# Statistical Model Checking for Hyperproperties

Yu Wang*, Siddhartha Nalluri†, Borzoo Bonakdarpour‡, and Miroslav Pajic*
*Department of Electrical & Computer Engineering, Duke University, USA
Email: {yu.wang094, miroslav.pajic}@duke.edu
†Department of Computer Science, Duke University, USA
Email: siddhartha.nalluri@duke.edu
‡Department of Computer Science and Engineering, Michigan State University, USA
Email: borzoo@msu.edu

***Abstract*—*Hyperproperties* have shown to be a powerful tool for expressing and reasoning about information-flow security policies. In this paper, we investigate the problem of *statistical model checking* (SMC) for hyperproperties. Unlike exhaustive model checking, SMC works based on drawing samples from the system at hand and evaluate the specification with statistical confidence. The main benefit of applying SMC over exhaustive techniques is its efficiency and scalability. To reason about probabilistic hyperproperties, we first propose the temporal logic HyperPCTL* that extends PCTL* and HyperPCTL. We show that HyperPCTL* can express important probabilistic information-flow security policies that cannot be expressed with HyperPCTL. Then, we introduce SMC algorithms for verifying HyperPCTL* formulas on discrete-time Markov chains, based on sequential probability ratio tests (SPRT) with a new notion of multi-dimensional indifference region. Our SMC algorithms can handle both non-nested and nested probability operators for any desired significance level. To show the effectiveness of our technique, we evaluate our SMC algorithms on four case studies focused on information security: timing side-channel vulnerability in encryption, probabilistic anonymity in dining cryptographers, probabilistic noninterference of parallel programs, and the performance of a randomized cache replacement policy that acts as a countermeasure against cache flush attacks.**

## I. Introduction

*Randomization* has been a powerful tool in the design and development of many algorithms and protocols that make probabilistic guarantees in the area of information security. Prominent examples such as *quantitative information flow* [38], [46], *probabilistic noninterference* [35], and *differential privacy* [23] quantify the amount of information leakage and the relation between two probabilistic execution traces of a system. These and similar requirements constitute *probabilistic hyperproperties* [5], [17]. They extend traditional trace properties from sets of execution traces to sets of sets of execution traces and allow for explicit and simultaneous quantification over the temporal behavior of multiple execution traces. Probabilistic hyperproperties stipulate the probability relation between independent executions.

*Model checking*, an automated technique that verifies the correctness of a system with respect to a formal specification, has arguably been the most successful story of using formal methods in the past three decades. Since many systems have stochastic nature (e.g., randomized distributed algorithms), model checking of such systems has been an active area of research. Temporal logics such as PCTL* [9] as well as model checkers PRISM [39] and STORM [21] have been developed as formalism and tools to express and reason about probabilistic systems. However, these techniques are unable to capture and verify probabilistic hyperproperties that are vital to reason about quantified information-flow security.

The state of the art in specification and verification of probabilistic hyperproperties is limited to the temporal logic HyperPCTL [5]. The model checking algorithm for HyperPCTL utilizes a numerical approach that iteratively computes the exact measure of paths satisfying relevant sub-formulas. In this context, we currently face two significant and orthogonal gaps to apply verification of probabilistic hyperproperties in practice:

- **Expressiveness.** First, HyperPCTL does *not* allow (1) nesting of temporal operators, which is necessary to express requirements such as performance guarantees in randomized cache replacement protocols that defend against cache-flush attacks, and (2) explicit quantification over execution paths, which is necessary to reason about the probability of reaching certain states.
- **Scalability.** Second, and perhaps more importantly, numerical algorithms for probabilistic model checking, including the one proposed in [5], tend to require substantial time and space, and often run into serious scalability issues. Indeed, these algorithms work only for small systems that have certain structural properties. On top of this difficulty, another major challenge in verifying hyperproperties is that the computation complexity for exhaustive verification grows at least exponentially in the number of quantifiers of the input formula [5], [7], [12], [18].

In this paper, our goal is to address the above stumbling blocks (expressiveness and scalability) by investigating *statistical model checking* (SMC) [6], [40], [41] for hyperproperties with probabilistic guarantees. To this end, we first introduce on discrete-time Markov chains the temporal logic HyperPCTL* that extends PCTL* [9] by (i) allowing explicit quantification over paths, and HyperPCTL [5] by (ii) allowing nested probability and temporal operators. These two features are crucial in expressing probabilistic hyperproperties, such as probabilistic noninterference. Specifically, consider a probabilistic program with a high-security input $h \in \{0,1\}$ and a low-security output $l \in \{0,1\}$. Probabilistic noninterference requires that the probability of observing the low-security output $l = 0$ (or $l = 1$) should be equal for two executions $\pi_{h=0}$ and $\pi_{h=1}$ that

have the high-security input $h = 0$ and $h = 1$, respectively. In other words, the high-security input cannot be inferred from the low-security output through a probabilistic channel – i.e.,

$$\mathbb{P}^{\pi_{h=0}}(\pi_{h=0} \text{ outputs } l = 0) = \mathbb{P}^{\pi_{h=1}}(\pi_{h=1} \text{ outputs } l = 0)$$

This property involves the relation between two executions $\pi_{h=0}$ and $\pi_{h=1}$, and cannot be expressed by non-hyper logics, such as PCTL$^*$. We also illustrate that HyperPCTL$^*$ can elegantly express properties such as generalized probabilistic causation, countermeasures for side-channel attacks, probabilistic noninterference, and probabilistic independence among executions. In addition, the latter is an important performance property for cache replacement policies that defend against cache flush attacks and cannot be expressed in HyperPCTL, as it requires using nested temporal operators.

To tackle the scalability problem, we turn to SMC – a popular approach in dealing with probabilistic systems that uses a *sample-based* technique, where one asserts whether the system satisfies a property by observing some of its executions [41], [42], [51]–[53]. The general idea of SMC is to treat the problem of checking a temporal logic formula on a probabilistic system as *hypothesis testing* [6], [45]. By drawing samples from the underlying probabilistic system, the satisfaction of the formula can be inferred with high confidence levels. To the best of our knowledge, the work on SMC for hyperproperties is limited to [50], where the authors propose an SMC algorithm for hyperproperties for cyber-physical systems using the *Clopper-Pearson* (CP) confidence intervals. In this work, we propose another SMC algorithm for hyperproperties using *sequential probability ratio tests* (SPRT) [49], which are more efficient for statistical inference than using the confidence intervals.

Developing SMC for HyperPCTL$^*$ formulas using SPRT has significant challenges that do not appear in SMC for non-hyper probabilistic temporal logics, such as PCTL$^*$. This is caused by the fact that in HyperPCTL$^*$, one can express complex probabilistic quantification among different paths. Specifically, HyperPCTL$^*$ allows for:

- **Probabilistic quantification of multiple paths.** For example, formula

$$\mathbb{P}^{(\pi_1,\pi_2)}\left(\mathsf{a}^{\pi_1} \,\mathcal{U}\, \mathsf{a}^{\pi_2}\right) > p \tag{1}$$

  means that the probability that atomic proposition $\mathsf{a}$ holds on a random path $\pi_1$ until it becomes true on another random path $\pi_2$ is greater than some $p \in [0, 1]$.

- **Arithmetics of probabilistic quantification.** For example, formula

$$\mathbb{P}^{\pi_1}\left(\Diamond \mathsf{a}^{\pi_1}\right) + \mathbb{P}^{\pi_2}(\Box \mathsf{a}^{\pi_2}) > p \tag{2}$$

  stipulates that the sum of the probability that $\mathsf{a}$ finally holds and the probability that $\mathsf{a}$ always holds, is greater than some $p \geq 0$.

- **Nested probabilistic quantification.** This is different from nested probabilistic quantification in PCTL$^*$. For example, formula

$$\mathbb{P}^{\pi_1}\left(\mathbb{P}^{\pi_2}\left(\mathsf{a}^{\pi_1} \,\mathcal{U}\, \mathsf{a}^{\pi_2}\right) > p_1\right) > p_2, \tag{3}$$

  requires that for a (given) path $\pi_1$, the probability that $(\mathsf{a}^{\pi_1} \,\mathcal{U}\, \mathsf{a}^{\pi_2})$ holds for a random path $\pi_2$, is greater than some $p_1 \in [0, 1]$; and, this fact should hold with probability greater than some $p_2 \in [0, 1]$ for a random path $\pi_1$.

The different kinds of complex probabilistic quantification among multiple paths cannot be handled by existing SMC algorithms for non-hyper probabilistic temporal logics [6].

To use SPRT to handle the aforementioned challenges, SMC requires a condition on the *indifference regions*. As a simple example, to statistically infer if $\mathbf{Pr}(A) > p$, for some random event $A$, using SPRT from sampling, it is required that the probability $\mathbf{Pr}(A)$ should not be too "close" to $p$; this means that there exists some known $\varepsilon > 0$ such that $\mathbf{Pr}(A) \notin (p - \varepsilon, p + \varepsilon)$, i.e., $\mathbf{Pr}(A) \geq p + \varepsilon$ or $\mathbf{Pr}(A) \leq p - \varepsilon$. This is a common assumption used for many SMC techniques [6], [45]. Therefore, it is sufficient to test between the two most indistinguishable cases $\mathbf{Pr}(A) \notin p - \varepsilon$ and $\mathbf{Pr}(A) \notin p + \varepsilon$. The interval $(p - \varepsilon, p + \varepsilon)$ is usually referred to as the *indifference region*. In this work, we propose new conditions on the *indifference regions* that enable the use of SPRT in the SMC of HyperPCTL$^*$.

For the SMC of arithmetics of probabilistic quantifications in (2), we consider the hypothesis testing problem:

$$\begin{aligned} H_0 &: \left(\mathbb{P}^{\pi_1}\left(\Diamond \mathsf{a}^{\pi_1}\right), \mathbb{P}^{\pi_2}(\Box \mathsf{a}^{\pi_2})\right) \in D, \\ H_1 &: \left(\mathbb{P}^{\pi_1}\left(\Diamond \mathsf{a}^{\pi_1}\right), \mathbb{P}^{\pi_2}(\Box \mathsf{a}^{\pi_2})\right) \in D^{\mathrm{c}}, \end{aligned} \tag{4}$$

where $D = \{(p_1, p_2) \in [0, 1]^2 \,|\, p_1 + p_2 > p\}$ and $D^{\mathrm{c}}$ is its complement set. To handle the joint probability $\left(\mathbb{P}^{\pi_1}\left(\Diamond \mathsf{a}^{\pi_1}\right), \mathbb{P}^{\pi_2}(\Box \mathsf{a}^{\pi_2})\right)$ in (4), we propose a novel *multi-dimensional* extension of the standard SPRT. Specifically, we first generalize the notion of the indifference region (namely, the parameter $\varepsilon$) to a multi-dimensional case. This new notion of indifference region ensures that our multi-dimensional SPRT algorithm provides provable probabilistic guarantees for any desired false positive $\alpha_{\mathrm{FP}} \in (0, 1)$ and false negative $\alpha_{\mathrm{FN}} \in (0, 1)$ ratios. Then, we note that the hypotheses $H_0$ and $H_1$ in (4) are composite, which contains infinitely many simple hypotheses. To use SPRT, which mainly deal with simple hypotheses, on the two composite hypotheses, we propose a geometric condition to identify the two most indistinguishable simple hypotheses from $H_0$ and $H_1$, respectively. We show that if SPRT can distinguish these two simple hypotheses, then any two simple hypotheses from $H_0$ and $H_1$ can be distinguished by the same test.

For the SMC of probabilistic quantification of multiple paths in (1), we note that the SMC of probabilistic quantification of multiple parallel paths can be handled by generalizing the common SPRT to tuples of samples. For the SMC of nested probabilistic quantification in (3), we can perform a compositional analysis for the probabilistic error in the SMC of the sub-formulas, to yield the global false positive and false negative ratios, in the same way as [50].

Finally, based on the above new statistical inference algorithms, we design SMC algorithms for HyperPCTL$^*$. These algorithms are fully implemented and evaluated by four prominent case studies.[1] Specifically, we apply our SMC algorithms to analyze: (i) the time side-channel vulnerability

[1]The simulation code is available at [20].

in encryption [1], [16], [48], (ii) probabilistic anonymity in dining cryptographers [15], (iii) probabilistic noninterference of parallel programs [33], and (iv) the performance of a random cache replacement policy [14] that defends against cache flush attacks. Our results show that the proposed SMC algorithms provide the correct answer with high confidence levels in all cases while requiring very short analysis times.

*Organization:* The rest of the paper is organized as follows. We introduce HyperPCTL$^*$ in Section II. The expressiveness of HyperPCTL$^*$ is discussed in Section IV, before illustrating its application in Section III. Our SMC algorithms for HyperPCTL$^*$ are introduced in Section V. We present our case studies and experimental results in Section VI. Related work is discussed in Section VII, before concluding remarks in Section VIII.

## II. The Temporal Logic HyperPCTL$^*$

We begin with some notation. We denote the set of natural and real numbers by $\mathbb{N}$ and $\mathbb{R}$, respectively. Let $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$. For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. The cardinality of a set is denoted by $|\cdot|$. For $n \in \mathbb{N}$, we use $\underline{s} = (s_1, \dots, s_n)$ to denote a *tuple*. We use $S = s(0)s(1)\dots$ to denote a *sequence*, and the $i$-suffix of the sequence is denoted by $S^{(i)} = s(i)s(i+1)\cdots$. For any set $D \subseteq \mathbb{R}^n$, we denote its *boundary*, *interior*, *closure* and *complement* by $\partial D$, $D^\circ$, $\overline{D}$, and $D^c$, respectively.

Our proposed temporal logic HyperPCTL$^*$ is an extension of PCTL$^*$ [9] that enables handling hyperproperties. It also can be viewed as a variation of HyperPCTL [5] that allows for nested temporal and probability operators. In this section, we introduce the formal syntax and semantics of HyperPCTL$^*$; its relation with PCTL$^*$, HyperLTL and HyperPCTL is discussed in the next section.

### A. Syntax

HyperPCTL$^*$ formulas are defined by the grammar

$$\varphi ::= \mathsf{a}^\pi \mid \varphi^\pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U}^{\leq k}\, \varphi \mid \rho \bowtie \rho \quad (5)$$

$$\rho ::= f(\rho, \dots, \rho) \mid \mathbb{P}^{\underline{\pi}}(\varphi) \mid \mathbb{P}^{\underline{\pi}}(\rho) \quad (6)$$

where

- $\mathsf{a} \in \mathsf{AP}$ is an atomic proposition;
- $\pi$ is a (fresh) *random path variable* from an infinite supply of such variables $\Pi$;[2]
- $\bigcirc$ and $\mathcal{U}^{\leq k}$ are the 'next' and 'until' operators, respectively, where $k \in \mathbb{N}_\infty$ is the time bound and $\mathcal{U}^{\leq\infty}$ means "*unbounded until*";
- $\bowtie \in \{<, >, =, \leq, \geq\}$, which allows comparing probabilities among different random paths;
- $\mathbb{P}^{\underline{\pi}}$ is the probability operator for a tuple of random path variables $\underline{\pi} = (\pi_1, \dots, \pi_n)$ for some $n \in \mathbb{N}$, and
- $f : \mathbb{R}^n \to \mathbb{R}$ is an $n$-ary elementary function,[3] with constants being viewed as a 0-ary function. This enables expressing arithmetic operations and entropy from probabilities.

