HyperPCTL: A Temporal Logic for Probabilistic Hyperproperties

Erika Ábrahám[1] and Borzoo Bonakdarpour[2]

[1] RWTH Aachen University, Germany
[2] Iowa State University, USA

Abstract. In this paper, we propose a new temporal logic for expressing and reasoning about probabilistic hyperproperties. *Hyperproperties* characterize the relation between different independent executions of a system. *Probabilistic* hyperproperties express quantitative dependencies between such executions. The standard temporal logics for probabilistic systems, i.e., PCTL and PCTL* can refer only to a single path at a time and, hence, cannot express many probabilistic hyperproperties of interest. The logic proposed in this paper, HyperPCTL, adds explicit and simultaneous quantification over multiple traces to PCTL. Such quantification allows expressing probabilistic hyperproperties. A model checking algorithm for the proposed logic is also introduced for discrete-time Markov chains.

1 Introduction

Four decades ago, Lamport [16] used the notion of *trace properties* as a means to specify the correctness of individual executions of concurrent programs. This notion was later formalized and classified by Alpern and Schneider [1] to *safety* and *liveness* properties. Temporal logics (e.g., LTL [17] and CTL [3]) were built based on these efforts to give formal syntax and semantics to requirements of trace properties. Subsequently, verification algorithms were developed to reason about individual traces of a system.

It turns out that many interesting requirements are not trace properties. For example, important information-flow security policies such as *noninterference*[3] [10] and *observational determinism*[4] [22] cannot be expressed as properties of individual execution traces of a system. Also, service level agreement requirements (e.g., mean response time and percentage uptime) that use statistics of a system across all executions of a system are not trace properties. Rather, they are properties of sets of execution traces, also known as *hyperproperties* [5]. Temporal logics HyperLTL and HyperCTL* [4] have been proposed to provide a unifying

[3] Noninterference stipulates that input commands from high-privileged users have no effect on the system's behavior observed by low-privileged observers.

[4] Observational determinism requires that two executions that start at two low initial states appear deterministic to a low user.

framework to express and reason about hyperproperties. They allow explicit and simultaneous quantification over multiple paths to LTL and to CTL*.

Hyperproperties can also be probabilistic. Such *probabilistic hyperproperties* generally express probabilistic relations between independent executions of a system. For example, in information-flow security, adding probabilities is motivated by establishing a connection between information theory and information flow across multiple traces. It is also motivated by using probabilistic schedulers, which opens up an opportunity for the attacker to set up a probabilistic *covert channel*, whereby information is obtained by statistical inferences drawn from the relative frequency of outcomes of a repeated computation. Policies that defend against such an attempt, known as *probabilistic noninterference*, stipulate that the probability of every low-observable trace be the same for every low-equivalent initial state. Such policies quantify on different execution traces and the probability of reaching certain states in the independent and simultaneous executions.

Consider the following classic example [21] comprising of two threads th and th':

$$th: \ \textbf{while}\ h > 0\ \textbf{do}\ \{h \leftarrow h - 1\};\ l \leftarrow 2 \qquad || \qquad th': \ l \leftarrow 1$$

where h is an input by a high-privileged user and l is an output observable by low-privileged users. Probabilistic noninterference would require that l obtains values of 1 and 2 with equal probabilities, regardless of the initial value of h. However, assuming that the scheduler chooses to execute atomic statements of the threads th and th' iteratively with uniform probability distribution, the likely outcome of the race between the two assignments $l \leftarrow 1$ and $l \leftarrow 2$ depends on the initial value of h: the larger the initial value of h, the greater the probability that the final value of l is 2. For example, if the initial value of h is 0 in one execution, then the final value of l is 1 with probability $1/4$ and 2 with probability $3/4$, but for the initial value $h = 5$ in another independent execution we can observe the final value $l = 1$ with probability $1/4096$ and $l = 2$ with probability $4095/4096$. Thus, it holds that for two independent executions with initial h values 0 resp. 5 the larger h value leads to a lower probability for $l = 1$ upon termination. I.e., this program does not satisfy probabilistic noninterference.

It is straightforward to observe that requirements such as probabilistic noninterference cannot be expressed in existing probabilistic temporal logics such as PCTL [12] and PCTL*, as they cannot draw connection between the probability of reaching certain states in independent executions. Also, introducing probability operators to HyperLTL is not quite natural, as the semantics of HyperLTL is trace-based and probabilistic logics are branching-time in nature. Moreoever, introducing probability operators to HyperCTL* cannot be done trivially. With this motivation, in this paper, we propose the temporal logic HyperPCTL that lifts PCTL by allowing explicit quantification over initial states and, hence, multiple computation trees simultaneously, as well as probability of occurring propositions that stipulate relationships among those traces. For the above example, the following HyperPCTL formula expresses probabilistic noninterference, which obviously does not hold:

$$\forall\sigma.\forall\sigma'.\Big(\mathit{init}_\sigma \wedge \mathit{init}_{\sigma'} \wedge h_\sigma \neq h_{\sigma'}\Big) \Rightarrow \Big(\ \Big(\mathbb{P}\Diamond(\mathit{fin}_\sigma \wedge (l{=}1)_\sigma) = \mathbb{P}\Diamond(\mathit{fin}_{\sigma'} \wedge (l{=}1)_{\sigma'})\Big) \wedge \Big(\mathbb{P}\Diamond(\mathit{fin}_\sigma \wedge (l{=}2)_\sigma) = \mathbb{P}\Diamond(\mathit{fin}_{\sigma'} \wedge (l{=}2)_{\sigma'})\Big)\Big)$$

That is, for any two executions from initial states σ and σ' (i.e., initial values of h), the probability distribution of terminating with value $l = 1$ (or $l = 2$) is uniform.

In addition to probabilistic noninterference, we show that HyperPCTL can express other important requirements and policies, some not related to information-flow security. First, we show that HyperPCTL subsumes probabilistic bisimulation. We also show that HyperPCTL can express requirements such as differential privacy, quantitative information flow, and probabilistic causation (a.k.a. causality). We also present a HyperPCTL model checking algorithm for discrete-time Markov chains (DTMCs). The complexity of the algorithm is polynomial-time in the size of the input DTMC and is PSPACE-hard in the size of the input HyperPCTL formula. We also discuss a wide range of open problem to be tackled by future research. We believe that this paper opens a new area in rigorous analysis of probabilistic systems.

