Probabilistic Hyperproperties of Markov Decision Processes*

Rayna Dimitrova[1], Bernd Finkbeiner[1], and Hazem Torfah[2]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[2] University of California at Berkeley, Berkeley, USA

Abstract Hyperproperties are properties that describe the correctness of a system as a relation between multiple executions. Hyperproperties generalize trace properties and include information-flow security requirements, like noninterference, as well as requirements like symmetry, partial observation, robustness, and fault tolerance. We initiate the study of the specification and verification of hyperproperties of Markov decision processes (MDPs). We introduce the temporal logic *PHL (Probabilistic Hyper Logic)*, which extends classic probabilistic logics with quantification over schedulers and traces. PHL can express a wide range of hyperproperties for probabilistic systems, including both classical applications, such as probabilistic noninterference, and novel applications in areas such as robotics and planning. While the model checking problem for PHL is in general undecidable, we provide methods both for proving and for refuting formulas from a fragment of the logic. The fragment includes many probabilistic hyperproperties of interest.

1 Introduction

Ten years ago, Clarkson and Schneider coined the term *hyperproperties* [10] for the class of properties that describe the correctness of a system as a relation between multiple executions. Hyperproperties include information-flow security requirements, like noninterference [17], as well as many other types of system requirements that cannot be expressed as trace properties, including symmetry, partial observation, robustness, and fault tolerance. Over the past decade, a rich set of tools for the specification and verification of hyperproperties have been developed. HyperLTL and HyperCTL* [9] are extensions to LTL and CTL* that can express a wide range of hyperproperties. There are a number of algorithms and tools for hardware model checking [16,11], satisfiability checking [15], and reactive synthesis [14] for hyperproperties.

The natural next step is to consider probabilistic systems. Randomization plays a key role in the design of security-critical and distributed systems. In fact,

* This work was partially supported by the Collaborative Research Center "Foundations of Perspicuous Software Systems" (TRR 248, 389792660), the European Research Council (ERC) Grant OSARES (No. 683300), the DARPA Assured Autonomy program, the iCyPhy center, and by Berkeley Deep Drive.

randomization is often added specifically to implement a certain hyperproperty. For example, randomized mutual exclusion protocols use a coin flip to decide which process gets access to the critical resource in order to avoid breaking the symmetry based on the process id [4]. Databases employ privacy mechanisms based on randomizaton in order to guarantee (differential) privacy [13].

Previous work on probabilistic hyperproperties [2] has focussed on the specification and verification of probabilistic hyperproperties for Markov chains. The logic HyperPCTL [2] extends the standard probabilistic logic PCTL with quantification over states. For example, the HyperPCTL formula

$$\forall s.\forall s'.\,(init_s \wedge init_{s'}) \rightarrow \mathbb{P}(\Diamond\, terminate_s) = \mathbb{P}(\Diamond\, terminate_{s'})$$

specifies that the probability that the system terminates is the same from all initial states. If the initial state encodes some secret, then the property guarantees that this secret is not revealed through the probability of termination.

Because Markov chains lack nondeterministic choice, they are a limited modeling tool. In an open system, the secret would likely be provided by an external environment, whose decisions would need to be represented by nondeterminism. In every step of the computation, such an environment would typically set the values of some low-security and some high-security input variables. In such a case, we would like to specify that the publicly observable behavior of our system does not depend on the infinite sequence of the values of the high-security input variables. Similarly, nondeterminism is needed to model the possible strategic decisions in autonomous systems, such as robots, or the content of the database in a privacy-critical system.

In this paper, we initiate the study of hyperproperties for *Markov decision processes* (MDPs). To formalize hyperproperties in this setting, we introduce PHL, a general temporal logic for probabilistic hyperproperties. The nondeterministic choices of an MDP are resolved by a *scheduler*[3]; correspondingly, our logic quantifies over schedulers. For example, in the PHL formula

$$\forall\sigma.\forall\sigma'.\,\mathbb{P}(\Diamond\, terminate_\sigma) = \mathbb{P}(\Diamond\, terminate_{\sigma'})$$

the variables σ and σ' refer to schedulers. The formula specifies that the probability of termination is the same for all of the possible (infinite) combinations of the nondeterministic choices. If we wish to distinguish different types of inputs, for example those that are provided through a high-security variable h vs. those provided through a low-security variable l, then the quantification can be restricted to those schedulers that make the same low-security choices:

$$\forall\sigma.\forall\sigma'.\,(\forall\pi : \sigma.\forall\pi' : \sigma'.\ \square(l_\pi \leftrightarrow l_{\pi'})) \rightarrow \mathbb{P}(\Diamond\, terminate_\sigma) = \mathbb{P}(\Diamond\, terminate_{\sigma'})$$

The path quantifier $\forall\pi : \sigma$ works analogously to the quantifiers in HyperCTL*, here restricted to the paths of the Markov chain induced by the scheduler assigned to variable σ. The formula thus states that all schedulers that agree on the low-security inputs induce the same probability of termination.

[3] In the literature, schedulers are also known as strategies or policies.

As we show in the paper, PHL is a very expressive logic, thanks to the combination of scheduler quantifiers, path quantifiers and a probabilistic operator. PHL has both classical applications, such as differential privacy, as well as novel applications in areas such as robotics and planning. For example, we can quantify the interference of the plans of different agents in a multi-agent system, such as the robots in a warehouse, or we can specify the existence of an approximately optimal policy that meets given constraints. A consequence of the generality of the logic is that it is impossible to simply reduce the model checking problem to that of a simpler temporal logic in the style of the reduction of HyperPCTL to PCTL [2]. In fact, we show that the emptiness problem for probabilistic Büchi automata (PBA) can be encoded in PHL, which implies that the model checking problem for PHL is, in general, undecidable.