HyperPCTL$^*$ can be viewed as a probabilistic adaptation of HyperLTL [18]. Following the terminology of HyperLTL (and more generally, the first-order logic [24]), in a given HyperPCTL$^*$ formula, we call a path variable *free* if it has not been associated by a probability operator; otherwise, the path variable is *quantified*. For example, in a HyperPCTL$^*$ formula $\mathbb{P}^{\pi_1}(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_2})$, the path variable $\pi_1$ is quantified and the path variable $\pi_2$ is free. Mostly, we are interested in HyperPCTL$^*$ formulas with all the path variables quantified.

Additional logic operators are derived as usual: $\mathtt{true} \equiv \mathsf{a}^\pi \vee \neg\mathsf{a}^\pi$, $\varphi \vee \varphi' \equiv \neg(\neg\varphi \wedge \neg\varphi')$, $\varphi \Rightarrow \varphi' \equiv \neg\varphi \vee \varphi'$, $\Diamond^{\leq k}\varphi \equiv \mathtt{true}\, \mathcal{U}^{\leq k}\, \varphi$, and $\Box^{\leq k}\varphi \equiv \neg\Diamond^{\leq k}\neg\varphi$. We denote $\mathcal{U}^{\leq\infty}$, $\Diamond^{\leq\infty}$, and $\Box^{\leq\infty}$ by $\mathcal{U}$, $\Diamond$, and $\Box$, respectively. We represent a 1-tuple by its element, i.e., $\sigma^{(\pi)}$ and $\mathbb{P}^{(\pi)}$ are written as $\sigma^\pi$ and $\mathbb{P}^\pi$, respectively.

### B. Semantics

We consider the semantics of HyperPCTL$^*$ on discrete-time Markov chains (DTMCs) with their states labeled by a set of atomic propositions AP. Formally, a DTMC is a tuple $\mathcal{M} = (\mathcal{S}, s_{\text{init}}, \mathbf{T}, \mathsf{AP}, L)$ where

- $\mathcal{S}$ is the finite set of *states*, and $s_{\text{init}}$ the *initial state*;
- $\mathbf{T} : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is the *transition probability function*, where for any state $s \in \mathcal{S}$, it holds that
$$\sum_{s' \in \mathcal{S}} \mathbf{T}(s, s') = 1;$$
- AP is the set of *atomic propositions*, and
- $L : \mathcal{S} \to 2^{\mathsf{AP}}$ is a *labeling function*.

An example DTMC labeled by the atomic propositions $\{\mathsf{a}_1, \mathsf{a}_2\}$ is illustrated in Figure 2. A *path* of a DTMC $\mathcal{M} = (\mathcal{S}, s_{\text{init}}, \mathbf{T}, \mathsf{AP}, L)$ is of the form $S = s(0)s(1)\cdots$, such that for every $i \in \mathbb{N}$, $s(i) \in \mathcal{S}$ and $\mathbf{T}(s(i), s(i+1)) \neq 0$. By $\mathsf{Paths}(s)$, we denote the set of paths that start from state $s$, while $\mathsf{Paths}(\mathcal{M})$ denotes the set of all paths of DTMC $\mathcal{M}$.

The semantics of HyperPCTL$^*$ formulas is described in terms of the interpretation tuple $(\mathcal{M}, V)$, where

- $\mathcal{M} = (\mathcal{S}, s_{\text{init}}, \mathbf{T}, \mathsf{AP}, L)$ is a DTMC, and
- $V : \Pi \to \mathsf{Paths}(\mathcal{M})$ is a *path assignment*, mapping each (random) path variable to a concrete path of $\mathcal{M}$, starting from the initial state $s_{\text{init}}$ by default.

We denote by $[\![\cdot]\!]_V$ the instantiation of the assignments $V$ on a HyperPCTL$^*$ formula. The judgment rules for semantics of a HyperPCTL$^*$ formula $\varphi$ are detailed in Figure 1, where

1) $V[\cdot]$ denotes the revision of the assignment $V$ by the rules given in $[\cdot]$.
2) $V^{(i)}$ is the $i$-shift of path assignment $V$, defined by $V^{(i)}(\pi) = (V(\pi))^{(i)}$.
3) By the second rule, associating the path variable $\pi$ to the formula $\varphi$ assigns the value of all path variables in $\varphi$ to $V(\pi)$. For a given $(\mathcal{M}, V)$, the satisfaction of $\varphi$ is preserved, if the free path variables in $\varphi$

[2] Technically, using a non-fresh path variable can be allowed. However, to avoid possible confusion about the meaning of the HyperPCTL$^*$ formulas, we only use fresh path variables here.

[3] Elementary functions are defined as a sum, product, and/or composition of finitely many polynomials, rational functions, trigonometric and exponential functions, and their inverse functions.

$$
\begin{array}{lll}
(\mathcal{M},V) \models \mathsf{a}^{\pi} & \text{iff} & \mathsf{a} \in L\big(V(\pi)(0)\big) \\
(\mathcal{M},V) \models \varphi^{\pi} & \text{iff} & (\mathcal{M}, V[\pi' \mapsto V(\pi)]) \models \varphi \\
(\mathcal{M},V) \models \neg\varphi & \text{iff} & (\mathcal{M},V) \not\models \varphi \\
(\mathcal{M},V) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (\mathcal{M},V) \models \varphi_1 \text{ and } (\mathcal{M},V) \models \varphi_2 \\
(\mathcal{M},V) \models \bigcirc\varphi & \text{iff} & (\mathcal{M},V^{(1)}) \models \varphi \\
(\mathcal{M},V) \models \varphi_1\, \mathcal{U}^{\leq k}\, \varphi_2 & \text{iff} & \text{there exists } i \leq k \text{ such that } \big((\mathcal{M},V^{(i)}) \models \varphi_2\big) \wedge \big(\text{for all } j < i.\ (\mathcal{M},V^{(j)}) \models \varphi_1\big) \\
(\mathcal{M},V) \models \rho \bowtie \rho & \text{iff} & (\mathcal{M},V) \models [\![\rho]\!]_V \bowtie [\![\rho]\!]_V \\
[\![f(\rho,\dots,\rho)]\!]_V & = & f\big([\![\rho]\!]_V,\dots,[\![\rho]\!]_V\big) \\
[\![\mathbb{P}^{(\pi_1,\dots,\pi_n)}(\varphi)]\!]_V & = & \mathbf{Pr}\Big\{\big(S_i \in \mathsf{Paths}(V(\pi_i)(0))\big)_{i\in[n]} : (\mathcal{M}, V[\pi_i \mapsto S_i \text{ for all } i \in [n]]) \models \varphi\Big\} \\
[\![\mathbb{P}^{(\pi_1,\dots,\pi_n)}(\rho)]\!]_V & = & \mathbf{Pr}\Big\{\big(S_i \in \mathsf{Paths}(V(\pi_i)(0))\big)_{i\in[n]} : (\mathcal{M}, V[\pi_i \mapsto S_i \text{ for all } i \in [n]]) \models \rho\Big\}
\end{array}
$$

Fig. 1. Semantics of HyperPCTL$^*$.

are replaced by $\pi$. (The quantified path variables are unaffected, as discussed in Point 4) below.)
For example, the following formulas are *semantically* equivalent – i.e., the truth value of the formulas on both sides are identical for any given $(\mathcal{M},V)$,

$$
\begin{aligned}
(\mathsf{a}^{\pi_1})^{\pi_2} &\equiv \mathsf{a}^{\pi_2}, \\
(\bigcirc \mathsf{a}^{\pi_1})^{\pi_2} &\equiv \bigcirc \mathsf{a}^{\pi_2}, \\
(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_2})^{\pi_3} &\equiv \mathsf{a}^{\pi_3}\, \mathcal{U}\, \mathsf{a}^{\pi_3}.
\end{aligned}
$$

In particular, in the first above equivalence, $\pi_2$ in $(\mathsf{a}^{\pi_1})^{\pi_2}$ can replace $\pi_1$, since $\pi_1$ is a free random path variable and obtain $\mathsf{a}^{\pi_2}$. However, these two formulas would not be equivalent if $\pi_1$ was not free.

4) In the last two rules, the probability $\mathbf{Pr}$ is taken for an $n$-tuple of sample paths $(S_1,\dots,S_n)$ to instantiate $\underline{\pi}$, and ":" means 'such that'. The evaluation of the probability operator $[\![\mathbb{P}^{(\pi_1,\dots,\pi_n)}(\varphi)]\!]_V$ means to (re)draw the random path variables $\pi_1,\dots,\pi_n$ from their current initial states on the DTMC $\mathcal{M}$ (regardless of their current assignment by $V$), and evaluate the satisfaction probability of $\varphi$. Thus, following Point 3), the quantified path variables are unaffected by the association of new path variables and we have the following *semantic* equivalence:

$$
\begin{aligned}
\big(\mathbb{P}^{\pi_1}(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_2})\big)^{\pi_3} &\equiv \mathbb{P}^{\pi_1}(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_3}), \\
\big(\mathbb{P}^{(\pi_1,\pi_2)}(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_2})\big)^{\pi_3} &\equiv \mathbb{P}^{(\pi_1,\pi_2)}(\mathsf{a}^{\pi_1}\, \mathcal{U}\, \mathsf{a}^{\pi_2}).
\end{aligned}
$$

In particular, in the first equivalence, $\pi_3$ in $(\mathbb{P}^{\pi_1}(\mathsf{a}^{\pi_1}\,\mathcal{U}\,\mathsf{a}^{\pi_2}))^{\pi_3}$ can replace $\pi_2$, since $\pi_2$ is free, obtaining $\mathbb{P}^{\pi_1}(\mathsf{a}^{\pi_1}\,\mathcal{U}\,\mathsf{a}^{\pi_3})$. However, $\pi_3$ cannot replace $\pi_1$, as $\pi_1$ is quantified by the probability operator.

### C. Discussion on HyperPCTL$^*$

Consider the DTMC $\mathcal{M}$ in Figure 2, and the following HyperPCTL$^*$ formula:

$$
\varphi = \mathbb{P}^{(\pi_1,\pi_2)}\Big((\mathsf{a}_1^{\pi_1} \wedge \mathsf{a}_1^{\pi_2}) \wedge \Diamond(\mathsf{a}_2^{\pi_1} \wedge \mathsf{a}_2^{\pi_2})\Big) > 1/6.
$$

The formula claims that two (independently) random paths $\pi_1$ and $\pi_2$ from $s_{\text{init}} = s_0$ satisfy $(\mathsf{a}_1^{\pi_1} \wedge \mathsf{a}_1^{\pi_2}) \wedge \Diamond(\mathsf{a}_2^{\pi_1} \wedge \mathsf{a}_2^{\pi_2})$, i.e., both paths should satisfy $\mathsf{a}_1$ in their initial state and satisfy $\mathsf{a}_2$ (later) at the same time with probability greater than $1/6$. By calculation from Figure 2, this probability is $1/4$, so we have $\mathcal{M} \models \varphi$.

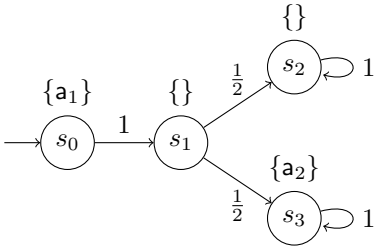


Fig. 2. HyperPCTL$^*$ example on DTMC $\mathcal{M}$.

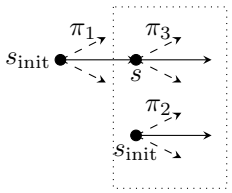


Fig. 3. Computation trees for (7). The dashed arrows show other possible sample values of the path variables to illustrate the probabilistic computation tree.

HyperPCTL$^*$ can generate complex nested formulas. We explain this using two formulas. First consider the formula:

$$
\mathbb{P}^{\pi_1}\Big(\Diamond\big(\mathbb{P}^{(\pi_2,\pi_3)}\big(\mathsf{a}^{\pi_2}\,\mathcal{U}\,(\mathsf{a}^{\pi_3})^{\pi_1}\big) > c_2\big)\Big) > c_1. \tag{7}
$$

The formula (7) states that with probability greater than $c_1$, we can find a path $\pi_1$, such that finally from some state $s$ on $\pi_1$, with probability greater than $c_2$, we can find a pair of paths $(\pi_2,\pi_3)$ from the pair of states $(s_{\text{init}}, s)$ to satisfy "$\mathsf{a}^{\pi_2}$ until $\mathsf{a}^{\pi_3}$". That is, the computation tree of $\pi_3$ is a subtree of the computation tree of $\pi_1$ (rooted at $s_{\text{init}}$), since $\pi_3$ in $(\mathsf{a}^{\pi_3})^{\pi_1}$ is in the scope of $\pi_1$. On the other hand, since $\pi_2$ is indexed by $\pi_1$, its computation tree is rooted at $s_{\text{init}}$ (see Figure 3). The inner subformula $\mathbb{P}^{(\pi_2,\pi_3)}\big(\mathsf{a}^{\pi_2}\,\mathcal{U}\,(\mathsf{a}^{\pi_3})^{\pi_1}\big) > c_2$ in (7) involves the probabilistic computation trees of $\pi_2$ and $\pi_3$, as shown by the dotted box in Figure 3.

Now, consider the formula:

$$
\mathbb{P}^{\pi_1}\Big(\Diamond\big(\mathbb{P}^{(\pi_2,\pi_3)}\big(\mathsf{a}^{\pi_2}\,\mathcal{U}\,\mathsf{a}^{\pi_3}\big) > c_2\big)^{\pi_1}\Big) > c_1. \tag{8}
$$

It requires that with probability greater than $c_1$, we can find a path $\pi_1$, such that finally from some state $s$ on $\pi_1$, with probability greater than $c_2$, we can find a pair of paths $(\pi_2,\pi_3)$ from the state $s$ that satisfy "$\mathsf{a}^{\pi_2}$ until $\mathsf{a}^{\pi_3}$". That is, the

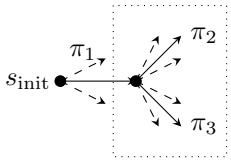


Fig. 4. Computation trees for (8). The dashed arrows show other possible sample values of the path variables to illustrate the probabilistic computation tree.

computation tree of $\pi_2$ and $\pi_3$ (rooted at $s$) is a subtree of the computation tree of $\pi_1$, rooted at $s_{\text{init}}$ (see Figure 4). Again, the inner subformula $\mathbb{P}^{(\pi_2,\pi_3)}\big(\mathsf{a}^{\pi_2}\,\mathcal{U}\,\mathsf{a}^{\pi_3}\big) > c_2$ in (8) involves the probabilistic computation trees of $\pi_2$ and $\pi_3$, as shown by the dotted box in Figure 4.