Organization The rest of the paper is organized as follows. Section 2 defines the syntax and semantics of HyperPCTL. Section 3 provides a diverse set of example requirements that HyperPCTL can express. We present our model checking algorithm in Section 4. Related work is discussed in Section 5. Finally, we make concluding remarks and discuss future work in Section 6.

2 HyperPCTL

We assume the systems to be described by HyperPCTL formulas to be modeled as discrete-time Markov chains.

Definition 1. *A* (discrete-time) Markov chain (DTMC) $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ *is a tuple with the following components:*

- S *is a finite nonempty set of* states*,*
- $\mathbf{P} : S \times S \to [0, 1]$ *is a* transition probability function *with* $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ *for all states* $s \in S$*,*
- AP *is a set of* atomic propositions*, and*
- $L : S \to 2^{\mathsf{AP}}$ *is a* labeling function*.* ■

A *path* of a Markov chain $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ is defined as an infinite sequence $\pi = s_0 s_1 s_2 \cdots \in S^\omega$ of states with $\mathbf{P}(s_i, s_{i+1}) > 0$, for all $i \geq 0$; we write $\pi[i]$ for s_i. Let $\mathit{Paths}^s(\mathcal{M})$ denote the set of all (infinite) paths starting in s in $\mathcal{M}$, and $\mathit{Paths}^s_{\mathit{fin}}(\mathcal{M})$ denote the set of all finite prefixes of paths from $\mathit{Paths}^s(\mathcal{M})$, which we sometimes call *finite paths*.

2.1 HyperPCTL Syntax

To be able to express probabilistic hyperproperties of DTMCs, the syntax of HyperPCTL differs from computation tree logic (CTL) in two different aspects. Firstly, CTL quantification over paths starting in a given state is replaced by a probability operator expressing the probability that a certain property holds on the paths starting in a given state; this extension is similar to probabilistic computation tree logic (PCTL), but whereas PCTL allows only the comparison of these probabilities to constant thresholds, we allow the arbitrary usage of such probabilities in arithmetic constraints. Secondly, we add quantification over states to express hyperproperties; note that whereas HyperCTL* extends CTL* by path quantification, in the probabilistic setting the argumentation moves from paths to the probabilities of paths, which are determined in the context of states (where the paths start).

HyperPCTL *state formulas* are inductively defined by the following grammar:

$$\psi ::= \forall\sigma.\psi \;\Big|\; \exists\sigma.\psi \;\Big|\; \texttt{true} \;\Big|\; a_\sigma \;\Big|\; \psi \wedge \psi \;\Big|\; \neg\psi \;\Big|\; p \sim p$$

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

where $c \in \mathbb{Q}$, $a \in \mathsf{AP}$ is an atomic proposition, $\sim \in \{<, \leq, =, \geq, >\}$, σ is a *state variable* from a countably infinite supply of variables $\mathcal{V} = \{\sigma_1, \sigma_2, \ldots\}$, p is a *probability expression*, and φ is a *path formula*. HyperPCTL path formulas are formed according to the following grammar:

$$\varphi \;::=\; \bigcirc\psi \;\Big|\; \psi \,\mathcal{U}\, \psi \;\Big|\; \psi \,\mathcal{U}^{[k_1,k_2]}\, \psi$$

where ψ is a state formula and $k_1, k_2 \in \mathbb{N}_{\geq 0}$ with $k_1 \leq k_2$.

As syntactic sugar, we introduce state formulas of the form $p \in J$, where $J = [l, u] \subseteq [0, 1]$ is an interval with rational bounds, defined as $l \leq p \wedge p \leq u$. We also define the syntactic sugar $\psi_1 \,\mathcal{U}^{\leq k}\, \psi_2$ for $\psi \,\mathcal{U}^{[0,k]}\, \psi$. As usual, we furthermore introduce $\psi_1 \vee \psi_2 = \neg(\neg\psi_1 \wedge \neg\psi_2)$, $\Diamond\psi = \texttt{true}\,\mathcal{U}\,\psi$, $\Diamond^{[k_1,k_2]}\psi = \texttt{true}\,\mathcal{U}^{[k_1,k_2]}\,\psi$, $\mathbb{P}(\Box\psi) = 1 - \mathbb{P}(\Diamond\neg\psi)$, and $\mathbb{P}(\Box^{[k_1,k_2]}\,\psi) = 1 - \mathbb{P}(\Diamond^{[k_1,k_2]}\,\neg\psi)$. We denote by $\mathcal{F}$ the set of all HyperPCTL state formulas.

An occurrence of an indexed atomic proposition a_σ in a HyperPCTL state formula ψ is *free* if it is not in the scope of a quantifier bounding σ and otherwise *bound*. HyperPCTL *sentences* are HyperPCTL state formulas in which all occurrences of all indexed atomic propositions are bound. HyperPCTL (*quantified*) *formulas* are HyperPCTL sentences.

Example Consider the following formula:

$$\forall\sigma_1.\exists\sigma_2.\mathbb{P}(\Diamond\, a_{\sigma_1}) = \mathbb{P}(\Diamond\, b_{\sigma_2}).$$

This formula holds if for each instantiated state s_1, there exists another instantiated state s_2, such that the probability to finally reach a state labeled with a from s_1 equals the probability of reaching b from s_2.

2.2 HyperPCTL Semantics

We present the semantics of HyperPCTL based on n-ary *self-composition* of a DTMC. We emphasize that it is possible to define the semantics in terms of the non-self-composed DTMC, but it will essentially result in a very similar setting, but more difficult to understand.