We present two verification procedures that approximate the model checking problem from two sides. The first algorithm *overapproximates* the model checking problem by quantifying over a combined monolithic scheduler rather than a tuple of independent schedulers. Combined schedulers have access to more information than individual ones, meaning that the set of allowed schedulers is overapproximated. This means that if a universal formula is true for all combined schedulers it is also true for all tuples of independent schedulers. The second procedure is a bounded model checking algorithm that *underapproximates* the model checking problem by bounding the number of states of the schedulers. This algorithm is obtained as a combination of a bounded synthesis algorithm for hyperproperties, which generates the schedulers, and a model checking algorithm for Markov chains, which computes the probabilities on the Markov chains induced by the schedulers. Together, the two algorithms thus provide methods both for proving and for refuting a class of probabilistic hyperproperties for MDPs.

Related work Probabilistic noninterference originated in information-flow security [18,21] and is a security policy that requires that the probability of every low trace should be the same for every low equivalent initial state. Volpano and Smith [24] presented a type system for checking probabilistic noninterference of concurrent programs with probabilistic schedulers. Sabelfeld and Sands [23] defined a secure type system for multi-threaded programs with dynamic thread creation which improves on that of Volpano and Smith. None of these works is concerned with models combining probabilistic choice with nondeterminism, nor with general temporal logics for probabilistic hyperproperties.

The specification and verification of probabilistic hyperproperties have recently attracted significant attention. Abraham and Bonakdarpour [2] are the first to study a temporal logic for probabilistic hyperproperties, called HyperPCTL. The logic allows for explicit quantification over the states of a Markov chain, and is capable of expressing information-flow properties like probabilistic noninterference. The authors present a model checking algorithm for verifying HyperPCTL on finite-state Markov chains. HyperPCTL was extended to a logic called HyperPCTL* [25] that allows nesting of temporal and probabilistic operators, and a statistical model checking method for HyperPCTL* was proposed. Our present work, on the other hand is concerned with the specification

and model checking of probabilistic hyperproperties for system models featuring both probabilistic choice and nondeterminism, which are beyond the scope of all previous temporal logics for probabilistic hyperproperties. Probabilistic logics with quantification over schedulers have been studied in [6] and [3]. However, these logics do not include quantifiers over paths.

Independently and concurrently to our work, probabilistic hyperproperties for MDPs were also studied in [1] (also presented at ATVA'20). The authors extend HyperPCTL with quantifiers over schedulers, while our new logic PHL extends HyperCTL* with the probabilistic operator and quantifiers over schedulers. Thus, HyperPCTL quantifies over states (i.e., the computation trees that start from the states), while PHL quantifies over paths. Both papers show that the model checking problem is undecidable for the respective logics. The difference is in how the approaches deal with the undecidability result. For both logics, the problem is decidable when quantifiers are restricted to non-probabilistic memoryless schedulers. [1] provides an SMT-based verification procedure for HyperPCTL for this class of schedulers. We consider general memoryful schedulers and present two methods for proving and for refuting formulas from a fragment of PHL.

Due to lack of space we have omitted the proofs of our results and details of the presented model checking procedures, which can be found in [12].

2 Preliminaries

Definition 1 (Markov Decision Process (MDP)). *A* Markov Decision Process (MDP) *is a tuple* $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ *where* S *is a finite set of states,* Act *is a finite set of actions,* $\mathbf{P} : S \times Act \times S \to [0,1]$ *is the transition probability function such that* $\sum_{s' \in S} \mathbf{P}(s,a,s') \in \{0,1\}$ *for every* $s \in S$ *and* $a \in Act$, $\iota : S \to [0,1]$ *is the initial distribution such that* $\sum_{s \in S} \iota(s) = 1$, AP *is a finite set of atomic propositions and* $L : S \to 2^{\mathsf{AP}}$ *is a labelling function.*

A finite path in an MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ is a sequence $s_0 s_1 \dots s_n$ where for every $0 \leq i < n$ there exists $a_i \in Act$ such that $\mathbf{P}(s_i, a_i, s_{i+1}) > 0$. Infinite paths in M are defined analogously. We denote with $Paths_{fin}(M)$ and $Paths_{inf}(M)$ the sets of finite and infinite paths in M. For an infinite path $\rho = s_0 s_1 \dots$ and $i \in \mathbb{N}$ we denote with $\rho[i, \infty)$ the infinite suffix $s_i s_{i+1} \dots$. Given $s \in S$, define $Paths_{fin}(M, s) = \{s_0 s_1 \dots s_n \in Paths_{fin}(M) \mid s_0 = s\}$, and similarly $Paths_{inf}(M, s)$. We denote with $M_s = (S, Act, \mathbf{P}, \iota_s, \mathsf{AP}, L)$ the MDP obtained from M by making s the single initial state, i.e., $\iota_s(s) = 1$ and $\iota_s(t) = 0$ for $t \neq s$.

For a set A we denote with $\mathcal{D}(A)$ the set of probability distributions on A.

Definition 2 (Scheduler). *A* scheduler *for an MDP* $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ *is a function* $\mathfrak{S} : (S \cdot Act)^* S \to \mathcal{D}(Act)$ *such that for all sequences* $s_0 a_0 \dots a_{n-1} s_n \in (S \cdot Act)^* S$ *it holds that if* $\mathfrak{S}(s_0 a_0 \dots a_{n-1} s_n)(a) > 0$ *then* $\sum_{t \in S} \mathbf{P}(s_n, a, t) > 0$, *that is, each action in the support of* $\mathfrak{S}(s_0 a_0 \dots a_{n-1} s_n)$ *is enabled in* s_n. *We define* $Sched(M)$ *to be the set consisting of all schedulers for an MDP* M.