## III. Applications of HyperPCTL*

In this section, we illustrate the application of HyperPCTL* by four examples related to information-flow security, ranging from timing attacks, scheduling of parallel programs, communication protocols, and computer hardware. These examples cannot be properly handled by existing temporal logics.

### A. Side-channel Vulnerability

Timing side-channel attacks are possible if an attacker can infer the secret values, which are set at the second step of an execution, by observing the execution time of a program. To prevent such attacks, it is required that the probability of termination within some $k \in \mathbb{N}$ steps should be approximately equal for two (random) executions $\pi_1$ and $\pi_2$, where the secret values are $\mathtt{S}_1$ and $\mathtt{S}_2$, respectively:

$$\begin{aligned}&\mathbb{P}^{\pi_1}\left((\bigcirc \mathtt{S}_1^{\pi_1}) \Rightarrow (\Diamond^{\leq k}\,\mathtt{F}^{\pi_1})\right)\\ &\quad\approx_\varepsilon \mathbb{P}^{\pi_2}\big((\bigcirc \mathtt{S}_2^{\pi_2}) \Rightarrow (\Diamond^{\leq k}\,\mathtt{F}^{\pi_2})\big),\end{aligned} \tag{9}$$

where the label $\mathtt{F}$ represents the end of execution, the next operator signifies that the secret is established in the first step of execution, and $\approx_\varepsilon$ stands for approximately equal within some $\varepsilon > 0$. If (9) holds, then an attacker cannot infer the the secret values from whether the program terminates in $k$ steps.

### B. Probabilistic Noninterference

Probabilistic noninterference [35] establishes the connection between information theory and information flow by employing probabilities to address covert channels. Intuitively, it requires that the probability of every low-observable trace pattern is the same for every low-equivalent initial state. For example, consider the parallel composition of the following $n$-threads:

$$\begin{aligned}\mathrm{Th}_k:\ \ &\texttt{for}\ \ i_k = 1\ \ \texttt{to}\ \ (h+1)\times k\\ &\qquad\texttt{do}\ \ \{\ldots; l \leftarrow (k \bmod 2)\},\end{aligned} \tag{10}$$

where $k \in [n]$ and $l \in \{0,1\}$ is a publicly observable output. The secret input $h$ is randomly set to 0 or 1 with probability 0.5. At each step, the processor randomly chooses one thread among the unfinished threads with equal probability and executes *one iteration* of the for-loop (including the assignment of $l$), until all the $n$ threads are finished. Clearly, the (random) execution of this $n$-thread program can be represented by a DTMC, where the states are labeled by the values of all the variables. Starting from the initial state, it sets the value of $h$ at the second step and then executes the threads until finished. The termination states are labeled by $\mathtt{F}$.

As the threads have different numbers of loops depending on $h$ and the scheduling is uniformly random, the whole process is more likely to terminate at a thread with more loops, whose thread number is partially indicated by $l$. This opens up the possibility that by observing $l$, an attacker can infer the difference in the number of loops among the threads, and hence infer $h$. On the other hand, the attack cannot happen if the probability of observing $\mathtt{L}_0 : l = 0$ (or $\mathtt{L}_1 : l = 1$) is approximately equal, regardless of $\mathtt{H}_0 : h = 0$ or $\mathtt{H}_1 : h = 1$ – i.e., the value of $h$ cannot be inferred from the value of $l$. This is formally defined in HyperPCTL* by:

$$\begin{aligned}&\mathbb{P}^{\pi_1}\left((\bigcirc \mathtt{H}_0{}^{\pi_1}) \Rightarrow \left(\Diamond(\mathtt{F}^{\pi_1} \wedge \mathtt{L}_0{}^{\pi_1})\right)\right)\\ &\quad\approx_\varepsilon \mathbb{P}^{\pi_2}\left((\bigcirc \mathtt{H}_1{}^{\pi_2}) \Rightarrow \left(\Diamond(\mathtt{F}^{\pi_2} \wedge \mathtt{L}_0{}^{\pi_2})\right)\right),\end{aligned} \tag{11}$$

and

$$\begin{aligned}&\mathbb{P}^{\pi_1}\left((\bigcirc \mathtt{H}_0{}^{\pi_1}) \Rightarrow \left(\Diamond(\mathtt{F}^{\pi_1} \wedge \mathtt{L}_1{}^{\pi_1})\right)\right)\\ &\quad\approx_\varepsilon \mathbb{P}^{\pi_2}\left((\bigcirc \mathtt{H}_1{}^{\pi_2}) \Rightarrow \left(\Diamond(\mathtt{F}^{\pi_2} \wedge \mathtt{L}_1{}^{\pi_2})\right)\right),\end{aligned} \tag{12}$$

where $\approx_\varepsilon$ stands for approximately equal within $\varepsilon$ and the next operator signifies that the secret is established in the first step of execution. In (11), $\pi_1$ is a random execution of the program, where it sets $h = 0$ at the second step and finally yields $l = 0$ and $\pi_2$ is a random execution of the program, where it sets $h = 1$ at the second step and finally yields $l = 0$; and similarly for (12).

### C. Dining Cryptographers

Several cryptographers sit around a table having dinner. Either one of the cryptographers or, alternatively, the National Security Agency (NSA) must pay for their meal. The cryptographers respect each other's right to make an anonymous payment but want to find out whether the NSA paid. So they decide to execute the following protocol:

- Every two cryptographers establish a shared one-bit secret by tossing an unbiased coin and only informs the cryptographer on the right of the outcome.
- Then, each cryptographer publicly states whether the two coins that it can see (the one it flipped and the one the left-hand neighbor flipped) agree if he/she did not pay.
- However, if a cryptographer actually paid for dinner, then it instead states the opposite – disagree if the coins are the same and agree if the coins are different.
- An even number of agrees indicates that the NSA paid, while an odd number indicates that a cryptographer paid.

The protocol can be modeled by a DTMC with the states labeled by the values of the Boolean variables mentioned below. In addition, the state labels $\mathtt{C}_i$ for $i = 1, 2, 3$ indicate that cryptographer $i$ paid, and $\mathtt{C}_0$ indicates that the NSA paid. The common shared secret between two cryptographers $i$ and $j$ is indicated by the label $\mathtt{S}_{ij}$. The final result of the process is

indicated by a Boolean variable $\mathtt{P}$, where $\mathtt{P}$ if a cryptographer paid, and $\neg\mathtt{P}$ otherwise. We define an information-flow security condition that given that some cryptographer paid, the probability that either cryptographer $i$ or $j$ paid are (approximately) equal irrespective of the common shared secret between them, i.e., the results of the coin tosses. This is specified by the following HyperPCTL$^*$ formula:

$$\mathbb{P}^{\pi_1}\left(\Diamond(\neg \mathtt{S}_{ij}^{\pi_1} \wedge \Diamond \mathtt{P}^{\pi_1})\right) \approx_\varepsilon \mathbb{P}^{\pi_2}\left(\Diamond(\mathtt{S}_{ij}^{\pi_2} \wedge \Diamond \mathtt{P}^{\pi_2})\right) \approx_\varepsilon \mathbb{P}^{\pi_3}\left(\Diamond(\neg \mathtt{S}_{ij}^{\pi_3} \wedge \Diamond \mathtt{P}^{\pi_3})\right) \approx_\varepsilon \mathbb{P}^{\pi_4}\left(\Diamond(\mathtt{S}_{ij}^{\pi_4} \wedge \Diamond \mathtt{P}^{\pi_4})\right). \tag{13}$$

where $\approx_\varepsilon$ stands for approximately equal within $\varepsilon$. In (13), $\pi_1$ is a random execution of the protocol, where the common shared secret between two cryptographers $i$ and $j$ is set to $\mathtt{S}_{ij}$ during the execution and the final return is $\mathtt{P}$ – i.e. some cryptographer paid; and similarly for $\pi_2$, $\pi_3$, and $\pi_4$.

### D. Randomized Cache Replacement Policy

Cache replacement policies decide which *cache lines* are replaced in case of a *cache miss*. Randomized policies employ random replacement as a countermeasure against cache flush attacks. On the negative side, they also introduce performance losses. Following [14], we model a cache as a *Mealy machine* with the access sequence as the input. Each state of the Mealy machine represents a unique configuration of the cache, i.e., the cache lines stored. The transition of the Mealy machine captures a *random replacement policy* that for access to memory data in address $b$, (i) if it is already stored in the cache, return Hit $\mathtt{H}$; (ii) if it is not stored and the cache has free space, return Miss $\mathtt{M}$ and write $b$ in free space, and (iii) if $b$ is not stored and the cache is full, then returns Miss $\mathtt{H}$, and randomly overwrite a line (with uniform distribution) with $b$.

The performance requirement of such a policy is that, from an empty cache, after $N$ steps (when the cache almost fills), in a time window of $T$, the probability of observing $T$ consecutive $\mathtt{H}$ should be greater than that of observing $\mathtt{H}$ only $T-1$ times in that window. This is formally expressed as:

$$\mathbb{P}^{\pi_1}(\bigcirc^{(N)} \Box^{\leq T}\, \mathtt{H}^{\pi_1}) > \mathbb{P}^{\pi_2}(\bigcirc^{(N)}\, \varphi^{\pi_2}) + \varepsilon, \tag{14}$$

where $\varepsilon > 0$ is a parameter, $\varphi^{\pi_2}$ means there is one $\mathtt{M}$ for $N$ consecutive accesses, formally expressed as

$$\varphi^{\pi_2} = \left(\mathtt{M}^{\pi_2} \wedge \bigcirc \mathtt{H}^{\pi_2} \wedge \ldots \wedge \bigcirc^{(T-1)} \mathtt{H}^{\pi_2}\right) \vee \ldots \vee \left(\mathtt{H}^{\pi_2} \wedge \ldots \wedge \bigcirc^{(T-2)} \mathtt{H}^{\pi_2} \wedge \bigcirc^{(T-1)} \mathtt{M}^{\pi_2}\right)$$

where $\mathtt{B}$ indicates the initial state of an empty cache, and $\bigcirc^{(N)}$ represents the $N$-fold composition of $\bigcirc$. In (14), $\pi_1$ is a random execution of the cache replacement policy, where starting from the step $N$, there are $T$ consecutive hits $\mathtt{H}$; $\pi_2$ is a random execution, where starting from the step $N$, there is only one miss $\mathtt{M}$ for the next $T$ steps.

### E. Generalized Probabilistic Causation

HyperPCTL$^*$ can express conditional probabilities over multiple independent computation trees, which is not possible in HyperPCTL [5]. *Probabilistic causation* [37] asserts that if the cause $\psi^{\underline{\pi}}$ happens, the probability of occurring an effect $\varphi^{\underline{\pi}}$ should be higher than the probability of occurring $\varphi^{\underline{\pi}}$ when $\psi^{\underline{\pi}}$ does not happen. Here, we allow the cause and effect to be hyperproperties to capture probabilistic causality between security properties, e.g., the existence of a side-channel (see Section III-A) results in another side-channel. We can specify that for any two premises (i.e., initial states), $\psi^{\underline{\pi}}$ probabilistically causes $\varphi^{\underline{\pi}}$ as follows:

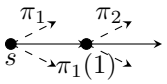


Fig. 5. Computation trees for (16). The dashed arrows show other possible sample values of the path variables to illustrate the probabilistic computation tree.

$$\frac{\mathbb{P}^{\underline{\pi}_1}(\psi^{\underline{\pi}_1} \wedge \varphi^{\underline{\pi}_1})}{\mathbb{P}^{\underline{\pi}_2}(\psi^{\underline{\pi}_2})} > \frac{\mathbb{P}^{\underline{\pi}_3}(\neg\psi^{\underline{\pi}_3} \wedge \varphi^{\underline{\pi}_3})}{\mathbb{P}^{\underline{\pi}_4}(\neg\psi^{\underline{\pi}_4})}. \tag{15}$$

In (15), $\underline{\pi}_1$ is a tuple of random executions where both the cause $\psi$ and effect $\varphi$ hold; $\underline{\pi}_2$ a tuple of random executions where the cause $\psi$ holds. Thus, the left-hand side of (15) is the conditional probability of the effect, when the cause holds. Similarly, the right-hand side of (15) is the conditional probability of the effect, when the cause does not hold.

The cause and effect in (15) can themselves be hyperproperties. For instance, the cause can be the violation of probabilistic noninterference (i.e., (11)) and the effect can be a breach of safety. That is, leakage of information increases the probability of compromising safety. This probabilistic causation of hyperproperties cannot be expressed by PCTL$^*$ or any of its existing extensions including [8].

## IV. Relation to Other Temporal Logics

In this section, we illustrate the expressive power of HyperPCTL$^*$ by comparing it with PCTL$^*$ [9], HyperPCTL [5], and HyperLTL [18].

### A. Relation to PCTL$^*$

In a PCTL$^*$ formula, a probability operator *implicitly* incorporates a single random sample path drawn from a (probabilistic) computation tree. In HyperPCTL$^*$, such random path variables are explicitly specified. For example, checking the nested PCTL$^*$ formula

$$\mathbb{P}^{J_1}\left(\bigcirc(\mathbb{P}^{J_2}(\varphi))\right),$$

involves two random sample paths from a root computation tree (for $\mathbb{P}^{J_1}$) and a sub computation tree (from the second state of the first path for $\mathbb{P}^{J_2}$), respectively. Thus, in order to specify this formula in HyperPCTL$^*$, we need to explicitly employ two random path variables $\pi_1$ and $\pi_2$ for the two probability operators, where sub-formula $\varphi$ is checked on $\pi_2$ of the sub computation tree, whose root is randomly given by $\pi_1(1)$ (see Figure 5). Hence, sub-formula $\bigcirc(\mathbb{P}^{J_2}(\varphi))$ is checked on $\pi_1$. The corresponding HyperPCTL$^*$ formula is:

$$\mathbb{P}^{\pi_1}\left(\bigcirc(\mathbb{P}^{\pi_2}(\varphi^{\pi_2}) \in J_2)^{\pi_1}\right) \in J_1. \tag{16}$$

Formally, we first show that HyperPCTL$^*$ subsumes PCTL$^*$. This is done by providing the set of rules to translate every PCTL$^*$ formula to a HyperPCTL$^*$ formula. We use the syntax and semantics of the PCTL$^*$ from Appendix A.

