $\tilde{\eta}[pr' \setminus pr \mathrel{+}= 1](\sigma) = d + 1$, otherwise.

We omit the definition of the observers for $\varphi = (\mathit{cnt} \bmod N) \bowtie d$ since it is similar to Definition 11. The main difference is to reset the "counter" $\tilde{\eta}$ to 0 when the value becomes N.

The observers semantically capture the original expressions intuitively because c_{σ^1} and c_{σ^2} correspond to $\mathrm{LAST}(\sigma^1_\pi)$ and $\mathrm{LAST}(\sigma^2_\pi)$, and $\tilde{\eta}$ is a sound abstraction of $[\vec{0}_{\mathrm{cnt}}]_{\Pi - \Pi'}$.

Lemma 1 (Correctness of the Observers): For each $\sigma \in \Sigma$, we let $\alpha_\sigma \in \mathbb{N}$ and $\alpha'_\sigma \in \mathbb{Z}$. Let $N \in \mathbb{N}$, $\sigma^1, \sigma^2 \in \Sigma$, $\bowtie \in \{<, \leq, =, \geq, >\}$, $d \in \mathbb{N}$, and lt be a linear term over $\mathbb{P}$. Let φ be one of the following: $\mathrm{LAST}(\sigma^1_\pi) - \mathrm{LAST}(\sigma^2_\pi) \bowtie \mathit{lt}$, $\sum_{\sigma \in \Sigma} \alpha_\sigma \mathrm{COUNT}(\sigma_\pi) \bowtie d$, or $(\sum_{\sigma \in \Sigma} \alpha'_\sigma \mathrm{COUNT}(\sigma_\pi) \bmod N) \bowtie d$. Let v be a valuation over $\mathbb{P}$, $(\Pi, \trianglelefteq_\Pi)$ be a path assignment satisfying $\mathbf{dom}(\Pi) = \{\pi\}$ and $\Pi(\pi) \in \mathrm{Paths}(v(\mathcal{O}_\varphi))$. For any $(\Pi', \trianglelefteq_{\Pi'})$ satisfying $(\Pi, \trianglelefteq_\Pi) \succeq (\Pi', \trianglelefteq_{\Pi'})$, we have $\Pi', \mathit{Init}(\Pi'(\pi)), [\vec{0}_{\mathrm{cnt}}]_{\Pi - \Pi'}, [\vec{0}_{\mathrm{rec}}]_{\Pi - \Pi'} \models_{v, \mathcal{O}_\varphi} \varphi$ if and only if we have $\varphi \in \Lambda(\mathit{Init}(\Pi'(\pi)))$.

For an Ext-PTCTL formula ψ, we let $\mathcal{O}_\psi$ be the PTA $\mathcal{O}_\psi = \times_{\mathit{ext} \in \mathrm{Ext}(\psi)} \mathcal{O}_{\mathit{ext}}$, where $\mathrm{Ext}(\psi)$ is the set of LASTExpr, CountExpr$_{\geq 0}$, and CountExpr$_{\mathrm{mod}}$ in ψ. For an Ext-PTCTL formula ψ, we let ψ_{noext} be the PTCTL formula with the same syntax but having $\mathrm{Ext}(\psi)$ as additional atomic propositions. The following shows that the model checking and synthesis for Ext-PTCTL formulas are reducible to those for the PTCTL formulas.

Theorem 2 (Correctness of the Reduction With $\mathcal{O}_\psi$): For a PTA $\mathcal{A}$, a parameter valuation $v \in (\mathbb{Q}_{\geq 0})^{\mathbb{P}}$, and a top-level Ext-PTCTL formula ψ, we have $\mathcal{A} \models_v \psi$ if and only if $\mathcal{A} \times \mathcal{O}_\psi \models_v \psi_{\mathrm{noext}}$.