Given an MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ and a scheduler $\mathfrak{S}$ for M, we denote with $M_{\mathfrak{S}}$ the *Markov chain of M induced by $\mathfrak{S}$*, which is defined as the tuple $M_{\mathfrak{S}} = ((S \cdot Act)^* S, \mathbf{P}_{\mathfrak{S}}, \iota, AP, L_{\mathfrak{S}})$ where for every sequence $h = s_0 a_0 \dots a_{n-1} s_n \in (S \cdot Act)^* S$ it holds that $\mathbf{P}_{\mathfrak{S}}(h, h \cdot s_{n+1}) = \sum_{a \in Act} \mathfrak{S}(h)(a) \cdot \mathbf{P}(s_n, a, s_{n+1})$ and $L_{\mathfrak{S}}(h) = L(s_n)$. Note that $M_{\mathfrak{S}}$ is infinite even when M is finite. The different types of paths in a Markov chain are defined as for MDPs.

Of specific interest are *finite-memory* schedulers, which are schedulers that can be represented as finite-state machines. Formally, a finite-memory scheduler for M is represented as a tuple $\mathcal{T}_{\mathfrak{S}} = (Q, \delta, q_0, act)$, where Q is a finite set of states, representing the memory of the scheduler, $\delta : Q \times S \times Act \to Q$ is a memory update function, q_0 is the initial state of the memory, and $act : Q \times S \to \mathcal{D}(Act)$ is a function that based on the current memory state and the state of the MDP returns a distribution over actions. Such a representation defines a function $\mathfrak{S} : (S \cdot Act)^* S \to \mathcal{D}(Act)$ as follows. First, let us define the function $\delta^* : Q \times (S \cdot Act)^* \to Q$ as follows: $\delta^*(q, \epsilon) = q$ for all $q \in Q$, and $\delta^*(q, s_0 a_0 \dots s_n a_n s_{n+1} a_{n+1}) = \delta(\delta^*(q, s_0 a_0 \dots s_n a_n), s_{n+1}, a_{n+1})$ for all $q \in Q$ and all $s_0 a_0 \dots s_n a_n s_{n+1} a_{n+1} \in (S \cdot Act)^*$. Now, we define the scheduler function represented by $\mathcal{T}_{\mathfrak{S}}$ by $\mathfrak{S}(s_0 a_0 \dots s_n a_n s_{n+1}) = act(\delta^*(s_0 a_0 \dots s_n a_n), s_{n+1})$.

Finite-memory schedulers induce finite Markov chains with simpler representation. A finite memory scheduler $\mathfrak{S}$ represented by $\mathcal{T}_{\mathfrak{S}} = (Q, \delta, q_0, act)$ induces the Markov chain $M_{\mathfrak{S}} = (S \times Q, \mathbf{P}_{\mathfrak{S}}, \iota_{\mathfrak{S}}, AP, L_{\mathfrak{S}})$ where $\mathbf{P}_{\mathfrak{S}}((s, q), (s', q')) = \sum_{a \in Act} act(q, s)(a) \cdot \mathbf{P}(s, a, s')$ if $q' = \delta(q, s)$, otherwise $\mathbf{P}_{\mathfrak{S}}((s, q), (s', q')) = 0$, and $\iota_{\mathfrak{S}}(s, q) = \iota(s)$ if $q = q_0$ and $\iota_{\mathfrak{S}}(s, q) = 0$ otherwise.

A scheduler $\mathfrak{S}$ is *deterministic* if for every $h \in (S \cdot Act)^* S$ it holds that $\mathfrak{S}(h)(a) = 1$ for exactly one $a \in Act$. By abuse of notation, a deterministic scheduler can be represented as a function $\mathfrak{S} : S^+ \to Act$, that maps a finite sequence of states to the single action in the support of the corresponding distribution. Note that for deterministic schedulers we omit the actions from the history as they are uniquely determined by the sequence of states. We write $DetSched(M)$ for the set of deterministic schedulers for the MDP M.

A *probability space* is a triple $(\Omega, \mathcal{F}, Prob)$, where Ω is a sample space, $\mathcal{F} \subseteq 2^{\Omega}$ is a σ-algebra and $Prob : \mathcal{F} \to [0, 1]$ is a probability measure.

Given a Markov chain $C = (S, \mathbf{P}, \iota, AP, L)$, it is well known how to associate a probability space $(\Omega^C, \mathcal{F}^C, Prob^C)$ with C. The sample space $\Omega^C = Paths_{inf}(C)$ is the set of infinite paths in C, where the sets of finite and infinite paths for a Markov chain are defined in the same way as for MDP. The σ-algebra $\mathcal{F}^C$ is the smallest σ-algebra that for each $\pi \in Paths_{fin}(C)$ contains the set $Cyl_C(\pi) = \{\rho \in Paths_{inf}(C) \mid \exists \rho' \in Paths_{inf}(C) : \rho = \pi \cdot \rho'\}$ called the cylinder set of the finite path π. $Prob^C$ is the unique probability measure such that for each $\pi = s_0 \dots s_n \in Paths_{fin}(C)$ it holds that $Prob^C(Cyl(\pi)) = \iota(s_0) \cdot \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})$.

Analogously, given any state $s \in S$ we denote with $(\Omega^C, \mathcal{F}^C, Prob^C_s)$ the probability space for paths in C originating in the state s, i.e., the probability space associated with the Markov chain C_s (where C_s is defined as for MDPs).

When considering a Markov chain $M_{\mathfrak{S}}$ induced by an MDP M and a scheduler $\mathfrak{S}$, we write $Prob_{M,\mathfrak{S}}$ and $Prob_{M,\mathfrak{S},s}$ for the sake of readability.

3 The Logic PHL

In this section we define the syntax and semantics of PHL, the logic which we introduce and study in this work. PHL allows for quantification over schedulers and integrates features of temporal logics for hyper properties, such as HyperLTL and HyperCTL* [9], and probabilistic temporal logics such as PCTL*.

3.1 Examples of PHL Specifications

We illustrate the expressiveness of PHL with two applications beyond information-flow security, from the domains of robotics and planning.