Definition 2. *The* n-ary self-composition *of a DTMC* $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ *is a DTMC* $\mathcal{M}^n = (S^n, \mathbf{P}^n, \mathsf{AP}^n, L^n)$ *with*

- $S^n = S \times \ldots \times S$ *is the n-ary Cartesian product of* S,
- $\mathbf{P}^n\big(s, s'\big) = \mathbf{P}\big(s_1, s'_1\big) \cdot \ldots \cdot \mathbf{P}(s_n, s'_n)$ *for all* $s = (s_1, \ldots, s_n) \in S^n$ *and* $s' = (s'_1, \ldots, s'_n) \in S^n$,
- $\mathsf{AP}^n = \cup_{i=1}^n \mathsf{AP}_i$, *where* $\mathsf{AP}_i = \{a_i \,|\, a \in \mathsf{AP}\}$ *for* $i \in [1, n]$, *and*
- $L^n(s) = \cup_{i=1}^n L_i(s_i)$ *for all* $s = (s_1, \ldots, s_n) \in S^n$ *with* $L_i(s_i) = \{a_i \,|\, a \in L(s_i)\}$ *for* $i \in [1, n]$. ■

The satisfaction of a HyperPCTL quantified formula by a DTMC $\mathcal{M}$=$(S, \mathbf{P}, \mathsf{AP}, L)$ is defined by:

$$\mathcal{M} \models \psi \qquad \textit{iff} \qquad \mathcal{M}, () \models \psi$$

where () is the empty sequence of states. Thus, the satisfaction relation $\models$ defines the values of HyperPCTL quantified, state, and path formulas in the context of a DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ and an n-tuple $s = (s_1, \ldots, s_n) \in S^n$ of states (which is () for $n = 0$). Intuitively, the state sequence s stores instantiations for quantified state variables. Remember that HyperPCTL quantified formulas are sentences. The semantics evaluates HyperPCTL formulas by structural recursion. Quantifiers are instantiated and the instantiated values for state variables are stored in the state sequence s. To maintain the connection between a state in this sequence and the state variable which it instantiates, we introduce the auxiliary syntax a_i with $a \in \mathsf{AP}$ and $i \in \mathbb{N}_{>0}$, and if we instantiate σ in $\exists\sigma.\psi$ or $\forall\sigma.\psi$ by state s, then we append s at the end of the state sequence and replace all a_σ that is bound by the given quantifier by a_i with i being the index of s in the state sequence. We will express the meaning of path formulas based on the n-ary self-composition of $\mathcal{M}$; the index i for the instantiation of σ also fixes the component index in which we keep track of the paths starting in σ. The semantics judgment rules to evaluate formulas in the context of a DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ and an

n-tuple $s = (s_1, \ldots, s_n) \in S^n$ of states are the following:

$$
\begin{array}{lll}
\mathcal{M}, s \models \forall\sigma.\psi & \textit{iff} & \forall s_{n+1} \in S.\ \mathcal{M}, (s_1, \ldots, s_n, s_{n+1}) \models \psi[\mathsf{AP}_{n+1}/\mathsf{AP}_\sigma] \\
\mathcal{M}, s \models \exists\sigma.\psi & \textit{iff} & \exists s_{n+1} \in S.\ \mathcal{M}, (s_1, \ldots, s_n, s_{n+1}) \models \psi[\mathsf{AP}_{n+1}/\mathsf{AP}_\sigma] \\
\mathcal{M}, s \models \texttt{true} & & \\
\mathcal{M}, s \models a_i & \textit{iff} & a \in L(s_i) \\
\mathcal{M}, s \models \psi_1 \wedge \psi_2 & \textit{iff} & \mathcal{M}, s \models \psi_1 \textit{ and } \mathcal{M}, s \models \psi_2 \\
\mathcal{M}, s \models \neg\psi & \textit{iff} & \mathcal{M}, s \not\models \psi \\
\mathcal{M}, s \models p_1 \sim p_2 & \textit{iff} & [\![p_1]\!]_{\mathcal{M},s} \sim [\![p_2]\!]_{\mathcal{M},s} \\
[\![\mathbb{P}(\varphi)]\!]_{\mathcal{M},s} & = & Pr\{\pi \in Paths^s(\mathcal{M}^n) \mid \mathcal{M}, \pi \models \varphi\} \\
[\![c]\!]_{\mathcal{M},s} & = & c \\
[\![p_1 + p_2]\!]_{\mathcal{M},s} & = & [\![p_1]\!]_{\mathcal{M},s} + [\![p_2]\!]_{\mathcal{M},s} \\
[\![p_1 - p_2]\!]_{\mathcal{M},s} & = & [\![p_1]\!]_{\mathcal{M},s} - [\![p_2]\!]_{\mathcal{M},s} \\
[\![p_1 \cdot p_2]\!]_{\mathcal{M},s} & = & [\![p_1]\!]_{\mathcal{M},s} \cdot [\![p_2]\!]_{\mathcal{M},s}
\end{array}
$$

where ψ, ψ_1, and ψ_2 are HyperPCTL state formulas; the substitution $\psi[\mathsf{AP}_{n+1}/\mathsf{AP}_\sigma]$ replaces for each atomic proposition $a \in \mathsf{AP}$ each free occurrence of a_σ in ψ by a_{n+1}; $a \in \mathsf{AP}$ is an atomic proposition and $1 \leq i \leq n$; p_1 and p_2 are probability expressions and $\sim \in \{<, \leq, =, \geq, >\}$; φ is a HyperPCTL path formula and c is a rational constant.

The satisfaction relation for HyperPCTL path formulas is defined as follows, where π is a path of $\mathcal{M}^n$ for some $n \in \mathbb{N}_{>0}$; ψ, ψ_1, and ψ_2 are HyperPCTL state formulas and $k_1, k_2 \in \mathbb{N}_{\geq 0}$ with $k_1 \leq k_2$:

$$
\begin{array}{lll}
\mathcal{M}, \pi \models \bigcirc\psi & \textit{iff} & \mathcal{M}, \pi[1] \models \psi \\
\mathcal{M}, \pi \models \psi_1\,\mathcal{U}\,\psi_2 & \textit{iff} & \exists j \geq 0.\Big(\mathcal{M}, \pi[j] \models \psi_2 \wedge \forall i \in [0, j).\mathcal{M}, \pi[i] \models \psi_1\Big) \\
\mathcal{M}, \pi \models \psi_1\,\mathcal{U}^{\,[k_1,k_2]}\,\psi_2 & \textit{iff} & \exists j \in [k_1, k_2].\Big(\mathcal{M}, \pi[j] \models \psi_2 \wedge \\
 & & \qquad \forall i \in [0, j).\mathcal{M}, \pi[i] \models \psi_1\Big)
\end{array}
$$

Note that each HyperPCTL formula can be transformed into an equivalent formula in prenex normal form $Q_1\sigma_1 \ldots Q_n\sigma_n.\psi$, where each $Q_i \in \{\forall, \exists\}$ is a quantifier, σ_i is a state variable, and ψ is a quantifier-free HyperPCTL formula. Note furthermore that the semantics assures that each path formula φ is evaluated in the context of a path of $\mathcal{M}^n$ such that $1 \leq i \leq n$ for each a_i in φ.