Proof (Sketch): Since the other cases are similar, we only outline the proof of $\mathcal{A} \models_v \varphi_\exists \implies \mathcal{A} \times \mathcal{O}_\psi \models_v \varphi_{\exists\mathrm{noext}}$, where $\varphi_\exists = \exists \pi .\, \varphi_1 \, \mathcal{U}_{\bowtie \gamma} \, \varphi_2$. Moreover, since we have $\mathcal{V} = \{\pi\}$, we discuss it based on the paths rather than the path assignments for simplicity. Suppose $\mathcal{A} \models_v \varphi_\exists$ holds. By the semantics of ExtPTCTL, there is a path ρ_1 of $v(\mathcal{A})$ and a suffix ρ_2 of ρ_1 such that $\mathit{Dur}(\rho_1 - \rho_2) \bowtie v(\gamma)$, φ_2 holds at ρ_2, and φ_1 holds at any position between ρ_1 and ρ_2. Since the observer $\mathcal{O}_\psi$ is complete, there is a path ρ'_1 of $v(\mathcal{A} \times \mathcal{O}_\psi)$ satisfying $\rho_1|_{v(\mathcal{A})} = \rho'_1$. Moreover, by taking a suitable suffix of ρ'_1, there is a path ρ'_2 of $v(\mathcal{A} \times \mathcal{O}_\psi)$ satisfying $\rho'_2|_{v(\mathcal{A})} = \rho_2$. By Lemma 1, $\varphi_{2\mathrm{noext}}$ holds at ρ'_2. For any position between ρ'_1 and ρ'_2, since there is a corresponding position between ρ_1 and ρ_2, $\varphi_{1\mathrm{noext}}$ holds at that position. Therefore, we have $\mathcal{A} \times \mathcal{O}_\psi \models_v \varphi_{\exists\mathrm{noext}}$. ∎

C. Worked Example

Here, we present an illustrative example of our Ext-HyperPTCTL synthesis semi-algorithm. Consider again the PTA $\mathcal{A}$ in Fig. 2. Let ψ be the Nest-Free Ext-HyperPTCTL formula in Example 2.

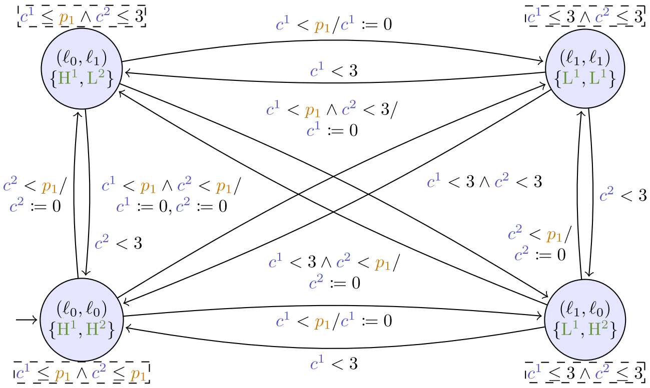

Fig. 3. Self-composition $\mathcal{A} \,||\, \mathcal{A}$ of $\mathcal{A}$ in Fig. 2.

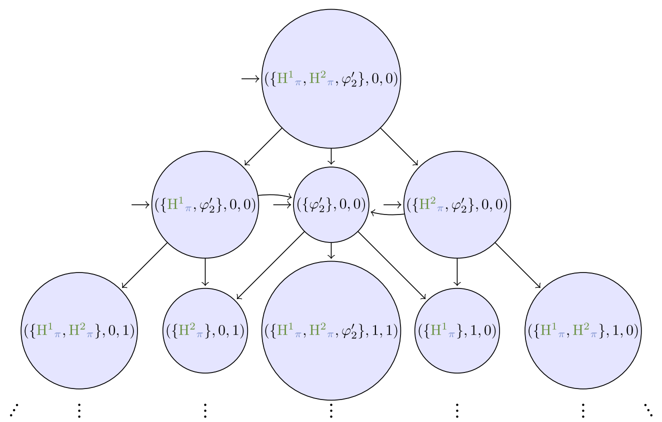

Fig. 4. Part of the observer of $\varphi_2' = (\mathrm{COUNT}(H^1{}_\pi) - \mathrm{COUNT}(H^2{}_\pi) \bmod 4) = 0$.