Example 1 (Action cause). Consider the question whether a car on a highway that enters the opposite lane (action b) when there is a car approaching from the opposite direction (condition p) increases the probability of an accident (effect e). This can be formalized as the property stating that there exist two deterministic schedulers σ_1 and σ_2 such that (i) in σ_1 the action b is never taken when p is satisfied, (ii) the only differences between σ_1 and σ_2 can happen when σ_2 takes action b when p is satisfied, and (iii) the probability of e being true eventually is higher in the Markov chain induced by σ_2 than in the one for σ_1. To express this property in our logic, we will use *scheduler quantifiers* quantifying over the schedulers for the MDP. To capture the condition on the way the schedulers differ, we will use *path quantifiers* quantifying over the paths in the Markov chain induced by each scheduler. The atomic propositions in a PHL formula are indexed with path variables when they are interpreted on a given path, and with scheduler variables when they are interpreted in the Markov chain induced by that scheduler. Formally, we can express the property with the PHL formula

$$\exists \sigma_1 \exists \sigma_2.\ (\forall \pi_1 : \sigma_1 \forall \pi_2 : \sigma_2.\ (\square \neg (p_{\pi_1} \wedge \bigcirc b_{\pi_1})) \wedge \psi) \wedge \mathbb{P}(\diamondsuit e_{\sigma_1}) < \mathbb{P}(\diamondsuit e_{\sigma_2}),$$

where $\psi = \big((\bigwedge_{a \in Act} (\bigcirc a_{\pi_1} \leftrightarrow \bigcirc a_{\pi_2})) \vee (p_{\pi_2} \wedge \bigcirc b_{\pi_2})\big)\, \mathcal{W} (\bigvee_{q \in \mathsf{AP} \setminus Act} (q_{\pi_1} \not\leftrightarrow q_{\pi_2}))$.

The two conjuncts of $\forall \pi_1 : \sigma_1 \forall \pi_2 : \sigma_2.\ (\square \neg (p_{\pi_1} \wedge \bigcirc b_{\pi_1})) \wedge \psi$ capture conditions (i) and (ii) above respectively, and $\mathbb{P}(\diamondsuit e_{\sigma_1}) < \mathbb{P}(\diamondsuit e_{\sigma_2})$ formalizes (iii). Here we assume that actions are represented in AP, i.e., $Act \subseteq \mathsf{AP}$ □

Example 2 (Plan non-interference). Consider two robots in a warehouse, possibly attempting to reach the same location. Our goal is to determine whether all plans for the first robot to move towards the goal are robust against interferences from arbitrary plans of the other robot. That is, we want to check whether for every plan of robot 1 the probability that it reaches the goal under an arbitrary plan of robot 2 is close to that of the same plan for robot 1 executed under any other plan for robot 2. We can express this property in PHL by using *quantifiers over schedulers* to quantify over the joint deterministic plans of the robots, and using *path quantifiers* to express the condition that in both joint plans robot 1 behaves the same. Formally, we can express the property with the PHL formula

$$\begin{aligned} \forall \sigma_1 \forall \sigma_2.\ (\forall \pi_1 : \sigma_1 \forall \pi_2 : \sigma_2.\ \square(\mathit{move1}_{\pi_1} \leftrightarrow \mathit{move1}_{\pi_2})) \rightarrow \\ \mathbb{P}(\diamondsuit(\mathit{goal1}_{\sigma_1} \wedge \neg \mathit{goal2}_{\sigma_1})) - \mathbb{P}(\diamondsuit(\mathit{goal1}_{\sigma_2} \wedge \neg \mathit{goal2}_{\sigma_2})) \leq \varepsilon, \end{aligned}$$

where σ_1 and σ_2 are scheduler variables, π_1 is a path variable associated with the scheduler for σ_1, and π_2 is a path variable associated with the scheduler for σ_2. The condition $\forall\pi_1 : \sigma_1 \forall\pi_2 : \sigma_2.\ \square(\mathit{move1}_{\pi_1} \leftrightarrow \mathit{move1}_{\pi_2})$ states that in both joint plans robot 1 executes the same moves, where the proposition *move1* corresponds to robot 1 making a move towards the goal. The formula $\mathbb{P}(\diamondsuit(\mathit{goal1}_{\sigma_1} \wedge \neg \mathit{goal2}_{\sigma_1})) - \mathbb{P}(\diamondsuit(\mathit{goal1}_{\sigma_2} \wedge \neg \mathit{goal2}_{\sigma_2})) \leq \varepsilon$ states that the difference in the probability of robot 1 reaching the goal under scheduler σ_1 and the probability of it reaching the goal under scheduler σ_2 does not exceed ε. □

3.2 Syntax

As we are concerned with hyperproperties interpreted over MDPs, our logic allows for quantification over schedulers and quantification over paths.

To this end, let $\mathcal{V}_{sched}$ be a countably infinite set of *scheduler variables* and let $\mathcal{V}_{path}$ be a countably infinite set of *path variables*. According to the semantics of our logic, quantification over path variables ranges over the paths in a Markov chain associated with the scheduler represented by a given scheduler variable. To express this dependency we will associate path variables with the corresponding scheduler variable, writing $\pi : \sigma$ for a path variable π associated with a scheduler variable σ. The precise use and meaning of this notation will become clear below, once we define the syntax and semantics of the logic.

Given a set AP of atomic propositions, PHL formulas over AP will use atomic propositions indexed with scheduler variables or with path variables. We define the sets of propositions indexed with scheduler variables as $\mathsf{AP}_{\mathcal{V}_{sched}} = \{a_\sigma \mid a \in \mathsf{AP}, \sigma \in \mathcal{V}_{sched}\}$ and with path variables as $\mathsf{AP}_{\mathcal{V}_{path}} = \{a_\pi \mid a \in \mathsf{AP}, \pi \in \mathcal{V}_{path}\}$.