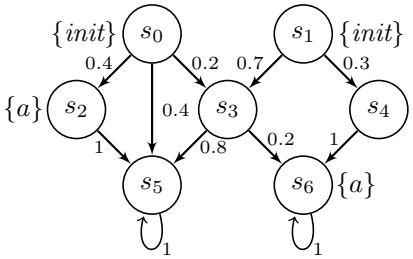

Fig. 1: Semantics example.

Example Consider the DTMC $\mathcal{M}$ in Fig. 1 and the following HyperPCTL formula:

$$\psi = \forall\sigma.\forall\sigma'.(init_\sigma \wedge init_{\sigma'}) \Rightarrow \Big(\mathbb{P}(\Diamond\, a_\sigma) = \mathbb{P}(\Diamond\, a_{\sigma'})\Big)$$

This formula is satisfied by $\mathcal{M}$ if for all pairs of initial states (labeled by the atomic proposition *init*), the probability to satisfy a is the same, i.e., for each

$(s_i, s_j) \in S^2$ with $\mathit{init} \in L(s_i)$ and $\mathit{init} \in L(s_j)$ it holds that $\mathcal{M}, (s_i, s_j) \models \mathbb{P}(\Diamond a_1) = \mathbb{P}(\Diamond a_2)$. The probability of reaching a from s_0 is $0.4 + (0.2 \times 0.2) = 0.44$. Moreover, the probability of reaching a from s_1 is $0.3 + (0.7 \times 0.2) = 0.44$. Hence, we have $\mathcal{M} \models \psi$.

3 HyperPCTL in Action

We now put HyperPCTL into action by formulating probabilistic requirements from different areas, such as information-flow security, privacy, and causality analysis.

3.1 Probabilistic Bisimulation

A *bisimulation* is an equivalence relation over a set of states of a system such that equivalent states cannot be distinguished by observing their behaviors. In the context of DTMC states and PCTL properties, a *probabilistic* bisimulation is an equivalence relation over the DTMC states such that any two equivalent states satisfy the same PCTL formulas. The latter property can be assured inductively by requiring that equivalent states have the same labels and the probability to move from them to any of the equivalence classes is the same.

Assume a partitioning $S_1, \ldots, S_k$ of S with $\cup_{i=1}^{k} S_i = S$ and $S_i \cap S_j = \emptyset$ for all $1 \leq i < j \leq k$. To express that the equivalence relation $R = \cup_{i=1}^{k} S_i \times S_i$ is a probabilistic bisimulation, we define $\mathcal{M}' = (S, \mathbf{P}, \mathsf{AP}', L')$ with $\mathsf{AP}' = \mathsf{AP} \cup \{a^1, \ldots, a^k\}$, where each a^i, for all $i \in [1, k]$, is a fresh atomic proposition not in AP, and for each $s \in S_i$, we set $L'(s) = L(s) \cup \{a^i\}$. The equivalence relation R is a bisimulation for $\mathcal{M}$ if $\mathcal{M}'$ satisfies the following HyperPCTL formula

$$\varphi_{\mathsf{pb}} = \forall\sigma.\forall\sigma'.\bigwedge_{i=1}^{k}\left[(a^i_\sigma \wedge a^i_{\sigma'}) \Rightarrow \left[\psi^{\mathsf{AP}} \wedge \bigwedge_{j=1}^{k} \mathbb{P}(\bigcirc a^j_\sigma) = \mathbb{P}(\bigcirc a^j_{\sigma'})\right]\right]$$

where $\psi^{\mathsf{AP}} = \bigwedge_{a \in \mathsf{AP}}(a_\sigma \Leftrightarrow a_{\sigma'})$.

3.2 Probabilistic Noninterfence

Noninterference is an information-flow security policy that enforces that a low-privileged user (e.g., an attacker) should not be able to distinguish two computations from their publicly observable outputs if they only vary in their inputs by a high-privileged user (e.g., a secret). *Probabilistic noninterference* [14] establishes 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. Probabilistic noninterference can be expressed in HyperPCTL as follows:

$$\varphi_{\mathsf{pni}} = \forall\sigma.\forall\sigma'.\Big(l_\sigma \wedge l_{\sigma'}\Big) \Rightarrow \Big(\mathbb{P}(\bigcirc l_\sigma) = \mathbb{P}(\bigcirc l_{\sigma'})\Big)$$

where l denotes a low-observable atomic proposition. Observe that formula φ_{pni} is a simplification of formula φ_{pb} in Section 3.1, but a stronger form of the noninterference formula for the example in Section 1. In fact, most approaches to prove probabilistic noninterference is by showing probabilistic bisimulation with respect to low-observable propositions.