*Theorem 1:* HyperPCTL$^*$ subsumes PCTL$^*$.

*Proof:* We prove this statement by showing that any PCTL* formula can be transformed into a HyperPCTL* formula with the same meaning. In other words, for any given DTMC, the satisfaction/dissatisfaction of the formula is preserved during the transformation.

Given a DTMC $\mathcal{M}$, the satisfaction of a PCTL* state formula $\Phi$ (as defined in Appendix A) transforms into the satisfaction of a HyperPCTL* formula by

$$\mathcal{M} \models \Phi \text{ if and only if } (\mathcal{M}, V) \models \mathcal{T}(\Phi, \pi), \qquad (17)$$

for any path assignment $V$. In (17), the PCTL* state formula $\Phi$ implicitly involves a random path (more precisely, a random computation tree), which is explicitly named by $\pi$ in the corresponding HyperPCTL* formula. The transformation $\mathcal{T}$ is defined inductively as follows:

- $\mathcal{T}(\mathsf{a}, \pi) = \mathsf{a}^\pi$
- $\mathcal{T}(\neg\varphi, \pi) = \neg\mathcal{T}(\varphi, \pi)$
- $\mathcal{T}(\neg\Phi, \pi) = \neg\mathcal{T}(\neg\Phi, \pi)$
- $\mathcal{T}(\varphi_1 \wedge \varphi_2, \pi) = \mathcal{T}(\varphi_1, \pi) \wedge \mathcal{T}(\varphi_2, \pi)$
- $\mathcal{T}(\Phi_1 \wedge \Phi_2, \pi) = \mathcal{T}(\Phi_1, \pi) \wedge \mathcal{T}(\Phi_2, \pi)$
- $\mathcal{T}(\bigcirc\varphi, \pi) = \bigcirc\mathcal{T}(\varphi, \pi)$
- $\mathcal{T}(\varphi_1 \,\mathcal{U}^{\leq k}\, \varphi_2, \pi) = \mathcal{T}(\varphi_1, \pi) \,\mathcal{U}^{\leq k}\, \mathcal{T}(\varphi_2, \pi)$
- $\mathcal{T}(\mathbb{P}^J(\varphi), \pi) = \big(\mathbb{P}^{\pi'}\,(\mathcal{T}(\varphi, \pi')) \in J\big)^\pi$ with $\pi' \neq \pi$,

where $\Phi$ is a PCTL* state formula, $\varphi$ is a PCTL* path formula (as defined in Appendix A). The correctness of the transformation follows directly from the semantics of the logics. The transformation (17) holds for any path assignment $V$, since it can be shown that the path variables in $\mathcal{T}(\Phi, \pi)$ are all (probabilistically) quantified and actually do not receive assignment from $V$. ∎

Next, we show that HyperPCTL* *strictly* subsumes PCTL*. Specifically, we construct a DTMC and a HyperPCTL* formula, and show that this formula cannot be expressed by PCTL*.

*Theorem 2:* HyperPCTL* is strictly more expressive than PCTL* with respect to DTMCs.

*Proof:* Consider the DTMC shown in Figure 6 and the following HyperPCTL* formula:

$$\varphi = \left(\frac{\mathbb{P}^{\pi_1}\big(\mathsf{init}^{\pi_1} \Rightarrow \Diamond(\mathsf{a}_1^{\pi_1} \wedge \mathsf{a}_2^{\pi_1})\big)}{\mathbb{P}^{\pi_2}\big(\mathsf{init}^{\pi_2} \Rightarrow \Diamond\,\mathsf{a}_2^{\pi_2}\big)} = \frac{1}{2}\right).$$

Now, we prove that $\varphi$ cannot be expressed in PCTL*. By the syntax and semantics of PCTL*, it suffices to show that $\varphi$ cannot be expressed by a formula $\mathbb{P}(\psi)$, where $\psi$ is a PCTL* path formula derived by concatenating a set of PCTL* state formulas $\Phi_1, \ldots, \Phi_n$ with $\wedge, \neg$, or the temporal operators. These state formulas are either true or false in the states $s_0$, $s_1$, $s_2$, and $s_3$. Thus, the satisfaction of $\psi$ defines a subset of the paths $\mathsf{Paths}(s_0) = \{s_0 s_1^\omega, s_0 s_2^\omega, s_0 s_3^\omega\}$ in the DTMC. Since every path in $\mathsf{Paths}(s_0)$ is taken with probability $1/3$, formula $\mathbb{P}(\psi)$ can only evaluate to a value in $\{0, 1/3, 2/3, 1\}$. However, by the semantics of HyperPCTL*, the fractional probability on the right side of the implication has value $1/2$; thus, $\varphi$ evaluates to true and cannot be expressed by $\mathbb{P}(\psi)$ in PCTL*. ∎

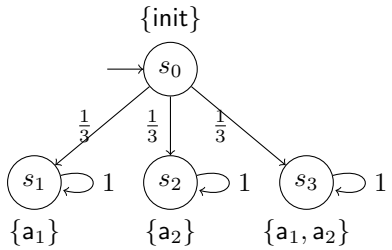


Fig. 6. DTMC where HyperPCTL* strictly subsumes PCTL*.

### B. Relation to HyperPCTL

Similar to HyperPCTL [5], HyperPCTL* allows probability arithmetics and comparison. For example, the HyperPCTL* formula

$$\varphi = \big(\mathbb{P}^{\pi_1}\,(\Diamond\,\mathsf{a}^{\pi_1}) - \mathbb{P}^{\pi_2}(\Diamond\,\mathsf{a}^{\pi_2}) > c\big)$$

for some $c \in \mathbb{R}$ means the satisfaction probability of "finally a" is greater at least by $c$ on a random path variable $\pi_1$ than another random path variable $\pi_2$. But in general, HyperPCTL and HyperPCTL* do not subsume each other.

*Theorem 3:* On DTMCs, HyperPCTL* strictly subsumes HyperPCTL.

*Proof:* From Theorem 2, HyperPCTL* subsumes PCTL*. However, HyperPCTL does not subsume PCTL* [5]. Thus, HyperPCTL does not subsume HyperPCTL*.

More specifically, HyperPCTL cannot express the satisfaction probability of a formula with more than two nested temporal operators. For example, the HyperPCTL* formula $\mathbb{P}^{(\pi_1,\pi_2,\pi_3)}(\mathsf{a}_1^{\pi_1}\,\mathcal{U}\,(\mathsf{a}_2^{\pi_2}\,\mathcal{U}\,\mathsf{a}_3^{\pi_3}))$ cannot be expressed by HyperPCTL. This is similar to the fact that PCTL cannot express the satisfaction probability of an LTL formula with more than two nested temporal operators (but PCTL* can).

On the other hand, HyperPCTL* contains all the syntactic rules of HyperPCTL, except for the (existential and universal) state quantifications [5]. A HyperPCTL formula with state quantifications can be expressed by HyperPCTL* by enumerating over the finite set of states of the DTMC. For example, HyperPCTL can specify that "there exists a state $s$, such that $\mathbb{P}^{(\pi_1,\pi_2)}(\mathsf{a}^{\pi_1}\,\mathcal{U}\,\mathsf{a}^{\pi_2}) > p$, where the initial state of $\pi_1, \pi_2$ is $s$." To express this in HyperPCTL*, we introduce an extra initial state that goes to all the states $\mathcal{S}$ of DTMC with probability $1/|\mathcal{S}|$. Then, the HyperPCTL specification can be expressed by

$$\bigvee_{s\in\mathcal{S}} \mathbb{P}^{(\pi_1,\pi_2)}\Big(\bigcirc\big((s^{\pi_1} \wedge s^{\pi_2}) \wedge (\mathsf{a}^{\pi_1}\,\mathcal{U}\,\mathsf{a}^{\pi_2})\big)\Big) > p/|\mathcal{S}|,$$

where "$\bigcirc$" appears because the paths $\pi_1, \pi_2$ now start from the new initial state. ∎

### C. Relation to HyperLTL

A HyperLTL formula can have multiple path variables. For example, let

$$\varphi_{\mathsf{hltl}} = \mathsf{a}_1^{\pi_1}\,\mathcal{U}\,\mathsf{a}_2^{\pi_2}$$

be a HyperLTL subformula (i.e., without path quantification), meaning that $\mathsf{a}_1$ is true on $\pi_1$ until $\mathsf{a}_2$ is true on $\pi_2$. Like PCTL*, which allows for reasoning over the satisfaction probability of LTL formulas, HyperPCTL* allows for reasoning over

the satisfaction probability of HyperLTL formulas. For example, HyperPCTL$^*$ subformula $\mathbb{P}^{(\pi_1,\pi_2)}(\varphi_{\mathsf{hltl}}) > c$ means that the satisfaction probability of the HyperLTL formula $\varphi_{\mathsf{hltl}}$ is greater than $c$. Moreover, in HyperPCTL$^*$, a HyperLTL formula can be probabilistically quantified in multiple ways. Specifically, the path variables of the HyperLTL formula can be quantified at one time, or one-by-one in a certain order. For example, instead of quantifying the HyperLTL formula $\varphi$ in a one-shot way for $\varphi_{\mathsf{hltl}}$, HyperPCTL$^*$ also allows formula

$$\psi_1 = \mathbb{P}^{\pi_1}\big(\,\mathbb{P}^{\pi_2}\,(\varphi_{\mathsf{hltl}}) > c_2\big) > c_1.$$

This means that the probability for finding path $\pi_1$ should be greater than $c_1$, such that the probability for finding another path $\pi_2$ to satisfy $\varphi_{\mathsf{hltl}}$ is greater than $c_2$. By flipping the order of the probabilistic quantification for $\pi_1$ and $\pi_2$, we derive the formula

$$\psi_2 = \mathbb{P}^{\pi_2}\big(\,\mathbb{P}^{\pi_1}\,(\varphi_{\mathsf{hltl}}) > c_2\big) > c_1.$$

Clearly, the meaning of $\psi_1$ and $\psi_2$ is different, showing the significance of the order of the probabilistic quantification.

## V. Statistical Model Checking

In this section, we design statistical model checking (SMC) algorithms for HyperPCTL$^*$ formulas on labeled discrete-time Markov chains. As with previous works on SMC [6], [41], [44], we focus on handling probabilistic operators by sampling. The temporal operators can be handled in the same way as for HyperLTL [18], and thus will not be discussed here.

### A. Challenges in Developing SMC for HyperPCTL$^*$

To statistically verify HyperPCTL$^*$, the main challenge is to use *sequential probability ratio tests* (SPRT) to handle the following issues:

- **Probabilistic quantification of multiple paths.** Consider the following formula:

$$\mathbb{P}^{\underline{\pi}}\,(\varphi) > p, \tag{18}$$

  where $\underline{\pi} = (\pi_1, \dots, \pi_n)$ is a tuple of path variables. Unlike the conventional SMC techniques, evaluating such a formula requires drawing *multiple* samples (we assume the truth value of $\varphi$ can be determined, given the sample value for $\underline{\pi}$).

- **Arithmetics of probabilistic quantifications.** Consider the following formula:

$$f\big(\,\mathbb{P}^{\underline{\pi}_1}\,(\varphi_1), \dots, \mathbb{P}^{\underline{\pi}_n}(\varphi_n)\big) > p,$$

  where for $i \in [n]$, $\underline{\pi}_i$ is a tuple of path variables and the truth value of $\varphi_i$ can be determined, given the sample value for $\underline{\pi}_i$. Equivalently, this can be expressed as

$$\big(\,\mathbb{P}^{\underline{\pi}_1}\,(\varphi_1), \dots, \mathbb{P}^{\underline{\pi}_n}(\varphi_n)\big) \in D, \tag{19}$$

  where

$$D = \{(x_1, \dots, x_n) \in [0,1]^n \mid f(x_1, \dots, x_n) > p\}.$$

  This can be viewed as an application of the currying technique in first-order logic that builds the equivalence between functions and relations [24]. In addition, since the functions $f$ is elementary from the syntax of HyperPCTL$^*$, the boundary of the domain $D$ is also elementary.

- **Nested probabilistic quantification.** Consider the following formula:

$$\mathbb{P}^{\underline{\pi}_1}\,\mathbb{P}^{\underline{\pi}_2} \dots \mathbb{P}^{\underline{\pi}_n}\,(\varphi) > p, \tag{20}$$

  where $i \in [n]$, $\underline{\pi}_i$ is a tuple of path variables and the truth value of $\varphi$ can be determined, given the sample value for all $\underline{\pi}_i$. This type of formula poses a challenge since the multiple paths drawn for each probability operator can be different from its previous or next operator.

These probabilistic quantifications are unique to HyperPCTL$^*$, therefore, they are not directly supported by existing statistical model checking algorithms designed for non-hyper probabilistic temporal logics [6]. In the next subsections, we address these challenges.

### B. Probabilistic Quantification of Multiple Parallel Paths

Consider the formula (18) again. We denote the satisfaction probability of the subformula $\varphi$ in (18) for a given DTMC $\mathcal{M}$ and path assignment $V$ by:

$$\begin{aligned} p_\varphi = \mathbf{Pr}\Big\{ &\Big(S_i \in \mathsf{Paths}\big(V(\pi_i)(0)\big)\Big)_{i\in[n]} : \\ &\Big(\mathcal{M}, V\big[\pi_i \mapsto S_i \text{ for all } i \in [n]\big]\Big) \models \varphi\Big\}. \end{aligned} \tag{21}$$

Following the standard procedure [6], [40], to simplify our discussion, we first assume that $\varphi$ is a bounded-time specification, i.e., its truth value can be evaluated on the finite prefixes of the sample paths. Unbounded-time specifications can be handled similarly with extra considerations on the time horizon. In addition, we make the following assumption on the *indifference region*.

*Assumption 1:* The satisfaction probability of $\varphi$ is not within the indifference region $(p-\varepsilon, p+\varepsilon)$ for some $\varepsilon > 0$; i.e.,

$$p_\varphi \notin (p-\varepsilon, p+\varepsilon). \tag{22}$$

From Assumption 1, to statistically verify (18), it suffices to solve the following hypothesis testing (HT) problem:

$$H_0 : p_\varphi \le p-\varepsilon, \quad H_1 : p_\varphi \ge p+\varepsilon. \tag{23}$$

The hypotheses $H_0$ and $H_1$ in (23) are *composite* since each of them contains infinitely many *simple* hypotheses of the form $H_0 : p_\varphi = p_0$ and $H_1 : p_\varphi = p_1$, respectively, where $p_0 \in [0, p-\varepsilon]$ and $p_1 \in [p+\varepsilon, 1]$.