PHL (Probabilistic Hyper Logic) formulas are defined by the grammar

$$\Phi ::= \quad \forall\sigma.\ \Phi \quad \mid \quad \Phi \wedge \Phi \quad \mid \quad \neg\Phi \quad \mid \quad \chi \quad \mid \quad P \bowtie c$$

where $\sigma \in \mathcal{V}_{sched}$ is a scheduler variable, χ is a HyperCTL* formula, P is a *probabilistic expression* defined below, $\bowtie \in \{\leq, \leq, \geq, \geq\}$, and $c \in \mathbb{Q}$.

Formulas in HyperCTL*, introduced in [9], are constructed by the grammar

$$\chi ::= a_\pi \ \mid\ \chi \wedge \chi \ \mid\ \neg\chi \ \mid\ \bigcirc\chi \ \mid\ \chi\,\mathcal{U}\,\chi \ \mid\ \forall\pi : \sigma.\ \chi$$

where π is a path variable associated with a scheduler variable σ, and $a \in \mathsf{AP}$.

Probability expressions are defined by the grammar

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

where $\mathbb{P}$ is the *probabilistic operator*, $c \in \mathbb{Q}$, and φ is an LTL formula [22] defined by the grammar below, where $a \in \mathsf{AP}$ and σ is a scheduler variable.

$$\varphi ::= a_\sigma \ \mid\ \varphi \wedge \varphi \ \mid\ \neg\varphi \ \mid\ \bigcirc\varphi \ \mid\ \varphi\,\mathcal{U}\,\varphi.$$

We call formulas of the form $P \bowtie c$ *probabilistic predicates*.

A PHL formula Φ is *well-formed* if each path quantifier for $\pi : \sigma$ that appears in Φ is in the scope of a scheduler quantifier with the scheduler variable σ.

A PHL formula is *closed* if all occurrences of scheduler and path variables are bound by scheduler and path quantifiers respectively.

In the following we consider only closed and well-formed PHL formulas.

Discussion Intuitively, a PHL formula is a Boolean combination of formulas consisting of a scheduler quantifier prefix followed by a formula without scheduler quantifiers constructed from probabilistic predicates and HyperCTL* formulas via propositional operators. Thus, interleaving path quantifiers and probabilistic predicates is not allowed in PHL. This design decision is in line with the fact that probabilistic temporal logics like PCTL* replace the path quantifiers with the probabilistic operator that can be seen as their quantitative counterpart. We further chose to not allow nesting of probabilistic predicates and temporal operators, as in all the examples that we considered we never encountered the need for nested $\mathbb{P}$ operators. Moreover, allowing arbitrary nesting of probabilistic and temporal operators would immediately make the model checking problem for the resulting logic undecidable, following from the results in [8].

3.3 Self-Composition for MDPs

In order to define the semantics of PHL we first introduce the self-composition operation for MDPs, which lifts to MDPs the well-known self-composition of transition systems that is often used in the model checking of hyperproperties.

Let us fix, for the reminder of the section, an MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$.

Definition 3 (n-self-composition of MDP). *Let $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ be an MDP and $n \in \mathbb{N}_{>0}$ be a constant. The n-*self-composition of M *is the MDP $M^n = (S^n, Act^n, \widehat{\mathbf{P}}, \widehat{\iota}, \mathsf{AP}, \widehat{L})$ with the following components. $S^n = \{(s_1, \ldots, s_n) \mid s_i \in S$ for all $1 \leq i \leq n\}$ is the set of states. $Act^n = \{(a_1, \ldots, a_n) \mid a_i \in Act$ for all $1 \leq i \leq n\}$ is the set of actions. The transition probability function $\widehat{\mathbf{P}}$ is such that for every $(s_1, \ldots, s_n), (s'_1, \ldots, s'_n) \in S^n$ and $(a_1, \ldots, a_n) \in Act^n$ we have $\widehat{\mathbf{P}}((s_1, \ldots, s_n), (a_1, \ldots, a_n), (s'_1, \ldots, s'_n)) = \prod_{i=1}^{n} \mathbf{P}(s_i, a_i, s'_i)$. The initial distribution such that $\widehat{\iota}((s_1, \ldots, s_n)) = \iota(s_1)$ if $s_1 = \ldots = s_n = s$ and $\widehat{\iota}((s_1, \ldots, s_n)) = 0$ otherwise. The labelling function $\widehat{L} : S^n \to (2^{\mathsf{AP}})^n$ maps states to n-tuples of subsets of AP (in contrast to Definition 1 where states are mapped to subsets of AP) and is given by $\widehat{L}((s_1, \ldots, s_n)) = (L(s_1), \ldots, L(s_n))$.*

Naturally, a scheduler $\widehat{\mathfrak{S}} \in Sched(M^n)$ induces a Markov chain $M^n_{\widehat{\mathfrak{S}}}$.

Given schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n \in Sched(M)$, their *composition*, a scheduler $\overline{\mathfrak{S}} : (S^n \cdot Act^n)^* S^n \to \mathcal{D}(Act^n)$ for M^n, is denoted $\overline{\mathfrak{S}} = \mathfrak{S}_1 \parallel \ldots \parallel \mathfrak{S}_n$ and such that for every $\overline{h} = (s_{1,1}, \ldots, s_{1,n})(a_{1,1}, \ldots, a_{1,n}) \ldots (s_{k,1}, \ldots, s_{k,n}) \in (S^n \cdot Act^n)^* S^n$ and $\overline{a} = (a_{k+1,1}, \ldots, a_{k+1,n}) \in Act^n$, $\overline{\mathfrak{S}}(\overline{h})(\overline{a}) = \prod_{i=1}^{n} \mathfrak{S}_i(s_{1,i} a_{1,i} \ldots s_{k,i})(a_{k+1,i})$.

3.4 Scheduler and Path Assignments

Let $\mathcal{V}_{sched}$ and $\mathcal{V}_{path}$ be the sets of scheduler and path variables respectively.