First, we reduce Nest-Free Ext-HyperPTCTL model checking to Nest-Free Ext-PTCTL model checking via the self-composition (Section V-A). Since ψ contains two path variables π_1 and π_2, we take the self-composition $\mathcal{A} \,||\, \mathcal{A}$ of $\mathcal{A}$ (Fig. 3). The corresponding Nest-Free Ext-PTCTL formula is $\mathit{reduce}^2(\psi) = \exists \pi . \varphi_1' \,\mathcal{U}_{\geq 0}\, (\varphi_2' \wedge \varphi_3')$, where $\varphi_1' = \mathit{reduce}^2(\mathrm{AtMostOneDiff})$, $\varphi_2' = \mathit{reduce}^2(\mathrm{SameCount})$, and $\varphi_3' = \mathit{reduce}^2(\mathrm{LargeDeviation})$. Then, we construct the observers $\mathcal{O}_{\varphi_1'}$, $\mathcal{O}_{\varphi_2'}$, and $\mathcal{O}_{\varphi_3'}$. Figs. 4 and 5 show a part of $\mathcal{O}_{\varphi_2'}$ and $\mathcal{O}_{\varphi_3'}$. Finally, we apply the PTCTL synthesis to $(\mathcal{A} \,||\, \mathcal{A}) \times \mathcal{O}_{\varphi_1'} \times \mathcal{O}_{\varphi_2'} \times \mathcal{O}_{\varphi_3'}$ with $\mathit{reduce}^2(\psi)$. In this case, the synthesized parameter constraint is as follows, where $2 \times p_1 > p_2 \wedge 3 \times p_1 + 3 > 2 \times p_2 \wedge p_1 + 3 > p_2 \wedge p_1$. We remark that our implementation supports more general formulas, e.g., $\exists \pi_1, \pi_2 . \Diamond_{\geq 0}(\mathrm{SameCount}' \wedge \mathrm{LargeDeviation})$, with $\mathrm{SameCount}' \equiv \mathrm{COUNT}(H_{\pi_1}) = \mathrm{COUNT}(H_{\pi_2})$.

VI. Decidable Subclasses

The model checking problem (and the synthesis counterpart) against the general PTAs is trivially undecidable, even for the nest-free existential fragment.

Proposition 1: Model checking PTAs against a Nest-Free $\tilde{\exists}$HyperPTCTL formula is undecidable.

Proof: The Nest-Free $\tilde{\exists}$HyperPTCTL formula "$\exists \pi . \Diamond_{\geq 0} \sigma$" is equivalent to the TCTL formula $\exists \Diamond \sigma$ denoting reachability. Reachability-emptiness is known to be undecidable for PTAs [1], which gives the result. ∎

This negative result leads us to exhibit subclasses of either the model or the formula for which decidability can be achieved, which we do in the following.

A. Nonparametric Model Against Parametric Formula

We consider here nonparametric TAs against a restriction of Nest-Free Ext-HyperPTCTL defined as follows: 1) Parameters cannot be used in LAST; 2) Parameters are integer-valued. Put it differently, parameters cannot be used in the extended syntax (the constructs that are turned into observers during our transformation in Section V-B); we insist that parameters can be used anywhere else in the formula. Let Nest-Free RPExt-HyperPTCTL denote this class (with "RP" denoting a restricted use of parameters). For instance, the opacity in Example 3 is in this class.

Theorem 3 (Complexity of the Model Checking Problem): Model checking TAs against a Nest-Free RPExt-HyperPTCTL formula is in 6EXPTIME.

Proof: We reduce to model checking a nonparametric TA against PTCTL [7]. Recall that our general construction (Section V) reduces model checking a PTA against a Nest-Free Ext-HyperPTCTL formula to the model checking a network of PTAs and a set of observers against a PTCTL formula. Since the observers are created for the extended syntax, and since they do not contain parameters in Nest-Free RPExt-HyperPTCTL, the synchronized product of the multiple TAs and the observers does not contain the parameters.

Now, model checking a nonparametric TA against PTCTL can be done in 5EXPTIME in the synchronized product of the size of $\mathcal{A}$ and ψ from [7, Th. 7.5]. Therefore, since $|\mathcal{A}^n| \leq |\mathcal{A}|^{|\psi|}$ and due to the fact that the observers are in constant size, and come in number at most linear in $|\psi|$, model checking a TA against a Nest-Free RPExt-HyperPTCTL formula is in 6EXPTIME.

Observe that this is only an upper bound because; 1) we only provide an one-way reduction; 2) the complexity in [7, Th. 7.5] only gives an upper bound anyway. ∎

Remark 1: Following the same reasoning, the model checking a TA against a formula expressed in Nest-Free $\tilde{\exists}$RPExt-HyperPTCTL is in 4EXPTIME, reusing the fact that the model checking a nonparametric TA against $\tilde{\exists}$PTCTL can be done in 3EXPTIME [7, Proposition 7.6].