To handle composite hypotheses with SPRT, a common technique is to consider the two most "indistinguishable" simple hypotheses

$$H_0 : p_\varphi = p-\varepsilon, \quad H_1 : p_\varphi = p+\varepsilon \tag{24}$$

from the two composite hypotheses in (23), respectively. From [44], if existing samples can test between $p-\varepsilon$ and $p+\varepsilon$ for some given statistical errors, then these samples are sufficient to test between $p-\varepsilon$ and $p_\varphi$ with the true satisfaction probability $p_\varphi \in [p+\varepsilon, 1]$ (or between $p+\varepsilon$ and $p_\varphi$ with

**Algorithm 1** SMC of $\mathbb{P}^{\underline{\pi}}(\varphi) > p$.

**Require:** Desired FP and FN ratios $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$, indifference parameter $\varepsilon$.

$N \leftarrow 0$, $T \leftarrow 0$.
**while** True **do**
  $N \leftarrow N + 1$.
  Draw a tuple of sample paths $\underline{S}_N$ (from the DTMC).
  **if** $\varphi$ is true on $\underline{S}_N$ **then**
    $T \leftarrow T + 1$.
  **end if**
  Update $\lambda(p+\varepsilon)$ and $\lambda(p-\varepsilon)$ by (25).
  Check the termination condition (27).
**end while**

the true satisfaction probability $p_\varphi \in [0, p-\varepsilon]$) for the same statistical errors (see Appendix D for details).

*Remark 1:* The indifference region assumption is necessary. If $\varepsilon = 0$, then $H_0$ and $H_1$ in (24) will be identical.

To statistically test between $H_0$ and $H_1$ from (24), suppose we have drawn $N$ statistically independent sample path tuples $\underline{S}_1, \dots, \underline{S}_N$ for the path variable $\underline{\pi}$ from the DTMC. Let $T$ be the number of sample path tuples, for which $\varphi$ is true. This is similar to the statistical model checking of PCTL$^*$ (see Appendix D for detailed description), except that the truth value of $\varphi$ needs to be evaluated for tuples of paths instead of single paths. Let us define, for $x \in (0,1)$, the log-likelihood function as

$$\lambda(x) = \ln\big(x^T (1-x)^{N-T}\big), \tag{25}$$

then $\lambda(p-\varepsilon)$ and $\lambda(p+\varepsilon)$ are the log-likelihood of the two hypotheses $H_0$ and $H_1$ in (24), respectively. As the number of sample path tuples $N$ increase, the log-likelihood ratio $\lambda(p+\varepsilon) - \lambda(p-\varepsilon)$ should increase (with high probability) if $H_1$ holds, and should decrease if $H_0$ holds. To achieve desired the *false positive* (FP) and *false negative* (FN) ratios $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$, respectively, defined by[4]:

$$\begin{aligned} \alpha_{\mathrm{FP}} &= \mathbf{Pr}\big(\text{assert } H_1 \,|\, H_0 \text{ is true}\big), \\ \alpha_{\mathrm{FN}} &= \mathbf{Pr}\big(\text{assert } H_0 \,|\, H_1 \text{ is true}\big), \end{aligned} \tag{26}$$

the SPRT algorithm should continue sampling, i.e., increase the number of samples $N$, until one of the two following termination conditions hold [49]:

$$\begin{cases} \text{assert } H_0, & \text{if } \lambda(p-\varepsilon) - \lambda(p+\varepsilon) > \ln \frac{1-\alpha_{\mathrm{FN}}}{\alpha_{\mathrm{FP}}}, \\ \text{assert } H_1, & \text{if } \lambda(p+\varepsilon) - \lambda(p-\varepsilon) > \ln \frac{1-\alpha_{\mathrm{FP}}}{\alpha_{\mathrm{FN}}}. \end{cases} \tag{27}$$

This process is summarized by Algorithm 1.

### C. Arithmetics of Probabilistic Quantifications

Now, consider formula (19). We denote the satisfaction probability of $\mathbb{P}^{\underline{\pi}_i}(\varphi_i)$ for each $i \in [n]$ for a given DTMC $\mathcal{M}$ and path assignment $V$ by:

$$\begin{aligned} p_{\varphi_i} = \mathbf{Pr}\Big\{ \Big(S_l \in \mathsf{Paths}\big(V(\pi_l)(0)\big)\Big)_{l\in[k_i]} : \\ \Big(\mathcal{M}, V\big[\pi_l \mapsto S_l \text{ for all } l \in [k_i]\big]\Big) \models \varphi_i \Big\}, \end{aligned} \tag{28}$$

where

$$\underline{\pi}_i = (\pi_{i1}, \dots, \pi_{ik_i}), \quad k_i = |\underline{\pi}_i|.$$

Again, we assume that each $\varphi_i$ is a bounded-time specification, as we did for (21). This problem can be converted into the (multi-dimensional) HT problem in $\mathbb{R}^n$ by

$$H_0 : \underline{p_\varphi} \in D, \quad H_1 : \underline{p_\varphi} \in D^{\mathrm{c}}, \tag{29}$$

where $D$ is as defined in (19), $D^{\mathrm{c}}$ is the complement of $D$, and $\underline{p_\varphi} = (p_{\varphi_1}, \dots, p_{\varphi_n})$.

We now propose a novel SPRT algorithm for this $n$-dimensional HT problem, by extending the common SPRT algorithm from Section V-B to multi-dimension. By following the same idea, we first generalize the notion of *indifference regions* to the multi-dimensional case. Based on this, we propose a geometric condition to identify the two most indistinguishable cases $\underline{r}$ and $\underline{q}$ from the test regions $D$ and $D^{\mathrm{c}}$, such that it suffices to consider the HT problem:

$$H_0' : \underline{p_\varphi} = \underline{r}, \quad H_1' : \underline{p_\varphi} = \underline{q}. \tag{30}$$

Once $\underline{q}$ and $\underline{r}$ in (30) are known, then we can solve it in the same way as done in Section V-B. Specifically, in (19), for each $i \in [n]$, we draw $N$ sample path tuples for the path variable $\underline{\pi}_i$ from the DTMC and let $T_i$ be the number of sample path tuples, for which $\varphi_i$ is true. Consider the log-likelihood function defined as

$$\lambda(\underline{x}) = \ln\left(\prod_{i\in[n]} x_i^{T_i}(1-x_i)^{N-T_i}\right), \tag{31}$$

where $\underline{x} = (x_1, \dots, x_n) \in (0,1)^n$. Clearly, $\lambda(\underline{r})$ and $\lambda(\underline{q})$ are the log-likelihood of the two hypotheses in (30), respectively. So, an SPRT algorithm can be constructed based on the log-likelihood ratio $\lambda(\underline{r}) - \lambda(\underline{q})$ (or equivalently $\lambda(\underline{q}) - \lambda(\underline{r})$). Below, we explain how to derive $\underline{q}$ and $\underline{r}$.

#### *Multi-Dimensional Indifference Region*

To ensure that we can find different values for $\underline{q}$ and $\underline{r}$ in (30) (so that $H_0$ and $H_1$ are not identical), we introduce a multi-dimensional version of the indifference region assumption. It ensures that the two test regions in (29) are separated, as formally stated below. This is similar to the case in Section V-B (see Assumption 1 and Remark 1).

*Assumption 2:* The test region $D$ is convex and there exists convex $D_0, D_1 \subseteq [0,1]^n$, such that $D_0 \subseteq D \subseteq D_1$, and the Hausdorff distance $d_{\mathrm{H}}(D_0, D_1) > 0$, where

$$d_{\mathrm{H}}(X,Y) = \max\Big\{ \sup_{x\in X}\inf_{y\in Y}\|x-y\|_2, \ \sup_{y\in Y}\inf_{x\in X}\|x-y\|_2 \Big\}.$$

For simplicity, we assume that the boundaries of $D_0$ and $D_1$ are respectively defined by the boundary equations

$$F_0(\underline{x}) = 0, \quad \text{and} \quad F_1(\underline{x}) = 0, \tag{32}$$

where $\underline{x} \in \mathbb{R}^n$, and $F_0$ and $F_1$ are elementary functions.[5]

In general, there exist $D_0$ and $D_1$ such that $D_0 \subseteq D \subseteq D_1$, when $\underline{p_\varphi}$ is not on the boundary of the test region $D$, i.e.,

[4] Here, $\mathbf{Pr}(\cdot \,|\, \cdot)$ stands for the conditional probability.

[5] For example, if the boundary of $D_0$ is a circle of radius 0.2 centered at $(0.5, 0.5)$, then the elementary function $F_0(p_1, p_2) = (p_1 - 0.5)^2 + (p_2 - 0.5)^2 - 0.2^2$.

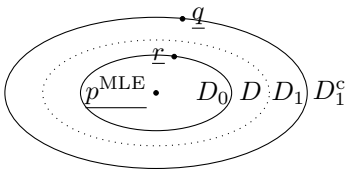


Fig. 7. Given the test region $D$, we assume there exists an indifference region formed by $D_1 \backslash D_0$. If $p^{\mathrm{MLE}}$ from (35) satisfies $p^{\mathrm{MLE}} \in D_0$, then we find $\underline{r} \in D_0$ by (41) and $\underline{q} \in D_1^{\mathrm{c}}$ by (37).

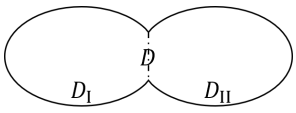


Fig. 8. Partition of a non-convex test region.

$\underline{p_\varphi} \notin \partial D$. Using Assumption 2, we derive the HT problem for verifying (29)

$$H_0 : \underline{p_\varphi} \in D_0, \quad H_1 : \underline{p_\varphi} \in D_1^{\mathrm{c}}, \tag{33}$$

where $\underline{p_\varphi} = (p_{\varphi_1}, \dots, p_{\varphi_n})$. As illustrated in Figure 7, the region $D_1 \backslash D_0$ is the *indifference region*, keeping $\underline{p_\varphi}$ statistically distinguishable from the boundary of the *test region* $D$. Again, the HT problem (33) is *composite*.

*Remark 2:* From Assumption 2, if $\overline{D} = \overline{D'}$, where $\overline{D}$ and $\overline{D'}$ are respectively the closure of $D$ and $D'$, then verifying $(\mathbb{P}^{\pi_1}\varphi_1, \dots, \mathbb{P}^{\pi_n}\varphi_n) \in D$ is equivalent to verifying $(\mathbb{P}^{\pi_1}\varphi_1, \dots, \mathbb{P}^{\pi_n}\varphi_n) \in D'$. Thus, they will not be differentiated in the rest of the paper.

*Remark 3:* The condition that the test region $D$ is convex in Assumption 2 is only technical. If $D$ is non-convex, then we can divide $D$ into several convex subregions, and convert the HT problem (29) into several sub-problems with convex test regions. From example, the non-convex test region $D$ illustrated in Figure 8 can be divided into the union of two convex test regions $D_I$ and $D_{II}$. Therefore, to test if $\underline{p_\varphi} \in D$, it suffices to test if $\underline{p_\varphi} \in D_I$ or $\underline{p_\varphi} \in D_{II}$, and the overall statistical test error is the sum of errors of these two sub-tasks.

### *Identifying Most Indistinguishable Simple Hypotheses*

To solve the HT problem (33), suppose that we have drawn $N$ sample path tuples for each path variable $\underline{\pi}_i$ ($i \in [n]$). Let $T_i$ be the number of sample path tuples, for which $\varphi_i$ is true. Then, we have

$$T_i \sim \mathrm{Binom}(N, p_{\varphi_i}). \tag{34}$$

The maximal likelihood estimator (MLE) of $\underline{p_\varphi}$ is

$$\underline{p^{\mathrm{MLE}}} = (p_1^{\mathrm{MLE}}, \dots, p_n^{\mathrm{MLE}}) = \Big(\frac{T_1}{N}, \dots, \frac{T_n}{N}\Big). \tag{35}$$

If $\underline{p^{\mathrm{MLE}}} \in D_0$, then intuitively we should assert the hypothesis $H_0$ against $H_1$. The statistical error of this assertion can be measured by the likelihood ratio $\lambda(\underline{r}) - \lambda(\underline{q})$ for some $\underline{r} \in D_0$ and $\underline{q} \in D_1^{\mathrm{c}}$, which will be decided below. Specifically, to assert $H_0$ (or $H_1$) with certain desired FP and FN ratios, the likelihood ratio should be greater (or less) than some threshold, which is (only) a function of the given FP and FN ratios (see Appendix D).

As illustrated in Figure 7, we can identify $\underline{q} \in D_1^{\mathrm{c}}$ by maximizing the likelihood for the (simple) hypothesis $\underline{p_\varphi} = \underline{q}$ for any $\underline{q} \in D_1^{\mathrm{c}}$. Intuitively, since any other simple hypothesis in $\underline{p_\varphi} \in D_1^{\mathrm{c}}$ yields a larger likelihood ratio, to use SPRT to solve the HT problem (33), it suffices to only consider the simple hypothesis $\underline{p_\varphi} = \underline{q}$ from the composite hypothesis $\underline{p_\varphi} \in D_1^{\mathrm{c}}$. This is formally stated below.

*Lemma 1:* If $\underline{p^{\mathrm{MLE}}} \in D_0$, to assert $H_0$ (against $H_1$) in the HT problem (33), it suffices to assert this $H_0$ (against $H_1'$) in the HT problem

$$H_0 : \underline{p_\varphi} \in D_0, \quad H_1' : \underline{p_\varphi} = \underline{q}, \tag{36}$$

where $\underline{q}$ is given by

$$\underline{q} = \mathrm{argmax}_{\underline{x} \in D_1^{\mathrm{c}}} \lambda(\underline{x}), \tag{37}$$

with $\lambda(\cdot)$ being the log-likelihood ratio given by (31).