3.3 Differential Privacy

Differential privacy [6] is a commitment by a data holder to a data subject (normally an individual) that he/she will not be affected by allowing his/her data to be used in any study or analysis. Formally, let ϵ be a positive real number and $\mathcal{A}$ be a randomized algorithm that makes a query to an input database and produces an output. Algorithm $\mathcal{A}$ is called *ϵ-differentially private*, if for all databases D_1 and D_2 that differ on a single element, and all subsets S of possible outputs of $\mathcal{A}$, we have:

$$Pr[\mathcal{A}(D_1) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(D_2) \in S].$$

Differential privacy can be expressed in HyperPCTL by the following formula:

$$\psi_{\mathsf{dp}} = \forall\sigma.\forall\sigma'.\Big[dbSim(\sigma,\sigma')\Big] \;\Rightarrow\; \Big[\mathbb{P}\Big(\diamondsuit(qOut \in S)_{\sigma}\Big) \leq e^{\epsilon} \cdot \mathbb{P}\Big(\diamondsuit(qOut \in S)_{\sigma'}\Big)\Big]$$

where $dbSim(\sigma,\sigma')$ means that two different dataset inputs have all but one similarity and $qOut$ is the result of the query. For example, one way to provide differential privacy is through *randomized response* in order to create noise and provide plausible deniability. Let A be an embarrassing or illegal activity. In a social study, each participant is faced with the query, "Have you engaged in activity A in the past week?" and is instructed to respond by the following protocol:

1. Flip a fair coin.
2. If tail, then answer truthfully.
3. If head, then flip the coin again and respond "Yes" if head and "No" if tail.

Thus, a "Yes" response may have been offered because the first and second coin flips were both heads. This implies that, there are no good or bad responses and an answer cannot be incriminating.

We now show that this social study is $(\ln 3)$-deferentially private. For each participant in the study, Fig. 2 shows the Markov chain of the response protocol, where $\{t{=}y\}$ (respectively, $\{t{=}n\}$) denotes that the truth is that the participant did (respectively, did not) engage in activity A, and $\{r{=}y\}$ (respectively, $\{r{=}n\}$) means that the participant responds "Yes" (respectively, "No"). The HyperPCTL

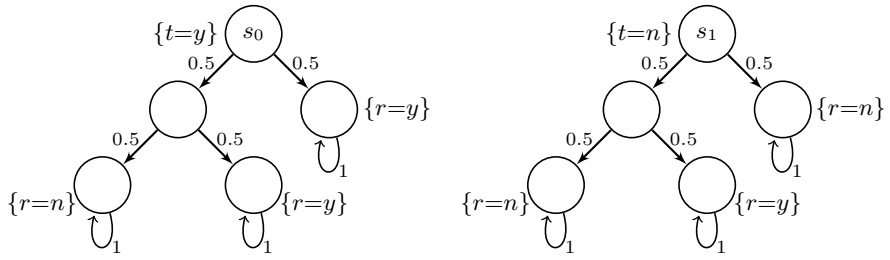

Fig. 2: Markov chain of the randomized response protocol.

formula to express $(\ln 3)$-deferentially privacy for this protocol is the following:

$$\forall\sigma.\forall\sigma'.\left[\left((t{=}n)_\sigma \wedge (t{=}y)_{\sigma'}\right) \Rightarrow \left(\mathbb{P}\Big(\Diamond(r{=}n)_\sigma\Big) \leq e^{\ln 3}\cdot\mathbb{P}\Big(\Diamond(r{=}n)_{\sigma'}\Big)\right)\right] \wedge$$
$$\left[\left((t{=}y)_\sigma \wedge (t{=}n)_{\sigma'}\right) \Rightarrow \left(\mathbb{P}\Big(\Diamond(r{=}y)_\sigma\Big) \leq e^{\ln 3}\cdot\mathbb{P}\Big(\Diamond(r{=}y)_{\sigma'}\Big)\right)\right]$$

Observe that compared to formula ψ_{dp}, we have decomposed $dbSim(\sigma,\sigma')$ to two cases of $t=y$ and $t=n$. Thus, in the left conjunct, the set S represents the case where the response is "No" and in the right conjunct, the set S represents the case where the response is "Yes". It is straightforward to see that the DTMC in Fig. 2 satisfies the formula, when for the left conjunct σ and σ' are instantiated by s_0 and s_1, respectively, and for the right conjunct σ and σ' are instantiated by s_1 and s_0, respectively.

3.4 Probabilistic Causation

Probabilistic causation [8] aims to characterize the relationship between *cause* and *effect* using the tools of probability theory. The reason for using probabilities is that most causes are not invariably followed by their effects. For example, smoking is a cause of lung cancer, even though some smokers do not develop lung cancer and some people who have lung cancer are not smokers. Thus, we need to somehow express that some causes are *more likely* to develop an effect. Specifically, the central idea in probabilistic causation is to assert that the probability of occurring effect e if cause c happens is higher than the probability of occurring e when c does not happen. We can express the most basic type of probabilistic causation in HyperPCTL as follows:

$$\psi_{\mathsf{pc}_1} = \forall\sigma.\forall\sigma'.c_\sigma \wedge \Big(\mathbb{P}(\Diamond\, e_\sigma) > \mathbb{P}(\neg c_{\sigma'}\ \mathcal{U}\, e_{\sigma'})\Big).$$

Observe that expressing causation in the standard PCTL by stripping the state quantifiers in formula ψ_{pc_1} will damage the meaning of causation. The resulting PCTL formula captures the causation relation from each initial state in isolation

and it wrongly allows the probability of $\Diamond\, e$ from one initial state to be less than the probability of $(\neg c\, \mathcal{U}\ e)$ from another initial state.

One problem with formula ψ_{pc_1} is spurious correlations. For example, if c is the drop in the level of mercury in a barometer, and e is the occurrence of a storm, then the above formula may hold in a system, though c is not really the cause of e. In fact, the real cause for both is the drop in atmospheric pressure. To address this problem, we add a constraint, where there should be no further event a that *screens off* e from c [18]:

$$\psi_{\mathsf{pc}_2} = \forall\sigma.\forall\sigma'.\neg\exists\sigma''.c_\sigma \wedge \Big(\mathbb{P}(\Diamond\, e_\sigma) > \mathbb{P}(\neg c_{\sigma'}\ \mathcal{U}\, e_{\sigma'})\Big) \wedge \bigwedge_{a\in\mathsf{AP}\setminus\{e,c\}} \Big[(a_{\sigma''} \wedge c_{\sigma''}) \wedge \Big(\mathbb{P}(\Diamond\, e_{\sigma''}) = \mathbb{P}(\Diamond\ e_\sigma)\Big)\Big].$$

The negation behind the existential quantifier can be pushed inside to obtain a proper HyperPCTL formula. We note that for simplicity, in formula ψ_{pc_2}, propositions a and c occur in the same state in σ''. A more general way is to allow a happen before or simultaneously with c. Finally, we note that other concepts in probabilistic causation such as Reichenbach's Common Cause Principle and Fork Asymmetry [18] (which emulates the second law of thermodynamics), as well as Skyrms's Background Contexts [20] can be expressed in a similar fashion.