A *scheduler assignment* is a vector of pairs $\Sigma \in \bigcup_{n \in \mathbb{N}} (\mathcal{V}_{sched} \times Sched(M))^n$ that assigns schedulers to some of the scheduler variables. Given a scheduler assignment $\Sigma = ((\sigma_1, \mathfrak{S}_1), \ldots, (\sigma_n, \mathfrak{S}_n))$, we denote by $|\Sigma|$ the length (number of pairs) of the vector. For a scheduler variable $\sigma \in \mathcal{V}_{sched}$ we define $\Sigma(\sigma) = \mathfrak{S}_i$

where i is the maximal index such that $\sigma_i = \sigma$. If such an index i does not exits, $\Sigma(\sigma)$ is undefined. For a scheduler assignment $\Sigma = ((\sigma_1, \mathfrak{S}_1), \ldots, (\sigma_n, \mathfrak{S}_n))$, a scheduler variable $\sigma \in \mathcal{V}_{sched}$, and a scheduler $\mathfrak{S} \in \mathit{Sched}(M)$ we define the scheduler assignment $\Sigma[\sigma \mapsto \mathfrak{S}] = ((\sigma_1, \mathfrak{S}_1), \ldots, (\sigma_n, \mathfrak{S}_n), (\sigma, \mathfrak{S}))$ obtained by adding the pair $(\sigma, \mathfrak{S})$ to the end of the vector Σ.

Given the MDP M, let $\Sigma = ((\sigma_1, \mathfrak{S}_1), \ldots, (\sigma_n, \mathfrak{S}_n))$ be a scheduler assignment, and consider $M^{|\Sigma|}$, the $|\Sigma|$-self composition of M. Σ defines a scheduler for $M^{|\Sigma|}$, which is the product of the schedulers in Σ, i.e., $\overline{\mathfrak{S}} = \mathfrak{S}_1 \parallel \ldots \parallel \mathfrak{S}_n$. Let M_Σ be the Markov chain induced by $\overline{\mathfrak{S}}$. If $\widehat{s}$ is a state in M_Σ, we denote by $M_{\Sigma,\widehat{s}}$ the Markov chain obtained from M_Σ by making $\widehat{s}$ the single initial state.

Note that the labeling function $\widehat{L}$ in $M^{|\Sigma|}$ maps the states in $S^{|\Sigma|}$ to $|\Sigma|$-tuples of sets of atomic predicates, that is $\widehat{L}(\widehat{s}) = (L_1, \ldots, L_{|\Sigma|})$. Given a scheduler variable σ for which $\Sigma(\sigma)$ is defined, we write $\widehat{L}(\widehat{s})(\sigma)$ for the set of atomic predicates L_i, where i is the maximal position in Σ in which σ appears.

We define path assignments similarly to scheduler assignments. A *path assignment* is a vector of pairs of path variables and paths in $\mathit{Paths}_{inf}(M)$. More precisely, a path assignment Π is an element of $\bigcup_{m\in\mathbb{N}}(\mathcal{V}_{path} \times \mathit{Paths}_{inf}(M))^m$. Analogously to scheduler assignments, for a path variable π and a path $\rho \in \mathit{Paths}_{inf}(M)$, we define $\Pi(\pi)$ and $\Pi[\pi \mapsto \rho]$. For $\Pi = ((\pi_1, \rho_1), \ldots, (\pi_n, \rho_n))$ and $j \in \mathbb{N}$, we define $\Pi[j, \infty] = ((\pi_1, \rho_1[j, \infty]), \ldots, (\pi_n, \rho_n[j, \infty]))$ to be the path assignment that assigns to each π_i the suffix $\rho_i[j, \infty]$ of the path ρ_i.

3.5 Semantics of PHL

We are now ready to define the semantics of PHL formulas. Recall that we consider only closed and well-formed PHL formulas. PHL formulas are interpreted over an MDP and a scheduler assignment. The interpretation of HyperCTL* formulas requires additionally a path assignment. Probabilistic expressions and LTL formulas are evaluated in the Markov chain for an MDP induced by a scheduler assignment. As usual, the satisfaction relations are denoted by $\models$.

For an MDP M and a scheduler assignment Σ we define

$$
\begin{array}{lll}
M, \Sigma \models \forall\sigma.\Phi & \text{iff} & \text{for all } \mathfrak{S} \in \mathit{Sched}(M) : \ M, \Sigma[\sigma \mapsto \mathfrak{S}] \models \Phi; \\
M, \Sigma \models \Phi_1 \wedge \Phi_2 & \text{iff} & M, \Sigma \models \Phi_1 \text{ and } M, \Sigma \models \Phi_2; \\
M, \Sigma \models \neg\Phi & \text{iff} & M, \Sigma \not\models \Phi; \\
M, \Sigma \models \chi & \text{iff} & M, \Sigma, \Pi_\emptyset \models \chi, \text{where } \Pi_\emptyset \text{ is the empty path assignment}; \\
M, \Sigma \models P \bowtie c & \text{iff} & [\![P]\!]_{M_\Sigma} \bowtie c.
\end{array}
$$

For an MDP M, scheduler assignment Σ, and path assignment Π we define

$$
\begin{array}{lll}
M, \Sigma, \Pi \models a_\pi & \text{iff} & a \in L(\Pi(\pi)[0]); \\
M, \Sigma, \Pi \models \chi_1 \wedge \chi_2 & \text{iff} & M, \Sigma, \Pi \models \chi_1 \text{ and } M, \Sigma, \Pi \models \chi_2; \\
M, \Sigma, \Pi \models \neg\chi & \text{iff} & M, \Sigma, \Pi \not\models \chi; \\
M, \Sigma, \Pi \models \bigcirc\chi & \text{iff} & M, \Sigma, \Pi[1, \infty] \models \chi; \\
M, \Sigma, \Pi \models \chi_1 \,\mathcal{U}\, \chi_2 & \text{iff} & \text{there exists } i \geq 0 : M, \Sigma, \Pi[i, \infty] \models \chi_2 \text{ and} \\
 & & \qquad \text{for all } j < i : M, \Sigma, \Pi[j, \infty] \models \chi_1; \\
M, \Sigma, \Pi \models \forall\pi : \sigma.\ \chi & \text{iff} & \text{for all } \rho \in \mathit{Paths}_{inf}(C) : M, \Sigma, \Pi[\pi \mapsto \rho] \models \chi,
\end{array}
$$

where in the last item C is the Markov chain $M_{\Sigma(\sigma)}$ when Π is the empty path assignment, and otherwise the Markov chain $M_{\Sigma(\sigma),\Pi(\pi')[0]}$ where π' is the path variable associated with scheduler variable σ that was most recently added to Π.