*Proof:* Given any possible value of $\underline{p_\varphi} \in D_0$, for any $\underline{q'} \in D_1^{\mathrm{c}}$, and any likelihood ratio threshold $B > 0$, we have

$$\lambda(\underline{p_\varphi}) - \lambda(\underline{q}) > B \implies \lambda(\underline{p_\varphi}) - \lambda(\underline{q'}) > B,$$

where $\underline{q}$ is given by (37). Thus, for given sample paths (from (34)), if the SPRT algorithm asserts $H_0$ for the HT problem (36), then it should also assert $H_0$ for the HT problem (33). The two assertions have the same statistical errors because they use the same likelihood ratio threshold $B$. ■

To obtain $\underline{q}$ from (37), by the convexity of the test region $D_1$ and the function $\lambda(\cdot)$, the maximum is achieved at the boundary of $D_1$. That is, from (32) it holds that

$$F_1(\underline{q}) = 0. \tag{38}$$

In addition, by the first-order condition of optimality under the constrained (38), the maximum of $\underline{q}$ is achieved when the direction of the gradient $\nabla\lambda(\underline{q})$ aligns with the normal vector $\nabla F_1(\underline{q})$ of the boundary. That is, for some $c \neq 0$ it holds that

$$\nabla F_1(\underline{q}) = c\nabla\lambda(\underline{q}) = \Big(\frac{c(p_i^{\mathrm{MLE}} - q_i)}{q_i(1-q_i)}\Big)_{i\in[n]}. \tag{39}$$

Given $\underline{q}$ from Lemma 1, we identify $\underline{r} \in D_0$ by minimizing the Kullback-Leibler divergence from the hypothesis $\underline{p_\varphi} = \underline{r}$ to the hypothesis $\underline{p_\varphi} = \underline{q}$ for any $\underline{r} \in D_0$, as illustrated in Figure 7. Generally, the Kullback-Leibler divergence measures the hardness of using SPRT to distinguish between two simple hypotheses [49]. Thus, to use SPRT to solve the HT problem (36), it suffices to only consider the simple hypothesis $\underline{p_\varphi} = \underline{r}$ from the composite hypothesis $\underline{p_\varphi} \in D_0$. This is formally stated below.

*Lemma 2:* If $\underline{p^{\mathrm{MLE}}} \in D_0$, to assert $H_0$ (against $H_1'$) in the HT problem (36), it suffices to assert this $H_0'$ (against $H_1'$) in the HT problem

$$H_0' : \underline{p_\varphi} = \underline{r}, \quad H_1' : \underline{p_\varphi} = \underline{q}, \tag{40}$$

where using $\underline{q}$ from (37), we have

$$\underline{r} = \mathrm{argmin}_{\underline{x} \in D_0} K(\underline{x} \| \underline{q}), \tag{41}$$

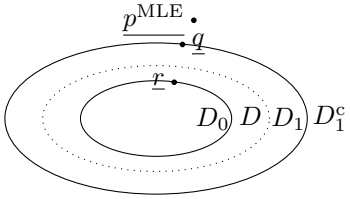


Fig. 9. Given the indifference region formed by $D_1 \backslash D_0$, if $p^{\mathrm{MLE}}$ from (35) satisfies $p^{\mathrm{MLE}} \in D_1^{\mathrm{c}}$, then we find $\underline{r} \in D_0$ by (44) and $\underline{q} \in D_1^{\mathrm{c}}$ by (45).

where the Kullback-Leibler divergence is given by

$$K(\underline{x} \| \underline{q}) = \sum_{i \in [n]} x_i \ln\left(\frac{x_i}{q_i}\right) + (1 - x_i) \ln\left(\frac{1 - x_i}{1 - q_i}\right).$$

*Proof:* We defer the proof to Theorem 4. ∎

To solve $\underline{r}$ from (41), by the convexity of the test region $D_0$ and the function $K(\cdot \| \underline{q})$, the maximum is achieved at the boundary of $D_0$, i.e.,

$$F_0(\underline{r}) = 0. \tag{42}$$

In addition, by the first-order condition of optimality under the constraint (42), the maximum of $\underline{r}$ is achieved when the direction of the gradient $\nabla_{\underline{r}} K(\underline{r} \| \underline{q})$ aligns with the normal vector $\nabla F_0(\underline{r})$ of the boundary – i.e., for some $c \neq 0$, it holds that

$$\nabla F_0(\underline{r}) = \left(c\big(\ln(\frac{r_i}{q_i}) - \ln(\frac{1 - r_i}{1 - q_i})\big)\right)_{i \in [n]}. \tag{43}$$

When $d_{\mathrm{H}}(D_0, D_1) \to 0$, we have $\underline{r} - \underline{q} \to 0$, and thus

$$\nabla F_0(\underline{r}) \to \left(\frac{c(r_i - q_i)}{q_i(1 - q_i)}\right)_{i \in [n]}.$$

The case that $p^{\mathrm{MLE}} \in D_1$ can be handled in the same way. As shown in Figure 9, we can first derive $\underline{r}$ in the same way as (37) by

$$\underline{r} = \mathrm{argmax}_{\underline{x} \in D_0} \lambda(\underline{x}), \tag{44}$$

Then, using $\underline{r}$ from (44), we derive $\underline{q}$ in the same way as (41); i.e.,

$$\underline{q} = \mathrm{argmin}_{\underline{x} \in D_1^{\mathrm{c}}} K(\underline{x} \| \underline{r}). \tag{45}$$

Thus, to achieve the FP and FN ratios $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$, the SPRT algorithm should continue sampling until one of the following termination conditions is satisfied:

$$\begin{cases} \text{assert } H_0, \text{ if } \underline{p^{\mathrm{MLE}}} \in D_0 \text{ and } \lambda(\underline{r}) - \lambda(\underline{q}) > \ln \frac{1 - \alpha_{\mathrm{FN}}}{\alpha_{\mathrm{FP}}} \\ \text{assert } H_1, \text{ if } \underline{p^{\mathrm{MLE}}} \in D_1^{\mathrm{c}} \text{ and } \lambda(\underline{q}) - \lambda(\underline{r}) > \ln \frac{1 - \alpha_{\mathrm{FP}}}{\alpha_{\mathrm{FN}}} \end{cases} \tag{46}$$

The above discussion is summarized by Theorem 4 and Algorithm 2.

*Theorem 4:* Under Assumption 2, Algorithm 2 terminates with probability 1, and its FP and FN ratios are no greater than $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$.

*Proof:* Without loss of generality, we consider $\underline{p_\varphi} \in D_0$; the same applies to the case $\underline{p_\varphi} \in D_1^{\mathrm{c}}$. By the central limit theorem, as the number of samples increases ($N \to \infty$), the probability that $p^{\mathrm{MLE}} \in D_1 \backslash D_0$ converges to 0.

**Algorithm 2** SMC of $(\mathcal{M}, V) \models (\mathbb{P}^{\pi_1} \varphi_1, \dots, \mathbb{P}^{\pi_n} \varphi_n) \in D$.

**Require:** Desired FP/FN ratios $\alpha_{\mathrm{FP}}/\alpha_{\mathrm{FN}}$, test regions $D_0, D_1$.

$N \leftarrow 0$, $T_i \leftarrow 0, \forall i \in [n]$.
**while** True **do**
  $N \leftarrow N + 1$.
  **for** $i \in [n]$ **do**
    Draw a tuple of sample paths $\underline{S_i}$.
    **if** $\varphi_i$ is true on $\underline{S_i}$ **then**
      $T_i \leftarrow T_i + 1$.
    **end if**
  **end for**
  Compute $\underline{p^{\mathrm{MLE}}}$ by (35).
  **if** $\underline{p^{\mathrm{MLE}}} \in D_0$ **then**
    Compute $\underline{q}$ and $\underline{r}$ by (37), (41) (via (38), (39), (42), and (43)).
  **else if** $\underline{p^{\mathrm{MLE}}} \in D_1^{\mathrm{c}}$ **then**
    Compute $\underline{r}$ and $\underline{q}$ by (44), (45).
  **else**
    Continue.
  **end if**
  Compute $\lambda(\underline{q})$ and $\lambda(\underline{r})$ by (31).
  Check the termination condition (46).
**end while**

In addition, the expected value of the log-likelihood ratio $\mathbb{E}(\lambda(\underline{r}) - \lambda(\underline{q})) \to \infty$, and thus the probability that (46) is not yet satisfied, converges to 0. Therefore, Algorithm 2 terminates with probability 1.

Now, we prove the FP and FN ratios of Algorithm 2. By (41), for any $\underline{r'} \in D_0$, the expectation of the log-likelihood ratio satisfies

$$\begin{aligned} \mathbb{E}_{\underline{p_\varphi} = \underline{r'}}\big(\lambda(\underline{r'}) - \lambda(\underline{q})\big) &= K(\underline{r'} \| \underline{q}) \\ &\geq K(\underline{r} \| \underline{q}) = \mathbb{E}_{\underline{p_\varphi} = \underline{r}}(\lambda(\underline{r}) - \lambda(\underline{q})), \end{aligned}$$

for $K$ from (41). Therefore, for any $B > 0$, we have

$$\mathbf{Pr}_{\underline{p_\varphi} = \underline{r'}}(\lambda(\underline{r'}) - \lambda(\underline{q}) > B) \geq \mathbf{Pr}_{\underline{p_\varphi} = \underline{r}}(\lambda(\underline{r}) - \lambda(\underline{q}) > B).$$

This implies that for any possible value of $\underline{p_\varphi} \in D_0$, the probability of asserting $H_0$ by (46) using the SPRT is not less than that of $\underline{p_\varphi} = \underline{r}$, which is $1 - \alpha_{\mathrm{FP}}$. Therefore, Lemma 2 and Theorem 4 hold. ∎

*Remark 4:* For computing (37) and (41), one can either use optimization or solve it via the necessary conditions (38), (39), (42), and (43), which may have analytic solutions as the boundary functions $F_0(\cdot)$ and $F_1(\cdot)$ are elementary functions (especially when $F_0(\cdot)$ and $F_1(\cdot)$ are linear functions). Since solving the optimization problem at every iteration can be inefficient for some cases this, we can reduce the frequency of computing the significance level by drawing samples in batches.

### D. Nested probabilistic quantification

The nested probabilistic quantification in HyperPCTL$^*$ can be handled in the same way as [50]. Thus, the nested probability operators in (20) can be handled in the same way as done in [50], and we omit describing it here.

## VI. Case Studies and Evaluation

We evaluated the presented SMC algorithms on the case studies described in Section III. It is important to highlight that all these HyperPCTL$^*$ specifications are currently not verifiable by existing probabilistic model checkers and SMC tools. The simulations were performed on a laptop with Intel® Core™ i7-7820HQ, 2.92GHz Processor with 32GB RAM. The simulation code is available at [20]. The assertions of the proposed SMC algorithms are compared with "the correct answers", which are derived by extensive simulations or exhaustive solutions. These "the correct answers" are also used to check the validity of the indifference region assumption on the case studies. The running time, number of samples, and the accuracy of the proposed algorithms (Number of correct assertions / Number of total assertions) are estimated based on 100 runs for each SMC task. The results are presented in Tables I to IV, respectively. In all the setups, the estimated accuracy agrees with the fixed desired significance levels ($\alpha_{\text{FP}} = \alpha_{\text{FN}} = 0.01$), except for one case in Table III. This is because of the statistical error of the estimated accuracy using only 100 runs. The average execution time in the worst case is less than 30 seconds.

### A. Side-channel Vulnerability

We verified the correctness of the HyperPCTL$^*$ specification (9) on GabFeed chat server [1]. The authentication algorithm in this version of GabFeed has been reported to have a side channel vulnerability that leaks the number of set bits in the secret key [48]. The vulnerability can be exploited by the attacker by observing the execution time across different public keys, as discussed in Section III; hence, as with [48], we verify the security policy (9) for a selection of security keys. We instrumented the source code to obtain the execution time for a combination of the secret key and public key, and generate a trace in a discrete-time fashion. For a given secret key, we select a random public key and generate a trace from it. Using this approach we were able to show the existence of side-channel – i.e., the negation of (9) holds with confidence level 0.99. The results are shown in Table I.

### B. Probabilistic Noninterference

We showed the violation of specification (11) for $N \in \{20, 50, 100\}$ threads (the results are similar for $l = 1$). The obtained results are presented in Table II. The total number of states of the DTMC is at least $N!$, so we simulate it using a transition-matrix-free approach to meet the memory constraint. As the significance level decreases, namely a more accurate assertion is asked for, the sample cost and the running time increase accordingly.

### C. Security of Dining Cryptographers

We verified the correctness of the specification (13) with $i = 1, j = 2$ on the model provided by [2] for $N \in \{100, 1000\}$ cryptographers and approximate equivalence parameter $\varepsilon \in \{0.2, 0.1, 0.0.5\}$. The obtained results are summarized in Table III. The total number of states of the DTMC is at least $2^N$, and we simulate it with a transition-matrix-free approach. As the approximate equivalence parameters increases, the specification is increasingly relaxed, so the sample cost and the running time decrease accordingly.

### D. Randomized cache Replacement Policy

We verified the correctness of the specification (14) for the performance of random replacement cache policy described in Section III. The performance of random replacement policy is evaluated on random memory accesses from a normal distribution with variance less than the cache size, to emulate the *locality of reference*. With the random replacement policy and the random access sequence, the dynamics of the cache modeled by the Mealy machine described in Section III can be captured by a DTMC.

We consider the paths of the DTMC with labels `H` or `M`, depending on the outcome of the cache access. We compared the probability of all hits to the probability of seeing a single miss `M` on a fully associative cache with 256 lines for a program of 1024 blocks. This can easily be extended to set associative cache with arbitrary program size. The results are shown in Table IV. We observe that the algorithm takes longer time for $T = 20$ than $T = 10$. This is because, for shorter $T$, the probability of observing all hits `H` is more than the probability of observing a miss `M`. As the trace length increases, these probabilities become closer.

## VII. Related Work

To the best of our knowledge, the only existing SMC algorithm for hyper temporal logics is the one proposed in [50]. It handles complex probabilistic quantifications similar to HyperPCTL$^*$ but using a multi-dimensional extension of Clopper-Pearson confidence interval, whereas, in this paper, our focus is on SPRT. Moreover, the application domain of [50] is on timed hyperproperties and cyber-physical systems, whereas, here, we concentrate on applications in information-flow security. This algorithm provides provable probabilistic guarantees for any desired false positive $\alpha_{\text{FP}} \in (0, 1)$ (the probability of wrongly claiming a false formula to be true) and false negative $\alpha_{\text{FN}} \in (0, 1)$ (the probability of wrongly claiming a true formula to be false).

Temporal logics for probabilistic hyperproperties was first introduced in [5] for DTMCs and in [4], [22] for MDPs. In these papers, the authors introduce exhaustive model checking algorithms for verification of probabilistic hyperproperties. Parameter synthesis for HyperPCTL was studied in [3].

Randomization is used in different contexts to quantify the amount of information leak as well as to provide probabilistic guarantees about the correctness of security policies. A classic example is probabilistic noninterference [34], [35], which requires that high-security input should not change the probability of reaching low-security outputs. There has been extensive work in this area including using probabilistic bisimulation to reason about probabilistic noninterference in multi-threaded programs [43]. Another prominent line of work is *quantitative information flow* [38], [46], which relates information theory to independent executions of a system and uses different notions of entropy to quantify the amount information leaked across different executions.