4 HyperPCTL Model Checking

In the following, we show that the HyperPCTL model checking problem is decidable by introducing a model checking algorithm. The space complexity of our algorithm is exponential in the number of quantifiers of the input formula, because for n state quantifiers, we build the n-ary self-composition of the input DTMC. We are uncertain whether there exists a PSPACE algorithm, but we show the PSPACE-hardness of the problem.

Let $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ be a DTMC and ψ be a HyperPCTL quantified formula. Let furthermore n be the number of state quantifiers in ψ if it has any and let $n = 1$ otherwise. Informally, our model checking algorithm decides whether $\mathcal{M} \models \psi$ as follows (detailed pseudo-code is formulated in the Algorithms 1–3):

1. Apply variable renaming such that the quantified state variables are named $\sigma_1, \ldots, \sigma_n$.
2. Build the self-composition $\mathcal{M}^n$.
3. Compute a labeling $\hat{L}^n(s)$ for all states $s \in S^n$ of $\mathcal{M}^n$ as follows. Initially $\hat{L}^n(s) = \emptyset$ for all $s \in S^n$ (Line 5 in Algorithm 1). For all sub-formulas ψ' of ψ inside-out do the following:
   - If the subformula ψ' has the form `true`, add `true` to the label sets $\hat{L}^n(s)$ of all states $s \in S^n$ (Line 3 in Algorithm 2).
   - If the subformula ψ' is an atomic proposition a_{σ_i}, add a_{σ_i} to the label set of each state $s \in S^n$ with $a_i \in L^n(s)$ (Line 5 in Algorithm 2).

Algorithm 1: HyperPCTL model checking algorithm I

Input : DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$, HyperPCTL quantified formula ψ
Output : Whether $\mathcal{M} \models \psi$

```
1  Function main(M, ψ)
2      n := number of quantifiers in ψ
3      if n = 0 then
4          n:=1                          % n will be the arity of the self-composition
5      let L̂^n : S^n → 2^F with L̂^n(s) = ∅ for all s ∈ S^n
6      L̂^n := HyperPCTL(M, ψ, n, L̂^n)          % see Algorithm 2
7      if ψ ∈ L̂^n(s) for some s ∈ S^n then
8          return true
9      else
10         return false
```

- If the subformula ψ' is $\psi_1 \wedge \psi_2$, then add $\psi_1 \wedge \psi_2$ to $\hat{L}^n(s)$ for each $s \in S^n$ with $\psi_1 \in \hat{L}^n(s)$ and $\psi_2 \in \hat{L}^n(s)$ (Lines 6 – 9 in Algorithm 2).
- If the subformula ψ' is $\neg\psi_1$, then add $\neg\psi_1$ to $\hat{L}^n(s)$ for each $s \in S^n$ with $\psi_1 \notin \hat{L}^n(s)$ (Lines 10 – 12 in Algorithm 2).
- If the subformula ψ' is $p_1 \sim p_2$ (respectively $p \in J$), then compute for all $\mathbb{P}(\varphi)$ appearing in $p_1 \sim p_2$ (respectively, $p \in J$) for all states $s \in S^n$ the probability that φ holds in s using standard PCTL model checking, and add for all $s \in S^n$ the property $p_1 \sim p_2$ (respectively, $p \in J$) to $\hat{L}^n(s)$ if $p_1 \sim p_2$ (respectively, $p \in J$) evaluates to `true` in s (Lines 13 – 16 in Algorithm 2).
- If the subformula ψ' is of the form $\exists\sigma_i.\psi_1$, then label all states $s = (s_1, \ldots, s_n) \in S^n$ with $\exists\sigma_i.\psi_1$ iff there exists an $s'_i \in S$, such that $\psi_1 \in \hat{L}^n(s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$ (Lines 17 – 19 in Algorithm 2).
- If the subformula ψ' is of the form $\forall\sigma_i.\psi_1$, then label all states $s = (s_1, \ldots, s_n) \in S^n$ with $\forall\sigma_i.\psi_1$ iff for all $s'_i \in S$ it holds that $\psi_1 \in \hat{L}^n(s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$ (Lines 20 – 22 in Algorithm 2).

4. Upon termination of the above iterative labeling procedure, as ψ is a sentence and thus state-independent, either all states are labelled with it or none of them. Return `true` if for an arbitrary state s we have $\psi \in \hat{L}^n(s)$ and return `false` otherwise.

Theorem 1. *For a finite Markov chain $\mathcal{M}$ and* HyperPCTL *formula ψ, the* HyperPCTL *model checking problem (to decide whether $\mathcal{M} \models \psi$) can be solved in time $O(\mathsf{poly}(|\mathcal{M}|))$.*

Theorem 2. *The* HyperPCTL *model checking problem is PSPACE-hard in the number of quantifiers in the formula.*

Proof. We show that the HyperPCTL model checking problem is PSPACE-hard by reducing the following PSPACE-hard *quantified Boolean formula* (QBF) satisfiability problem [9] to it:

Algorithm 2: HyperPCTL model checking algorithm II

Input : DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$, HyperPCTL quantified formula ψ, non-negative integer n, $\hat{L}^n : S^n \to 2^{\mathcal{F}}$

Output : An extension of $\hat{L}^n$ to label each state $s \in S^n$ with sub-formulas of ψ that hold in s