Theorem 4 (Effective Parameter Synthesis): The solution to the parameter synthesis problem for the TAs against a Nest-Free Ext-HyperPTCTL formula can be effectively computed.

Proof: As in the proof of Theorem 3, we reduce to the model checking a nonparametric TA against PTCTL [7]. From [7], the solution of the parameter synthesis of a nonparametric TA against PTCTL can be effectively computed. ∎

B. L/U-PTAs Against Nest-Free Ext-HyperPTCTL

Definition 12 (L/U-PTA [10]): An *L/U-PTA* (lower-bound/upper-bound PTA) is a PTA where the set of parameters is partitioned into the lower-bound and upper-bound parameters, where each upper-bound (resp, lower-bound) parameter p_i must be such that, for every guard or invariant constraint $c \bowtie p_i$, we have $\bowtie \in \{\leq, <\}$ (resp, $\bowtie \in \{\geq, >\}$).

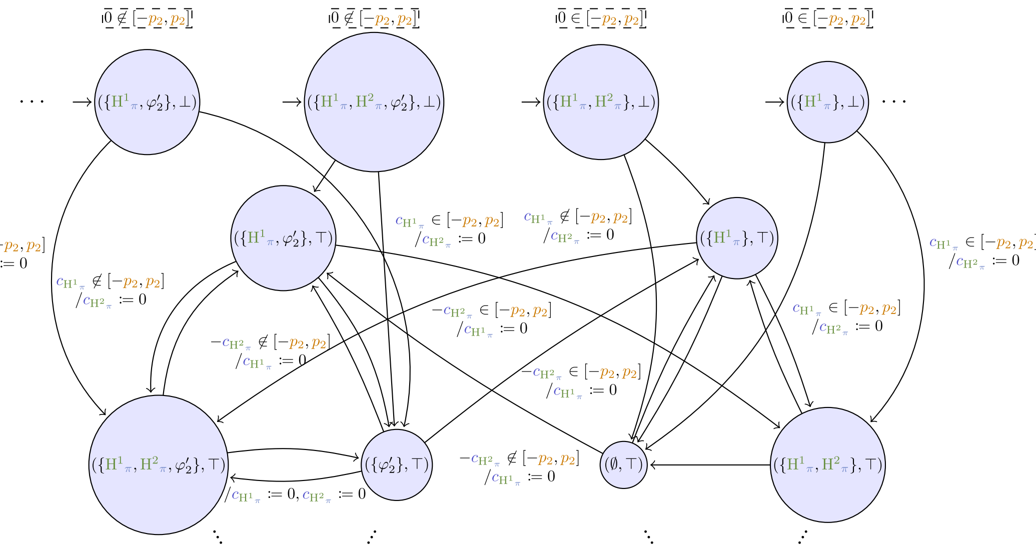

Fig. 5. Part of the observer of $\varphi'_3 = \text{LAST}(\text{H}^1{}_\pi) - \text{LAST}(\text{H}^2{}_\pi) \notin [-p_2, p_2]$. Most of the edges from the initial locations are omitted for simplicity. The initial satisfaction of φ'_3 is conditioned with the invariant.

Example 8: The PTA in Fig. 2 is an L/U-PTA with an upper-bound parameter p_1. The PTA in Fig. 5 is not L/U.

This is because the mere $\forall\Diamond$-emptiness (emptiness of the valuations set for which a location is always eventually reachable) is undecidable for L/U-PTAs [5], we restrict ourselves to the reachability fragment of Nest-Free Ext-HyperPTCTL, where the temporal operators are only $\exists\Diamond$. Let Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL denote this fragment.

Theorem 5 (Decidability of Nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL for L/U-PTAs): Model checking L/U-PTAs against a nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL formula is PSPACE-complete. The synthesis is, however, intractable.

Proof: We reduce to reachability for L/U-PTAs. Our general construction (Section V) reduces the model checking a PTA against a Nest-Free Ext-HyperPTCTL formula to the model checking a network of (L/U-)PTAs and a set of nonparametric observers against a TCTL formula. Here, we consider the reachability fragment only, leading to a reachability property. Reachability-emptiness is PSPACE-complete for the L/U-PTAs [10], and therefore the model checking L/U-PTAs against a nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL formula is PSPACE-complete (the hardness following immediately).

The nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL formula "$\exists\pi.\Diamond\sigma$" is equivalent to the (T)CTL formula $\exists\Diamond\sigma$ denoting reachability. Reachability-synthesis is known to be intractable for L/U-PTAs [5], and therefore the synthesis for the L/U-PTAs against a nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL is intractable. ■

By using as proof argument a result from [8] showing that nest-free TCTL emptiness is decidable for the L/U-PTAs with integer-valued parameters and *without invariants*, we can show as follows.

Theorem 6 (Decidability of Nonparametric Nest-Free Ext-HyperPTCTL for L/U-PTAs): Model checking L/U-PTAs with integer-valued parameters without invariants against a nonparametric Nest-Free Ext-HyperPTCTL formula is PSPACE-complete. The synthesis is however intractable.

C. $(1, *, 1)$-*PTAs Against Nonparametric Formula*

We use here a common notation $(n, *, m)$ to denote the n parametric clocks, arbitrarily many nonparametric clocks and m parameters. We finally show decidability in a restrictive setting, by reduction to the decidable setting of [6].

Theorem 7 (Decidability With One Discrete Clock): Model checking a $(1, *, 1)$-PTA is decidable over discrete time against a nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL with (at most) two path quantifiers for each temporal level formula.

Proof: We reduce to reachability for $(2, *, 1)$-PTAs. Model checking nonparametric Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL reduces to model checking a network of PTAs (including the observers necessary to encode the extended syntax of the formula) against a reachability property. Further, the model contains a single parametric clock, and the formula contains two paths quantifiers for each temporal level formula, leading the self-composed model to contain two parametric clocks. Since the (unique) parameter is shared between both the copies, then the resulting composition is a $(2, *, 1)$-PTA. Reachability-emptiness is EXPSPACE-complete for $(2, *, 1)$-PTAs for $\mathbb{T} = \mathbb{N}$ [6]. ■

It remains open whether the *synthesis* problem is tractable in this latter case.

This result is not tight in the number of clocks, in the sense that it remains open whether the model checking a $(2, *, 1)$-PTA against Nest-Free $\tilde{\exists}$-$\exists\Diamond$Ext-HyperPTCTL with two path quantifiers per temporal level formula is decidable or not. However, by allowing $\exists\mathcal{U}$ instead of $\exists\Diamond$ in the formula, we can show undecidability. The proof encodes a $(4, 0, 1)$-PTA (for which $\exists\Diamond$-emptiness is undecidable [18]) into two $(2, 0, 1)$-PTAs, the synchronization of which is enforced thanks to an $\exists\mathcal{U}$-based Nest-Free $\tilde{\exists}$HyperPTCTL.

Theorem 8 (Undecidability Over Discrete Time With Two Clocks): Model checking a $(2, 0, 1)$-PTA is undecidable over discrete time against a nonparametric Nest-Free $\tilde{\exists}$HyperPTCTL formula using only $\exists\mathcal{U}$ and two path quantifiers.

Proof (Sketch): We reduce from the reachability for $(3, 0, 1)$-PTAs [18]. Let $\mathcal{A}$ be a $(4, 0, 1)$-PTA with the clocks t, x, y, and z. Let us "split" it into two $(2, 0, 1)$-PTAs with the same structure (same locations and edges), such that the first (resp, second) PTA only contains clock constraints containing t and x (resp, y and z). Add to each location of the first PTA an unique location label with the same name as the location, i.e., $\Lambda(\ell_i) = \{\ell_i\}$, and to the second a primed label, i.e., $\Lambda(\ell_i) = \{\ell'_i\}$. Fix ℓ a target location. Then, let $\mathcal{A}'$ be the PTA union of these two PTAs, i.e., starting with an initial nondeterministic choice in 0-time, and then "choosing" between either components (we assume that y and z are renamed into t and x). Reaching ℓ in $\mathcal{A}$ is equivalent to checking the following Nest-Free $\tilde{\exists}$HyperPTCTL formula in $\mathcal{A}'$: $\exists\pi_1, \pi_2. (\bigwedge_i (\ell_i)_{\pi_1} = (\ell'_i)_{\pi_2})\, \mathcal{U}_{\geq 0}\, (\ell_{\pi_1} \wedge \ell'_{\pi_2})$.