Recently, there has been significant progress in automatically verifying [19], [30]–[32] and monitoring [7], [11], [13], [28], [29], [36], [47] HyperLTL specifications. HyperLTL is also supported by a growing set of tools, including the model checker MCHyper [19], [32], the satisfiability checkers

| $\tau$ | $\varepsilon$ | $\delta$ | Acc. | No. Samples | Time (s) |
|---|---|---|---|---|---|
| 60 | 0.05 | 0.01 | 1.00 | 5.5e+02 | 0.54 |
| 60 | 0.05 | 0.001 | 1.00 | 5.5e+03 | 5.76 |
| 60 | 0.1 | 0.01 | 1.00 | 6.1e+02 | 0.60 |
| 60 | 0.1 | 0.001 | 1.00 | 6.2e+03 | 7.16 |
| 90 | 0.05 | 0.01 | 1.00 | 3.7e+02 | 0.46 |
| 90 | 0.05 | 0.001 | 1.00 | 3.7e+03 | 4.94 |
| 90 | 0.1 | 0.01 | 1.00 | 4.1e+02 | 0.48 |
| 90 | 0.1 | 0.001 | 1.00 | 4.1e+03 | 5.37 |
| 120 | 0.05 | 0.01 | 1.00 | 3.8e+02 | 6.96 |
| 120 | 0.05 | 0.001 | 1.00 | 2.2e+03 | 11.24 |
| 120 | 0.1 | 0.01 | 1.00 | 3.8e+02 | 6.05 |
| 120 | 0.1 | 0.001 | 1.00 | 2.3e+03 | 9.46 |

TABLE I. SHOWING THE VIOLATION OF TIMING SIDE CHANNEL VULNERABILITY FOR DIFFERENT COMBINATIONS OF TIME THRESHOLDS $\tau$ SECONDS, APPROXIMATE EQUIVALENCE PARAMETER $\varepsilon$ AND INDIFFERENCE REGION $\delta$ BASED ON THE AVERAGE OF 100 RUNS.

| $N$ | $\delta$ | Acc. | No. Samples | Time (s) |
|---|---|---|---|---|
| 20 | 0.01 | 1.00 | 7.7e+02 | 0.49 |
| 20 | 0.001 | 1.00 | 7.6e+03 | 6.45 |
| 50 | 0.01 | 1.00 | 7.0e+02 | 0.48 |
| 50 | 0.001 | 1.00 | 6.8e+03 | 6.39 |
| 100 | 0.01 | 1.00 | 6.5e+02 | 0.54 |
| 100 | 0.001 | 1.00 | 6.6e+03 | 7.10 |

TABLE II. SHOWING THE VIOLATION OF PROBABILISTIC NONINTERFERENCE FOR DIFFERENT COMBINATIONS OF NUMBER OF THREADS $N$ AND INDIFFERENCE REGION $\delta$, BASED ON THE AVERAGE OF 100 RUNS.

| $N$ | $\varepsilon$ | Acc. | No. Samples | Time (s) |
|---|---|---|---|---|
| 100 | 0.05 | 1.00 | 1.0e+03 | 0.91 |
| 100 | 0.1 | 1.00 | 5.2e+02 | 0.39 |
| 100 | 0.2 | 1.00 | 2.8e+02 | 0.14 |
| 1000 | 0.05 | **0.98** | 1.1e+03 | 3.27 |
| 1000 | 0.1 | 1.00 | 5.5e+02 | 1.52 |
| 1000 | 0.2 | 1.00 | 2.8e+02 | 0.69 |

TABLE III. VERIFYING THE SECURITY OF DINING CRYPTOGRAPHERS FOR DIFFERENT COMBINATIONS OF NUMBER OF CRYPTOGRAPHERS $N$ AND APPROXIMATE EQUIVALENCE PARAMETER $\varepsilon$ FOR INDIFFERENCE REGION $\delta = 0.01$, BASED ON THE AVERAGE OF 100 RUNS.

| $T$ | $\varepsilon$ | $\delta$ | Acc. | No. Samples | Time (s) |
|---|---|---|---|---|---|
| 10 | 0.05 | 0.01 | 1.00 | 1.1e+02 | 0.13 |
| 10 | 0.05 | 0.001 | 1.00 | 1.0e+03 | 2.56 |
| 10 | 0.01 | 0.01 | 1.00 | 1.2e+02 | 0.14 |
| 10 | 0.01 | 0.001 | 1.00 | 1.2e+03 | 2.79 |
| 20 | 0.05 | 0.01 | 1.00 | 6.0e+02 | 1.49 |
| 20 | 0.05 | 0.001 | 1.00 | 6.2e+03 | 16.73 |
| 20 | 0.01 | 0.01 | 0.99 | 1.2e+03 | 2.97 |
| 20 | 0.01 | 0.001 | 1.00 | 1.1e+04 | 28.99 |

TABLE IV. VERIFYING THE PERFORMANCE OF RANDOM REPLACEMENT POLICY FOR DIFFERENT COMBINATIONS OF TRACE LENGTH ($T$) AND APPROXIMATION PARAMETER ($\varepsilon$) AND INDIFFERENCE REGION ($\delta$), BASED ON THE AVERAGE OF 100 RUNS.

EAHyper [27] and MGHyper [25], and the runtime monitoring tool RVHyper [28]. Synthesis techniques for HyperLTL are studied in [26] and in [10].

## VIII. CONCLUSION

In this paper, we studied the problem of statistical model checking (SMC) of hyperproperties on discrete-time Markov chains (DTMCs). First, to reason about probabilistic hyperproperties, we introduced the probabilistic temporal logic HyperPCTL$^*$ that extends PCTL$^*$ by allowing explicit and simultaneous quantification over paths. In addition, we proposed an SMC algorithm for HyperPCTL$^*$ specifications on DTMCs. Unlike existing SMC algorithms for hyperproperties based on Clopper-Pearson confidence interval, we proposed sequential probability ratio tests (SPRT) with a new notion of indifference margin. Finally, we evaluated our SMC algorithms on four case studies: time side-channel vulnerability in encryption, probabilistic anonymity in dining cryptographers, probabilistic noninterference of parallel programs, and the performance of a random cache replacement policy.

For future work, we are currently developing SMC algo-

rithms for verification of timed hyperproperties in probabilistic systems. Another interesting research avenue is developing exhaustive model checking algorithms for HyperPCTL$^*$. One can also develop symbolic techniques for verification of HyperPCTL$^*$ specifications. We also note that our approach has the potential of being generalized to reason about the conformance of two systems (e.g., an abstract model and its refinement) with respect to hyperproperties.

## Acknowledgment

This work is sponsored in part by the ONR under agreements N00014-17-1-2504 and N00014-20-1-2745, AFOSR under award number FA9550-19-1-0169, as well as the NSF CNS-1652544 and NSF SaTC-1813388 grants.

## References

[1] https://github.com/Apogee-Research/STAC/tree/master/Engagement_Challenges/Engagement_2/gabfeed_1.

[2] https://www.prismmodelchecker.org/casestudies/index.php.

[3] E. Ábrahám, E. Bartocci, B. Bonakdarpour, and O. Dobe, "Parameter synthesis for probabilistic hyperproperties," in *Proceedings of the 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, 2020, to appear.

[4] ——, "Probabilistic hyperproperties with nondeterminism," in *Proceedings of the 18th Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2020, to appear.

[5] E. Ábrahám and B. Bonakdarpour, "HyperPCTL: A temporal logic for probabilistic hyperproperties," in *Proceedings of the 15th International Conference on Quantitative Evaluation of Systems (QEST)*, 2018, pp. 20–35.

[6] G. Agha and K. Palmskog, "A Survey of Statistical Model Checking," *ACM Transactions on Modeling and Computer Simulation*, vol. 28, no. 1, pp. 6:1–6:39, 2018.

[7] S. Agrawal and B. Bonakdarpour, "Runtime verification of k-safety hyperproperties in HyperLTL," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, 2016, pp. 239–252.

[8] M. E. Andrés and P. van Rossum, "Conditional probabilities over probabilistic and nondeterministic systems," in *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2008, pp. 157–172.

[9] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[10] B. Bonakdarpour and B. Finkbeiner, "Program repair for hyperproperties," in *Proceedings of the 17th Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2019, pp. 423–441.

[11] B. Bonakdarpour, C. Sánchez, and G. Schneider, "Monitoring hyperproperties by combining static analysis and runtime verification," in *Proceedings of the 8th Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, 2018, pp. 8–27.

[12] B. Bonakdarpour and B. Finkbeiner, "The complexity of monitoring hyperproperties," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 162–174.

[13] N. Brett, U. Siddique, and B. Bonakdarpour, "Rewriting-based runtime verification for alternation-free HyperLTL," in *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2017, pp. 77–93.

[14] P. Cañones, B. Köpf, and J. Reineke, "Security analysis of cache replacement policies," in *International Conference on Principles of Security and Trust*. Springer, 2017, pp. 189–209.

[15] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, 1988.

[16] J. Chen, Y. Feng, and I. Dillig, "Precise detection of side-channel vulnerabilities using quantitative cartesian hoare logic," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 875–890.

[17] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *Journal of Computer Security*, vol. 18, no. 6, pp. 1157–1210, 2010.

[18] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *Principles of Security and Trust*, 2014, vol. 8414, pp. 265–284.

[19] N. Coenen, B. Finkbeiner, C. Sánchez, and L. Tentrup, "Verifying hyperliveness," in *Proceedings of the 31st International Conference on Computer Aided Verification (CAV)*, 2019, pp. 121–139.

[20] CPSL@Duke, https://gitlab.oit.duke.edu/cpsl/hpctls, 2020.

[21] C. Dehnert, S. Junges, J. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Proc. CAV 2017*, ser. LNCS, vol. 10427, 2017, pp. 592–600.

[22] R. Dimitrova, B. Finkbeiner, and H. Torfah, "Probabilistic hyperpropertiesof markov decision processes," in *Proceedings of the 18th Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2020, to appear.

[23] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, Aug. 2014.

[24] H. Enderton and H. B. Enderton, *A mathematical introduction to logic*. Elsevier, 2001.

[25] B. Finkbeiner, C. Hahn, and T. Hans, "MGHyper: Checking satisfiability of HyperLTL formulas beyond the $\exists^*\forall^*$ fragment," in *Proceedings of the 16th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2018, pp. 521–527.

[26] B. Finkbeiner, C. Hahn, P. Lukert, M. Stenger, and L. Tentrup, "Synthesizing reactive systems from hyperproperties," in *Proceedings of the 30th International Confer ence on Computer Aided Verification (CAV)*, 2018, pp. 289–306.

[27] B. Finkbeiner, C. Hahn, and M. Stenger, "Eahyper: Satisfiability, implication, and equivalence checking of hyperproperties," in *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)*, 2017, pp. 564–570.

[28] B. Finkbeiner, C. Hahn, M. Stenger, and L. Tentrup, "RVHyper: A runtime verification tool for temporal hyperproperties," in *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2018, pp. 194–200.

[29] ——, "Monitoring hyperproperties," *Formal Methods in System Design (FMSD)*, vol. 54, no. 3, pp. 336–363, 2019.

[30] B. Finkbeiner, C. Hahn, and H. Torfah, "Model checking quantitative hyperproperties," in *Proceedings of the 30th International Conference on Computer Aided Verification*, 2018, pp. 144–163.

[31] B. Finkbeiner, C. Müller, H. Seidl, and E. Zalinescu, "Verifying security policies in multi-agent workflows with loops," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, 2017.

[32] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL*," in *Proceedings of the 27th International Conference on Computer Aided Verification (CAV)*, 2015, pp. 30–48.

[33] J. A. Goguen and J. Meseguer, "Security policies and security models," in *1982 IEEE Symposium on Security and Privacy*. IEEE, 1982, pp. 11–11.

[34] J. W. Gray III, "Probabilistic interference," in *Proceedings of the 1990 IEEE Symposium on Security and Privacy (S&P)*, 1990, pp. 170–179.

[35] ——, "Toward a mathematical foundation for information flow security," *Journal of Computer Security*, vol. 1, no. 3-4, pp. 255–294, May 1992.

[36] C. Hahn, M. Stenger, and L. Tentrup, "Constraint-based monitoring of hyperproperties," in *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2019, pp. 115–131.

[37] C. Hitchcock, "Probabilistic Causation," in *The Stanford Encyclopedia of Philosophy*, fall 2018 ed., 2018.

[38] B. Köpf and D. A. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2007, pp. 286–296.

[39] M. Z. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV)*, 2011, pp. 585–591.

[40] K. G. Larsen and A. Legay, "Statistical model checking: Past, present, and future," in *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, 2016, pp. 3–15.

[41] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification*, 2010, vol. 6418, pp. 122–135.

[42] N. Roohi, Y. Wang, M. West, G. E. Dullerud, and M. Viswanathan, "Statistical verification of the Toyota powertrain control verification benchmark," in *The 20th International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 65–70.

[43] A. Sabelfeld and D. Sands, "Probabilistic noninterference for multi-threaded programs," in *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW)*, 2000, pp. 200–214.

[44] K. Sen, M. Viswanathan, and G. Agha, "Statistical model checking of black-box probabilistic systems," in *Computer Aided Verification*, 2004, pp. 202–215.

[45] ——, "On statistical model checking of stochastic systems," in *Computer Aided Verification*, 2005, vol. 3576, pp. 266–280.

[46] G. Smith, "On the foundations of quantitative information flow," in *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures (FOSSACS)*, 2009, pp. 288–302.

[47] S. Stucki, C. Sánchez, G. Schneider, and B. Bonakdarpour, "Graybox monitoring of hyperproperties," in *Proceedings of the 23rd International Symposium on Formal Methods (FM)*, 2019, pp. 406–424.

[48] S. Tizpaz-Niari, P. Cerny, and A. Trivedi, "Data-driven debugging for functional side channels," *arXiv preprint arXiv:1808.10502*, 2018.

[49] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. pp. 117–186, 1945.

[50] Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic, "Statistical verification of hyperproperties for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

[51] H. L. S. Younes, "Ymer: A statistical model checker," in *Computer Aided Verification*, 2005, pp. 429–433.

[52] M. Zarei, Y. Wang, and M. Pajic, "Statistical verification of learning-based cyber-physical systems," in *International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–7.

[53] P. Zuliani, "Statistical model checking for biological applications," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 527–536, 2015.

## APPENDIX

We recap the temporal logics relevant to this paper with the notations adapted to that of HyperPCTL$^*$, and introduce the basics on the statistical model checking of PCTL$^*$ using the sequential probability ratio test (SPRT).