Function *HyperPCTL*$(\mathcal{M}, \psi, n, \hat{L}^n)$

- **if** $\psi = \texttt{true}$ **then**
  - for all $s \in S^n$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\texttt{true}\}$
- **else if** $\psi = a_{\sigma_i}$ **then**
  - for all $s \in S^n$ with $a_i \in L^n(s)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{a_{\sigma_i}\}$
- **else if** $\psi = \psi_1 \wedge \psi_2$ **then**
  - $\hat{L}^n$:=HyperPCTL($\mathcal{M}$,ψ_1,n,$\hat{L}^n$)
  - $\hat{L}^n$:=HyperPCTL($\mathcal{M}$,ψ_2,n,$\hat{L}^n$)
  - for all states $s \in S^n$ with $\{\psi_1, \psi_2\} \subseteq \hat{L}^n(s)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\psi\}$
- **else if** $\psi = \neg\psi_1$ **then**
  - $\hat{L}^n$:=HyperPCTL($\mathcal{M}$,ψ_1,n,$\hat{L}^n$)
  - for all states $s \in S^n$ with $\psi_1 \notin \hat{L}^n(s)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\psi\}$
- **else if** $\psi = p_1 \sim p_2$ **then**
  - $L_1^n := \text{ProbMC}(\mathcal{M}, p_1, n, \hat{L}^n)$ *% see Algorithm 3*
  - $L_2^n := \text{ProbMC}(\mathcal{M}, p_2, n, \hat{L}^n)$ *% see Algorithm 3*
  - for all states $s \in S^n$ with $L_1^n(s) \sim L_2^n(s)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\psi\}$
- **else if** $\psi = \exists\sigma_i.\psi_1$ **then**
  - $\hat{L}^n$:=HyperPCTL($\mathcal{M}$,ψ_1,n,$\hat{L}^n$)
  - for all states $s = (s_1, \ldots, s_n) \in S^n$ with $\psi_1 \in \hat{L}^n(s')$ for some $s'_i \in S$ and $s' = (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\psi\}$
- **else if** $\psi = \forall\sigma_i.\psi_1$ **then**
  - $\hat{L}^n$:=HyperPCTL($\mathcal{M}$,ψ_1,n,$\hat{L}^n$)
  - for all states $s = (s_1, \ldots, s_n) \in S^n$ with $\psi_1 \in \hat{L}^n(s')$ for all $s'_i \in S$ and $s' = (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$ set $\hat{L}^n(s) := \hat{L}^n(s) \cup \{\psi\}$
- **return** $\hat{L}^n$

Given is a set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables and a quantified Boolean formula

$$y = \mathbb{Q}_1 x_1 . \mathbb{Q}_1 x_2 \ldots \mathbb{Q}_{n-1} x_{n-1} . \mathbb{Q}_n x_n . \psi$$

where $\mathbb{Q}_i \in \{\forall, \exists\}$ for each $i \in [1, n]$ and ψ is an arbitrary Boolean formula over variables $\{x_1, \ldots, x_n\}$. Is y true?

We reduce the satisfiability problem for a quantified Boolean formula to the model checking problem for a HyperPCTL formula with the same quantifier structure as follows. We define the simple DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$ shown in Fig. 3, which contains two states s_0 and s_1 and has two paths s_0^ω and s_1^ω. The

Algorithm 3: HyperPCTL model checking algorithm III

Input : DTMC $\mathcal{M} = (S, \mathbf{P}, \mathsf{AP}, L)$, HyperPCTL probability expression p, non-negative integer n, $\hat{L}^n : S^n \to 2^{\mathcal{F}}$

Output : $L_p^n : S^n \to \mathbb{Q}$ specifying the values $L_p^n(s)$ of p in all states $s \in S^n$

Function *ProbMC*$(\mathcal{M}, p, n, \hat{L}^n)$
- let $L_p^n : S^n \to \mathbb{Q}$ with $L_p^n(s) = 0$ for all $s \in S^n$
- **if** $p = c$ **then**
  - for all $s \in S^n$ set $L_p^n(s) = c$
- **else if** $p = p_1$ *op* p_2 *with* $op \in \{+, -, \cdot\}$ **then**
  - $L_1^n :=$ probMC$(\mathcal{M}, p_1, n, \hat{L}^n)$
  - $L_2^n :=$ probMC$(\mathcal{M}, p_2, n, \hat{L}^n)$
  - for each $s \in S^n$ set $L_p^n(s) := L_1^n(s)$ *op* $L_2^n(s)$
- **else if** $p = \mathbb{P}(\varphi)$ **then**
  - **if** $\varphi = \bigcirc \psi$ **then**
    - for all $s \in S^n$ set $L_p^n(s) = \sum_{s' \in S^n,\ \psi \in \hat{L}^n(s')} \mathbf{P}^n(s, s')$
  - **else if** $\varphi = \psi_1 \,\mathcal{U}\, \psi_2$ **then**
    - compute the unique solution ν for the following equation system:
    - (1) $p_s = 0$ for all states $s \in S^n$ with $\psi_1 \notin \hat{L}^n(s)$ and $\psi_2 \notin \hat{L}^n(s)$, or if no state s' with $\psi_2 \in \hat{L}^n(s')$ is reachable from s
    - (2) $p_s = 1$ for all states $s \in S^n$ with $\psi_2 \in \hat{L}^n(s)$
    - (3) $p_s = \sum_{s' \in S^n} \mathbf{P}^n(s, s') \cdot p_{s'}$ for all other states
    - for all $s \in S^n$ set $L_p^n(s) = \nu(p_s)$
  - **else if** $\varphi = \psi_1 \,\mathcal{U}^{[k_1, k_2]} \psi_2$ **then**
    - for each $s \in S^n$ set $P_0^n(s) = 1$ if $\psi_2 \in \hat{L}^n(s)$ and $P_0^n(s) = 0$ otherwise
    - **for** $i = 1$ *to* k_2 **do**
      - for each $s \in S^n$ set $P_i^n(s) = \sum_{s' \in S^n} \mathbf{P}^n(s, s') \cdot P_{i-1}^n(s')$ if $\psi_1 \in \hat{L}^n(s)$ and $P_i^n(s) = 0$ otherwise
    - for all $s \in S^n$ set $L_p^n(s) = \sum_{i=k_1}^{k_2} P_i^n(s)$
- **return** L_p^n

HyperPCTL formula in our mapping is the following:

$$\mathbb{Q}_1 \sigma_1 . \mathbb{Q}_1 \sigma_2 \ldots \mathbb{Q}_{n-1} \sigma_{n-1} . \mathbb{Q}_n \sigma_n . \, \psi' \tag{1}$$

where ψ' is constructed from ψ by replacing every occurrence of a variable x_i in ψ by x_{σ_i}. The given quantified Boolean formula is `true` if and only if the DTMC obtained by our mapping satisfies HyperPCTL formula (1). We translate every assignment to the trace quantifiers to a corresponding assignment of the Boolean variables, and vice versa, as follows: Assigning state s_0 (s_1) to σ_i means that x_i is set to `true` (`false`). ■

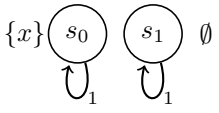

Fig. 3: DTMC in the proof of Thm 2.