Since $\exists\Diamond$-emptiness is undecidable for $(3, 0, 1)$-PTAs [18], then model checking this formula against $\mathcal{A}'$ is undecidable. $\mathcal{A}'$ contains only two clocks, and the formula is made of a single $\exists\mathcal{U}$, with only two path quantifiers. ∎

VII. Experiment

We experimentally evaluated the efficiency of our model checking semi-algorithm using our prototype tool HyPTCTLchecker.[5] Given a PTA and a Nest-Free Ext-HyperPTCTL formula, HyPTCTLchecker translates them into a PTA and a PTCTL formula via the reduction presented in Section V, and outputs them as the format supported by IMITATOR [4], a verification tool for PTAs. Then, we execute IMITATOR to solve the synthesis problem. HyPTCTLchecker supports all the Nest-Free Ext-HyperPTCTL formulas except for the following operator only because IMITATOR does not support its nonhyper versions: $\exists\pi_1, \pi_2, \ldots, \pi_n. \varphi_1\, \mathcal{R}_{\bowtie\gamma}\, \varphi_2$.

We pose the following research questions.

RQ1: Is HyPTCTLchecker efficient for practical properties?

RQ2: How many path variables can HyPTCTLchecker handle at most?

We conducted all the experiments on an AWS EC2 m7i.4xlarge instance (with 16vCPU and 64 GiB RAM) that runs Ubuntu 22.04 LTS. We set 6 h as the timeout.

A. Benchmarks

Table I summarizes the benchmarks we used and the experimental results. The translation time is negligible (typically $<$ 0.05 s) and is not integrated in Table I. We used five classes of properties: 1) Deviation; 2) Opacity; 3) Unfair; 4) RobOND; and 5) EF_i. Deviation, Opacity, Unfair, and RobOND are the properties shown in Examples 2, 3, 5, and 6, respectively. EF_i is an artificial property to evaluate the scalability of our semi-algorithm with respect to the number of path variables. Concretely, EF_i is $\exists\pi_1, \pi_2, \ldots, \pi_i. \Diamond_{[p,p]} \bigwedge_{j \in \{1,2,\ldots,i-1\}} \mathrm{COUNT}(\mathrm{a}_{\pi_j}) - \mathrm{COUNT}(\mathrm{a}_{\pi_{j+1}}) = 1$.

[5]HyPTCTLchecker is publicly available at https://github.com/MasWag/HyPTCTLChecker in an open-source manner with all the data to reproduce the experiments.

TABLE I
Summary of the Benchmarks and the Runtime of IMITATOR. Columns $|L|$ and $|\mathbb{C}|$ Show the Number of Locations and Clocks in the PTAs. Columns $|\mathbb{P}|_\psi$ and $|\mathbb{P}|_\mathcal{A}$ Show the Number of Parameters Used in the Properties and the PTAs. Column $|\mathcal{V}|$ Shows the Number of the Quantified Path Variables in ψ. "T.O." Denotes No Termination Within 6 h

Prop. (ψ)	PTA ($\mathcal{A}$)	$\lvert L\rvert$	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{P}\rvert_\psi$	$\lvert\mathbb{P}\rvert_\mathcal{A}$	$\lvert\mathcal{V}\rvert$	Time [sec.]
Deviation	ClkGen	2	1	1	1	2	4.116
Opacity	Coffee	6	2	0	3	2	0.723
Opacity	STAC1:n	8	2	0	2	2	0.178
Opacity	STAC4:n	9	2	0	5	2	$<$ 0.001
Unfair	FIFO	63	2	0	4	2	71.955
Unfair	Priority	72	2	0	4	2	6.855
Unfair	R.R.	81	3	0	4	2	12550.979
RobOND	Coffee	6	2	1	3	2	3.182
RobOND	WFAS_0^1	24	4	1	0	2	1.665
RobOND	WFAS_0^2	24	4	1	0	2	2.570
RobOND	WFAS_1	24	4	1	1	2	67.644
RobOND	WFAS_2	24	4	1	2	2	1332.310
RobOND	ATM	7	2	1	0	2	**T.O.**
RobOND	ATM′	5	2	1	0	2	4179.197
EF_2	Coffee	6	2	1	0	2	0.034
EF_3	Coffee	6	2	1	0	3	159.541
EF_4	Coffee	6	2	1	0	4	**T.O.**