### A. PCTL$^*$

*Syntax:* The syntax of PCTL$^*$ [9] consists of *state* formulas $\Phi$ and *path* formulas $\varphi$ that are defined respectively over the set of atomic propositions AP by:

$$\Phi ::= \ \mathsf{a} \mid \neg\Phi \mid \Phi \wedge \Phi \mid \ \mathbb{P}^J(\varphi)$$

and

$$\varphi ::= \ \Phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U}^{\leq k}\, \varphi$$

where $\mathsf{a} \in \mathsf{AP}$, and $J \subseteq [0,1]$ is an interval with rational bounds.

*Semantics:* The satisfaction relation $\models$ of the PCTL$^*$ state and path formulas is defined for a state and a path of a labeled DTMC $\mathcal{M}$ respectively by

$$\begin{array}{lll}
(\mathcal{M}, s) \models \mathsf{a} & \text{iff} & \mathsf{a} \in L(s) \\
(\mathcal{M}, s) \models \neg\Phi & \text{iff} & (\mathcal{M}, s) \not\models \Phi \\
(\mathcal{M}, s) \models \Phi_1 \wedge \Phi_2 & \text{iff} & (\mathcal{M}, s) \models \Phi_1 \text{ and } (\mathcal{M}, s) \models \Phi_2 \\
(\mathcal{M}, s) \models \mathbb{P}^J(\varphi) & \text{iff} & \mathbf{Pr}\big((\mathcal{M}, s) \models \varphi\big) \in J
\end{array}$$

and

$$\begin{array}{lll}
(\mathcal{M}, S) \models \Phi & \text{iff} & (\mathcal{M}, S(0)) \models \Phi \\
(\mathcal{M}, S) \models \neg\varphi & \text{iff} & (\mathcal{M}, S) \not\models \varphi \\
(\mathcal{M}, S) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (\mathcal{M}, S) \models \varphi_1 \text{ and } (\mathcal{M}, S) \models \varphi_2 \\
(\mathcal{M}, S) \models \bigcirc\varphi & \text{iff} & (\mathcal{M}, S^{(1)}) \models \varphi \\
(\mathcal{M}, S) \models \varphi_1\, \mathcal{U}^{\leq k}\, \varphi_2 & \text{iff} & \text{there exists } i \leq k \text{ such that} \\
 & & \big((\mathcal{M}, S^{(i)}) \models \varphi_2\big) \wedge \\
 & & \big(\text{for all } j < i, (\mathcal{M}, S^{(j)}) \models \varphi_1\big)
\end{array}$$

where $S^{(i)}$ is the $i$-suffix of path $S$.

### B. HyperLTL

*Syntax:* HyperLTL [18] formulas are defined over the set of atomic propositions AP respectively by:

$$\psi ::= \ \exists\pi.\ \psi \mid \forall\pi.\ \psi \mid \varphi$$

and

$$\varphi ::= \ \mathsf{a}^\pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U}\, \varphi$$

where $\mathsf{a} \in \mathsf{AP}$.

*Semantics:* The semantics of HyperLTL is defined for a trace assignment $V : \Pi \to (\mathbb{N} \to 2^{\mathsf{AP}})$ by:

$$\begin{array}{lll}
V \models \mathsf{a}^\pi & \text{iff} & \mathsf{a} \in L(V(\pi(0))) \\
V \models \exists\pi.\ \psi & \text{iff} & \text{there exists } S \in T \\
 & & \text{such that } V[\pi \mapsto S] \models \psi \\
V \models \forall\pi.\ \psi & \text{iff} & \text{for all } S \in T \\
 & & \text{such that } V[\pi \mapsto S] \models \psi \\
V \models \neg\varphi & \text{iff} & V \not\models \varphi \\
V \models \varphi_1 \wedge \varphi_2 & \text{iff} & V \models \varphi_1 \text{ and } V \models \varphi_2 \\
V \models \bigcirc\varphi & \text{iff} & V^{(1)} \models \varphi \\
V \models \varphi_1\, \mathcal{U}\, \varphi_2 & \text{iff} & \text{there exists } i \geq 0 \text{ such that} \\
 & & \big(T, V^{(i)} \models \varphi_2\big) \wedge \\
 & & \big(\text{for all } j < i, \text{ we have } V^{(j)} \models \varphi_1\big)
\end{array}$$

where $V^{(i)}$ is the $i$-shift of path assignment $V$, defined by $V^{(i)}(\pi) = (V(\pi))^{(i)}$.

### C. HyperPCTL

*Syntax:* HyperPCTL [5] formulas are defined over the set of atomic propositions AP respectively by:

$$\psi ::= \ \mathsf{a}_\sigma \mid \exists\sigma.\ \psi \mid \forall\sigma.\ \psi \mid \neg\psi \mid \psi \wedge \psi \mid p \bowtie p$$

$$p ::= \ \mathbb{P}(\varphi) \mid c \mid p + p \mid p - p \mid p \cdot p$$

$$\varphi ::= \ \bigcirc\psi \mid \psi\, \mathcal{U}^{\leq k}\, \psi$$

where $\mathsf{a} \in \mathsf{AP}$, $c \in \mathbb{Q}$ and $\bowtie \in \{<, >, \leq, \geq, =\}$.

*Semantics:* The satisfaction relation $\models$ of HyperPCTL is defined for state and path formulas of a labeled DTMC $\mathcal{M}$ respectively by:

$$\begin{array}{lll}
(\mathcal{M}, X) \models \mathsf{a}_\sigma & \text{iff} & \mathsf{a} \in X(\sigma) \\
(\mathcal{M}, X) \models \exists\sigma.\ \psi & \text{iff} & \text{there exists } s \in \mathcal{S} \\
 & & \text{such that } X[\sigma \mapsto s] \models \psi \\
(\mathcal{M}, X) \models \forall\sigma.\ \psi & \text{iff} & \text{for all } s \in \mathcal{S} \\
 & & \text{such that } X[\sigma \mapsto s] \models \psi \\
(\mathcal{M}, X) \models \neg\psi & \text{iff} & (\mathcal{M}, X) \not\models \psi \\
(\mathcal{M}, X) \models \psi_1 \wedge \psi_2 & \text{iff} & (\mathcal{M}, X) \models \psi_1 \\
 & & \text{and } (\mathcal{M}, X) \models \psi_2 \\
(\mathcal{M}, X) \models p_1 \bowtie p_2 & \text{iff} & [\![p_1]\!]_{(\mathcal{M},X)} \bowtie [\![p_1]\!]_{(\mathcal{M},X)}
\end{array}$$

$$
\begin{array}{lcl}
[\![c]\!]_{(\mathcal{M},X)} & = & c \\
[\![p_1 + p_2]\!]_{(\mathcal{M},X)} & = & [\![p_1]\!]_{(\mathcal{M},X)} + [\![p_2]\!]_{(\mathcal{M},X)} \\
[\![p_1 - p_2]\!]_{(\mathcal{M},X)} & = & [\![p_1]\!]_{(\mathcal{M},X)} - [\![p_2]\!]_{(\mathcal{M},X)} \\
[\![p_1 \cdot p_2]\!]_{(\mathcal{M},X)} & = & [\![p_1]\!]_{(\mathcal{M},X)} \cdot [\![p_2]\!]_{(\mathcal{M},X)} \\
[\![\mathbb{P}(\varphi)]\!]_{(\mathcal{M},X)} & = & \mathbf{Pr}\big\{(S_i \in \mathsf{Paths}(X(\pi_i))_{i\in[n]} \mid \\
 & & \quad (M,\underline{S}) \models \varphi\big\}
\end{array}
$$

$$
\begin{array}{lll}
(M,\underline{S}) \models \bigcirc\varphi & \text{iff} & (M,\underline{S}(1)) \models \varphi \\
(M,\underline{S}) \models \varphi_1\, \mathcal{U}^{\leq k}\, \varphi_2 & \text{iff} & \text{there exists } i \leq k \text{ such that} \\
 & & \big((M,\underline{S}(i)) \models \varphi_2\big) \wedge \\
 & & \big(\text{for all } j < i, (M,\underline{S}(i)) \models \varphi_1\big)
\end{array}
$$

where $\underline{S} = (S_1, \ldots, S_n)$ and $\underline{S}(i) = (S_1(i), \ldots, S_n(i))$.

### D. Statistical Model Checking of PCTL*

The key issue in the statistical model checking of PCTL* is to deal with the probabilistic operator by sampling [6], [40]. Specifically, consider the satisfaction of a PCTL* formula

$$(\mathcal{M}, s) \models \mathbb{P}^{[0,p]}(\varphi) \tag{47}$$

where $\varphi$ is a linear temporal logic formula and $p \in (0,1)$ is a given real number. From the semantics of PCTL* in Appendix A, it means that the satisfaction probability

$$p_\varphi = \mathbf{Pr}\big((\mathcal{M}, s) \models \varphi\big) \tag{48}$$

of $\varphi$ for a given model $\mathcal{M}$ with the initial state $s$ satisfies

$$p_\varphi \leq p$$

For simplicity, we assume $\varphi$ is bounded-time and contains no probabilistic operator, thus its truth value can be decided on finite-length sample paths of $\mathcal{M}$. Unbounded-time non-nested formulas can be handled similarly with extra considerations.

To statistically infer (47), the assumption of an *indifference region* is usually adopted [6], [40]. That is, there exists $\varepsilon > 0$, such that the satisfaction probability $p_\varphi$ from (48) satisfies

$$p_\varphi \notin (p - \varepsilon, p + \varepsilon), \tag{49}$$

where $(p-\varepsilon, p+\varepsilon) \subseteq [0,1]$. The interval $(p-\varepsilon, p+\varepsilon)$ is commonly referred to as the indifference region. By the assumption (49), to statistically model check (47), it suffices to consider the hypothesis testing (HT) problem

$$\begin{array}{l} H_0 : p_\varphi \leq p - \varepsilon, \\ H_1 : p_\varphi \geq p + \varepsilon. \end{array} \tag{50}$$

and infer whether $H_0$ or $H_1$ holds by sampling.

The HT problem (50) is *composite*, since it contains (infinitely) many *simple* HT problems:

$$\begin{array}{l} H_0^{p_0} : p_\varphi = p_0, \\ H_1^{p_1} : p_\varphi = p_1. \end{array} \tag{51}$$

where $p_0$ and $p_1$ can take values from $[0, p-\varepsilon]$ and $[p+\varepsilon, 1]$, respectively. Intuitively, among all the possible values of $p_0$ and $p_1$, the following is the most "indistinguishable":

$$\begin{array}{l} H_0^{p-\varepsilon} : p_\varphi = p - \varepsilon, \\ H_1^{p+\varepsilon} : p_\varphi = p + \varepsilon. \end{array} \tag{52}$$

(We will discuss the meaning of "indistinguishable" later.)

To solve the hypothesis testing problem (52), we draw statistically independent sample paths $S_1, S_2, \ldots$ from the given model $\mathcal{M}$ with the initial state $s$. For $N$ such samples, the log-likelihood of observing $N$ such sample paths under the two hypotheses $H_0^{p-\varepsilon}$ and $H_1^{p+\varepsilon}$, are respectively $\lambda(p-\varepsilon)$ and $\lambda(p+\varepsilon)$, where

$$\lambda_{N,T}(p) = \ln\big(p^T(1-p)^{N-T}\big). \tag{53}$$

Accordingly, the log-likelihood ratio of the two hypotheses is

$$\Lambda_{N,T}(p+\varepsilon, p-\varepsilon) = \lambda(p+\varepsilon) - \lambda(p-\varepsilon). \tag{54}$$

Clearly, as $\Lambda_{N,T}(p+\varepsilon, p-\varepsilon)$ increases, $H_1^{p+\varepsilon}$ is more likely to be true, and the less statistical error is made when we assert $H_1^{p+\varepsilon}$ is true; and *vice versa*.

The sequential probability ratio test (SPRT) explicitly tells us how to make these assertions from $\Lambda_{N,T}$ to achieve certain levels of statistical errors [49]. The statistical errors are formally given by the probability of falsely asserting $H_1^{p+\varepsilon}$ while $H_0^{p-\varepsilon}$ holds, and the probability of falsely asserting $H_0^{p-\varepsilon}$ while $H_1^{p+\varepsilon}$ holds:

$$\begin{aligned} \alpha_{\mathrm{FP}} &= \mathbf{Pr}\big(\text{SPRT assert } H_1^{p+\varepsilon} \mid H_0^{p-\varepsilon} \text{ is true}\big), \\ \alpha_{\mathrm{FN}} &= \mathbf{Pr}\big(\text{SPRT assert } H_0^{p-\varepsilon} \mid H_1^{p+\varepsilon} \text{ is true}\big), \end{aligned}$$

where $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$ are called *false positive* (FP) and *false negative* (FN) ratios. When $\alpha_{\mathrm{FP}} = \alpha_{\mathrm{FN}}$, we may also refer to them as the *significance level*.

To achieve the given desired $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$, the SPRT is implemented *sequential* – i.e., it continuously draws samples until the following condition is satisfied:

$$\begin{cases} \text{assert } H_0^{p-\varepsilon}, & \text{if } \Lambda_{N,T}(p+\varepsilon, p-\varepsilon) < \ln\frac{\alpha_{\mathrm{FP}}}{1-\alpha_{\mathrm{FN}}} \\ \text{assert } H_1^{p+\varepsilon}, & \text{if } \Lambda_{N,T}(p+\varepsilon, p-\varepsilon) > \ln\frac{1-\alpha_{\mathrm{FP}}}{\alpha_{\mathrm{FN}}} \end{cases} \tag{55}$$

It can be proved that this SPRT algorithm always terminates with probability 1 and it strictly achieves the desired $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$ [49].

Finally, we discuss the "indistinguishability". Suppose the true satisfaction probability satisfies $p_\varphi \in [p+\varepsilon, 1]$. (The case $p_\varphi \in [0, p-\varepsilon]$ is similar.) Following (55), if the SPRT asserts $H_1^{p+\varepsilon}$ for certain $N$ and $T$, then we have that

$$\Lambda_{N,T}(p_\varphi, p-\varepsilon) > \ln\frac{1-\alpha_{\mathrm{FP}}}{\alpha_{\mathrm{FN}}}.$$

This means that if $N$ and $T$ are sufficient to assert $H_1^{p+\varepsilon}$ against $H_0^{p-\varepsilon}$ with the desired $\alpha_{\mathrm{FP}}$ and $\alpha_{\mathrm{FN}}$, then they are sufficient to assert $H_1^{p_\varphi}$ against $H_0^{p-\varepsilon}$. In other words, $p+\varepsilon$ is the worst case. Similarly, if the SPRT asserts $H_0^{p-\varepsilon}$, then we can show that

$$\Lambda_{N,T}(p_\varphi, p-\varepsilon) < \ln\frac{\alpha_{\mathrm{FP}}}{1-\alpha_{\mathrm{FN}}},$$

and the same argument follows. For more detailed discussions, please refer to [44].