5 Related Work

Probabilistic noninterference [13,14] establishes connection between information theory and information flow by employing probabilities to address covert channels. Intuitively, it requires that the probability of every pattern of low-observable trace be the same for every low-equivalent initial state. Most efforts in reasoning about probabilistic noninterference is through probabilistic weak bisimulation (e.g. [21]). More recently, Sabelfeld and Sands [19] introduce a framework to ensure nonintereference for multi-threaded programs, where a probabilistic scheduler non-deterministically manages the execution of threads. They introduce operational semantics for a simple imperative language with dynamic thread creation, and how compositionality is ensured.

Epistemic logic [7] is a subfield of modal logic that is concerned with reasoning about knowledge. The semantic model of the logic is a Kripke structure, where a set of agents are related with each other based on which states they consider possible. A probabilistic version of the logic [11] assigns a probability function to each agent at each state such that its domain is a non-empty subset of the set of possible states. Epistemic temporal logic has been used to express information-flow security policies (e.g., [2]). The relation between the expressive power of probabilistic epistemic logic and `HyperPCTL` remains an open question in this paper. Gray and Syverson [15] propose a modal logic for multi-level reasoning about security of probabilistic systems. The logic is axiomatic and is based on the Halpern and Tuttle [11] framework for reasoning about knowledge and probability. The logic is sound, but it may run into undecidability.

Clarkson and Schneider [5] introduce the notion of *hyperproperties*, a set-theoretic framework for expressing security policies. A hyperproperty is a set of sets of traces. In other words, a hyperproperty is a second-order property of properties. The expressive power of hyperproperties do not exceed the second-order logic, but it is currently unclear whether the full power of second-order logic is needed to express hyperproperties of interest. Clarkson and Schneider have shown two fundamental things: (1) a hyperproperty is an intersection of a safety and a liveness hyperproperty, and (2) hyperproperties can express many important requirements such as information-flow security policies (e.g., nonintereference, observational determinism, etc), service-level agreement, etc.

Second-order logic is not verifiable in general, as it cannot be effectively and completely axiomatized. Thus, temporal logics for subclasses of hyperproperties have emerged [4]. HyperLTL and HyperCTL* allow explicit and simultaneous quantification over multiple paths to LTL and to CTL*, respectively. As the names suggest, HyperLTL allow quantification of linear traces and HyperCTL* permits quantification over multiple execution traces simultaneously while allowing branching-time paths for each trace. HyperLTL and HyperCTL* are not equipped with probabilistic operators and cannot reason about probabilistic systems.

6 Conclusion and Future Work

In this paper, we proposed the temporal logic HyperPCTL to express and reason about probabilistic hyperproperties. HyperPCTL is a natural extension to PCTL by allowing explicit and simultaneous quantification over model states. We defined the syntax and semantics and presented a model checking algorithm for discrete-time Markov chains. The complexity of the algorithm is PSPACE-hard in the number of quantifiers in the input HyperPCTL formula. We presented multiple examples from different domains, where HyperPCTL can elegantly express the requirements.

We believe the results in this paper pave the path for new research directions. As for future work, an important unanswered question in this paper is to determine tighter lower and upper bounds for the the complexity of HyperPCTL model checking in the size of the formula. We believe most of the literature and fundamental lines of research on PCTL verification should now be revisited in the context of HyperPCTL. Examples include HyperPCTL model checking for Markov decision processes (MDPs), Markov chains with costs, parameter synthesis and model repair for probabilistic hyperproperties, HyperPCTL conditional probabilities, developing abstraction/refinement, comparing expressive power to existing related logics such as probabilistic epistemic logic [11], etc. An orthogonal direction is deeper investigation of the examples presented in Section 3. Each of those areas (e.g., differential privacy and probabilistic causation) deserve more research to develop effective and efficient model checking techniques.

References

1. B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.
2. M. Balliu, M. Dam, and G. Le Guernic. Epistemic temporal logic for information flow security. In *Proceedings of the 2011 Workshop on Programming Languages and Analysis for Security (PLAS)*, page 6, 2011.
3. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.
4. M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *Proceedings of the 3rd Conference on Principles of Security and Trust POST*, pages 265–284, 2014.
5. M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.
6. C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, August 2014.
7. R. Fagin, J.Y. Halpern nad Y. Moses, and M. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.
8. J. H. Fetzer, editor. *Probability and Causality*. Synthesis Library. Springer, 1988.
9. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
10. J. A. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symp. on Security and Privacy*, pages 11–20, 1982.

11. J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. In *Proceedings of the Eighth ACM Symposium on Principles of Distributed Computing (PODC)*, pages 103–118, 1989.
12. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
13. J. W. Gray III. Probabilistic interference. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy (S&P)*, pages 170–179, 1990.
14. J. W Gray III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3-4):255–294, May 1992.
15. J. W. Gray III and P. F. Syverson. A logical approach to multilevel security of probabilistic systems. *Distributed Computing*, 11(2):73–90, 1998.
16. L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2), March 1977.
17. A. Pnueli. The temporal logic of programs. In *Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.
18. H. Reichenbach. The direction of time, 1956.
19. A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 200–214, 2000.
20. B. Skyrms. *Causal Necessity*. Yale University Press, New Haven and London, 1980.
21. Geoffrey Smith. Probabilistic noninterference through weak probabilistic bisimulation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSF)*, pages 3–13, 2003.
22. S. Zdancewic and A. C. Myers. Observational determinism for concurrent program security. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW)*, page 29, 2003.