ClkGen is the PTA in Fig. 2. Coffee (a toy coffee machine), STAC1:n and STAC4:n (two Java programs without timing leaks, translated to PTAs) are based on the PTAs in [19]. FIFO, Priority, and R.R. are our original PTAs modeling FIFO, Fixed-Priority, and Round-Robin schedulers, respectively. WFAS_i is a wireless fire alarm system taken from [18], where i shows the number of parameters. WFAS_0^1 and WFAS_0^2 are instances of WFAS with different parameter valuations. ATM is a simple PTA model of an ATM from [20, Fig. 1], and ATM′ is its variant without the branch "check." The PTAs taken from the literature are modified to align with our encoding and evaluation, e.g., by adding the locations and the edges to encode the input and output propositions with labels on the locations and by instantiating some parameters to evaluate the scalability.

B. RQ1: Performance on Practical Properties

In Table I, we observe that for most of the benchmarks, the runtime of IMITATOR is less than a few minutes. Particularly, the runtime for Opacity is always less than 1 s. This aligns with the efficiency of opacity verification with a similar reduction in [19]. For Unfair and RobOND, the runtime largely depends on the complexity of the PTA. For instance, R.R. has more locations and clocks than FIFO and Priority for preemptive scheduling, which blows up the result of the self-composition in Section V-A and increases the runtime of IMITATOR. Similarly, having more parameters (in WFAS_i) or locations (in ATM) increases the runtime. Nevertheless, HyPTCTLchecker can still handle the benchmarks with the parameters in properties or PTAs if the PTAs are of mild complexity. Thus, we answer RQ1 as follows.

> *Answer to RQ1:* HyPTCTLchecker can efficiently handle practical properties for mild size of PTAs, i.e., with roughly up to 4 clocks, and against formulas with up to 3 path variables.

We failed to verify ATM within 6 h although it has smaller $|L|$, $|\mathbb{C}|$, and $|\mathbb{P}|_{\mathcal{A}}$ than WFAS_2. It is difficult to discuss its detail, but this can be partly due to the structure of the PTAs. ATM has two loops whereas ATM′ has only one of them. After the self-composition, they have four and two loops, respectively. It is possible that this caused a combinatorial explosion of the search space.

C. RQ2: Scalability to the Number of Path Variables

In Table I, we observe that HyPTCTLchecker can handle EF_3 but not EF_4. This is because the self-composition in Section V-A exponentially blows up the PTAs with respect to the number of path variables. Given the simplicity of Coffee and EF_3, we answer RQ2 as follows.

> *Answer to RQ2:* HyPTCTLchecker can handle at most three path variables in a reasonable time.

Although the above answer might seem quite restrictive, we remark that the three path variables are likely enough to capture most of the interesting properties. For example, all the HyperLTL or HyperCTL* formulas in the case studies in [21] and [22] are with at most two path variables (potentially with the nested temporal operators, which is out of the scope of our semi-algorithm).

VIII. Conclusion

We introduced HyperPTCTL as the first extension to hyperlogics of parametric timed CTL, enabling reasoning simultaneously on different execution traces. After giving a syntax and semantics for the general logics, we restricted ourselves to a nest-free fragment, extended with COUNT and LAST constructs, allowing for reasoning about the number of actions and the duration from their final occurrence, respectively. To our knowledge, this logic is the first of its kind to reason about parametric timed hyperproperties. Model checking this logic Nest-Free Ext-HyperPTCTL reduces to the model checking PTCTL. While this is, in general, undecidable, we exhibited decidable subclasses. In addition, our implementation within HyPTCTLchecker (built on the top of IMITATOR) goes beyond the decidable fragment, and showed good results, both for the nonparametric and parametric case.

Future works include exhibiting further decidable subclasses, perhaps forbidding equality ("$= p$") in the formula, as in [23], or with restrictions in the formula, such as in [9]. Some undecidability results are not tight, i.e., the exact border between decidability and undecidability remains blurred. Devising and implementing a semi-algorithm for the full HyperPTCTL, beyond the nest-free fragment, is also on our agenda. A comparison of the expressive power between Ext-HyperPTCTL and the other hyperlogics is also a possible future direction.