For Markov chain C of the form M_Σ or $M_{\Sigma,\widehat{s}}$, where Σ is a scheduler assignment and $\widehat{s}$ is a state in M_Σ the semantics $[\![\cdot]\!]_C$ of probabilistic expressions is:

$$\begin{array}{ll} [\![\mathbb{P}(\varphi)]\!]_C & = \mathit{Prob}^C(\{\rho \in \mathit{Paths}_{\mathit{inf}}(C) \mid C, \rho \models \varphi\}); \\ [\![P_1 + P_2]\!]_C & = [\![P_1]\!]_C + [\![P_2]\!]_C; \quad [\![c \cdot P]\!]_C \;=\; c \cdot [\![P]\!]_C, \end{array}$$

where the semantics of path formulas (i.e., LTL formulas) is defined by

$$\begin{array}{lll} C, \rho \models a_\sigma & \text{iff} & a \in \widehat{L}(\rho[0])(\sigma); \\ C, \rho \models \varphi_1 \wedge \varphi_2 & \text{iff} & C, \rho \models \varphi_1 \text{ and } C, \rho \models \varphi_2; \\ C, \rho \models \neg\varphi & \text{iff} & C, \rho \not\models \varphi; \\ C, \rho \models \bigcirc\varphi & \text{iff} & C, \rho[1,\infty] \models \varphi; \\ C, \rho \models \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{iff} & \text{there exists } i \geq 0 : C, \rho[i,\infty] \models \varphi_2 \text{ and} \\ & & \qquad\qquad \text{for all } j < i : C, \rho[j,\infty] \models \varphi_1. \end{array}$$

Note that $\mathit{Prob}^C(\{\rho \in \mathit{Paths}_{\mathit{inf}}(C) \mid C, \rho \models \varphi\})$ is well-defined as it is a known fact [7] that the set $\{\rho \in \mathit{Paths}_{\mathit{inf}}(C) \mid C, \rho \models \varphi\}$ is measurable.

We say that an MDP M *satisfies* a closed well-formed PHL formula Φ, denoted $M \models \Phi$ iff $M, \Sigma_\emptyset \models \Phi$, where $\Sigma_\emptyset$ is the empty scheduler assignment.

Since PHL includes both scheduler and path quantification, the sets of deterministic and randomized schedulers are not interchangeable with respect to the PHL semantics. That is, there exists an MDP M and formula Φ such that if quantifiers are interpreted over $\mathit{Sched}(M)$, then $M \models \Phi$, and if quantifiers are interpreted over $\mathit{DetSched}(M)$ then $M \not\models \Phi$. See [12] for an example.

3.6 Undecidability of PHL Model Checking

Due to the fact that PHL allows quantification over both schedulers and paths, the model checking problem for PHL is undecidable. The proof is based on a reduction from the emptiness problem for probabilistic Büchi automata (PBA), which is known to be undecidable [5].

Theorem 1. *The model checking problem for PHL is undecidable.*

We saw in the previous section an example of a probabilistic hyperproperty expressed as a PHL formulas of the form $\forall\sigma_1 \ldots \forall\sigma_n.\ \big((\forall\pi_1 : \sigma_1 \ldots \forall\pi_n : \sigma_n.\ \psi) \rightarrow P \bowtie c\big)$. Analogously to Theorem 1, we can show that the model checking problem for PHL formulas of the form $\exists\sigma_1 \ldots \exists\sigma_n.\ (\forall\pi_1 : \sigma_1 \ldots \forall\pi_n : \sigma_n.\ \psi \wedge P \bowtie c)$ is undecidable. The undecidability for formulas of the form $\forall\sigma_1 \ldots \forall\sigma_n.\ \big((\forall\pi_1 : \sigma_1 \ldots \forall\pi_n : \sigma_n.\ \psi) \rightarrow P \bowtie c\big)$ then follows by duality. In the next two sections, we present an approximate model checking procedure and a bounded model checking procedure for PHL formulas in these two classes.

However, since there are finitely many deterministic schedulers with a given fixed number of states, the result stated in the next theorem is easily established.

Theorem 2. *For any constant $b \in \mathbb{N}$, the model checking problem for PHL restricted to deterministic finite-memory schedulers with b states is decidable.*

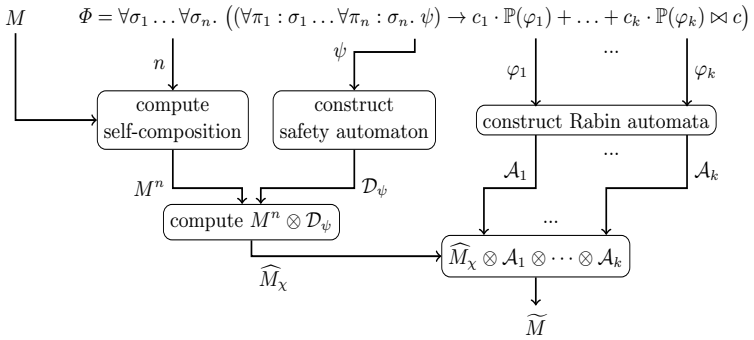

Figure 1. Approximate model checking of PHL formulas of the form (1).