Finally, optimizing IMITATOR in order to address HyperPTCTL will be an interesting challenge. The blowup due to the self-composition may be addressed using the partial order (e.g., [24]) or symmetry reductions.

References

[1] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in *Proc. STOC*, 1993, pp. 592–601.

[2] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.

[3] L. Aceto, P. Bouyer, A. Burgueño, and K. G. Larsen, "The power of reachability testing for timed automata," *Theor. Comput. Sci.*, vol. 300, nos. 1–3, pp. 411–475, 2003.

[4] É. André, "IMITATOR 3: Synthesis of timing parameters beyond decidability," in *Proc. CAV*, 2021, pp. 1–14.

[5] A. Jovanović, D. Lime, and O. H. Roux, "Integer parameter synthesis for real-time systems," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 445–461, May 2015.

[6] S. Göller and M. Hilaire, "Reachability in two-parametric timed automata with one parameter is EXPSPACE-complete," in *Proc. STACS*, 2021, pp. 36:1–36:18.

[7] V. Bruyère, E. Dall'Olio, and J.-F. Raskin, "Durations and parametric model-checking in timed automata," *ACM Trans. Comput. Logic*, vol. 9, no. 2, pp. 1–23, 2008.

[8] É. André, D. Lime, and M. Ramparison, "TCTL model checking lower/upper-bound parametric timed automata without invariants," in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, 2018, pp. 1–17.

[9] L. Bozzelli and S. La Torre, "Decision problems for lower/upper bound parametric timed automata," *Formal Methods Syst. Design*, vol. 35, no. 2, pp. 121–151, 2009.

[10] T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager, "Linear parametric model checking of timed automata," *J. Logic Algebraic Program.*, vols. 52–53, pp. 183–220, Aug. 2002.

[11] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *Proc. Int. Conf. Princ. Secur. Trust*, 2014, pp. 265–284.

[12] H. Ho, R. Zhou, and T. M. Jones, "Timed hyperproperties," *Inf. Comput.*, vol. 280, Oct. 2021, Art. no. 104639.

[13] E. Bartocci, C. Mateis, E. Nesterini, and D. Nickovic, "Mining hyperproperties using temporal logics," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 5, pp. 1–26, 2023.

[14] B. Bonakdarpour, P. Prabhakar, and C. Sánchez, "Model checking timed hyperproperties in discrete-time systems," in *Proc. NASA Formal Methods Symp.*, 2020, pp. 311–328.

[15] J. S. Miller, "Decidability and complexity results for timed automata and semi-linear hybrid automata," in *Proc. HSCC*, 2000, pp. 296–309.

[16] É. André, E. Lefaucheux, D. Lime, D. Marinho, and J. Sun, "Configuring timing parameters to ensure execution-time opacity in timed automata," in *Proc. TiCSA*, 2023, pp. 1–26.

[17] F. Cassez, "The dark side of timed opacity," in *Proc. ICSA, Part II*, 2009, pp. 21–30.

[18] N. Beneš, P. Bezděk, K. G. Larsen, and J. Srba, "Language emptiness of continuous-time parametric timed automata," in *Proc. ICALP*, 2015, pp. 69–81.

[19] É. André, D. Lime, D. Marinho, and J. Sun, "Guaranteeing timed opacity using parametric timed model checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 4, pp. 1–36, 2022.

[20] L. Dai, T. Chen, Z. Liu, B. Xia, N. Zhan, and K. G. Larsen, "Parameter synthesis problems for one parametric clock timed automata," 2018, *arXiv:1809.07177*.

[21] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL*," in *Proc. CAV, Part I*, 2015, pp. 30–48.

[22] R. Beutner and B. Finkbeiner, "AutoHyper: Explicit-state model checking for HyperLTL," in *Proc. Int. Conf. TACAS, Part I*, 2023, pp. 145–163.

[23] V. Bruyère and J.-F. Raskin, "Real-time model-checking: Parameters everywhere," *Logical Methods Comput. Sci.*, vol. 3, no. 1, pp. 1–30, 2007.

[24] É. André, T. Chatain, and C. Rodríguez, "Preserving partial order runs in parametric time petri nets," *Trans. Embed. Comput. Syst.*, vol. 16, no. 2, pp. 1–26, Mar. 2017.