4 Approximate Model Checking

In this section we provide a sound but incomplete procedure for model checking a fragment of PHL. The fragment we consider consists of those PHL formulas that are positive Boolean combinations of formulas of the form

$$\Phi = \forall\sigma_1 \ldots \forall\sigma_n.\ \big(\chi \to c_1 \cdot \mathbb{P}(\varphi_1) + \ldots + c_k \cdot \mathbb{P}(\varphi_k) \bowtie c\big) \tag{1}$$

where $\chi = \forall\pi_1 : \sigma_1 \ldots \forall\pi_n : \sigma_n.\ \psi$ and the formula ψ does not contain path quantifiers and describes an n-safety property (i.e., a safety property on M^n [10]). The PHL formula in Example 2 falls into this class.

The formula ψ contains at most one path variable associated with each scheduler variable in $\{\sigma_1, \ldots \sigma_n\}$. This allows us to use the classical self-composition approach to obtain an automaton for χ. Requiring that ψ describes an n-safety property enables us to consider a deterministic safety automaton for χ which, intuitively, represents the most general scheduler in M^n, such that every scheduler that refines it results in a Markov chain in which all paths satisfy ψ.

Since for every Markov chain C we have $Prob^C(\{\pi \in Paths_{inf}(C) \mid \pi \models \varphi\}) = 1 - Prob^C(\{\pi \in Paths_{inf}(C) \mid \pi \models \neg\varphi\})$, it suffices to consider the case when $\bowtie$ is $\leq$ (or $<$) and $c_i \geq 0$ for each $i = 1, \ldots, k$.

We now describe a procedure for checking whether a given MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$ satisfies a PHL formula Φ of the form (1). If the answer is positive, then we are guaranteed that $M \models \Phi$, but otherwise the result is inconclusive. The method, outlined in Figure 1, proceeds as follows.

We first compute a deterministic safety automaton $\mathcal{D}_\psi$ for the n-hypersafety property ψ. The language of $\mathcal{D}_\psi$ is defined over words in $(S^n)^\omega$. It holds that $w \in \mathcal{L}(\mathcal{D}_\psi)$ if and only if for an arbitrary scheduler assignment Σ it holds that $M, \Sigma, \Pi_w \models \psi$, where Π_w is the path assignment corresponding to the word w. As a second step we construct the n-self-composition MDP M^n, and then build the product of M^n with the deterministic safety automaton $\mathcal{D}_\psi$. The language of the resulting automaton $\widehat{M}_\chi$ consists of the n-tuples of infinite paths in M such that each such tuple satisfies the n-hypersafety property ψ.

After constructing the MDP $\widehat{M}_\chi$, our goal is to check that for every scheduler assignment $\Sigma = ((\sigma_1, \mathfrak{S}_1), \ldots, (\sigma_n, \mathfrak{S}_n))$ for M such that $\overline{\mathfrak{S}} = \mathfrak{S}_1 \parallel \ldots \parallel \mathfrak{S}_n \in \mathit{Sched}(\widehat{M}_\chi)$ the inequality $\sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\overline{\mathfrak{S}}}(\varphi_i)) \leq c$ is satisfied. That would mean, intuitively, that every scheduler assignment that satisfies χ also satisfies the above inequality, which is the property stated by Φ. Note that, if we establish that $\max_{\overline{\mathfrak{S}}=\mathfrak{S}_1\parallel\ldots\parallel\mathfrak{S}_n} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\overline{\mathfrak{S}}}(\varphi_i)) \leq c$, then we have established the above property. Computing exactly the value $\max_{\overline{\mathfrak{S}}=\mathfrak{S}_1\parallel\ldots\parallel\mathfrak{S}_n} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\overline{\mathfrak{S}}}(\varphi_i))$, however, is not algorithmically possible in light of the undecidability results in the previous section. Therefore, we will overapproximate this value by computing a value $c^* \geq \max_{\overline{\mathfrak{S}}=\mathfrak{S}_1\parallel\ldots\parallel\mathfrak{S}_n} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\overline{\mathfrak{S}}}(\varphi_i))$ and if $c^* \leq c$, then we can conclude that the property holds. The value c^* is computed as $c^* = \max_{\widehat{\mathfrak{S}}\in\mathit{Sched}(\widehat{M}_\chi)} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\widehat{\mathfrak{S}}}(\varphi_i))$. For the schedulers $\widehat{\mathfrak{S}}$ considered in this maximization, it is not in general possible to decompose $\widehat{\mathfrak{S}}$ into schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n \in \mathit{Sched}(M)$. Therefore we have that

$$\max_{\widehat{\mathfrak{S}}\in\mathit{Sched}(\widehat{M}_\chi)} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\widehat{\mathfrak{S}}}(\varphi_i)) \geq \max_{\overline{\mathfrak{S}}=\mathfrak{S}_1\parallel\ldots\parallel\mathfrak{S}_n} \sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{\widehat{M}_\chi,\overline{\mathfrak{S}}}(\varphi_i)),$$

which implies that c^* has the desired property. We compute c^* as follows.

We construct deterministic Rabin automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ for the formulas $\varphi_1, \ldots, \varphi_k$. We compute the product $\widetilde{M}$ of the MDP $\widehat{M}_\chi$ constructed earlier and $\mathcal{A}_1, \ldots, \mathcal{A}_k$. Let $\widetilde{S}$ be the set of states of $\widetilde{M}$. We consider each combination of formulas in $\{\varphi_1, \ldots, \varphi_k\}$, i.e., each subset $I \subseteq \{1, \ldots, k\}$ such that $I \neq \emptyset$. For each I, we take the conjunction of the accepting conditions of the deterministic Rabin automata $\mathcal{A}_i$ for $i \in I$ and apply the methods in [7] to compute the so called *success set* $U_I \subseteq \widetilde{S}$ for this conjunction. Intuitively, in the states in U_I there exists a scheduler that can enforce the conjunction of the properties in I.

Finally, we solve the linear program that asks to minimize $\sum_{\widetilde{s}\in\widetilde{S}} x_{\widetilde{s}}$ subject to (i) $x_{\widetilde{s}} \geq 0$ for all $\widetilde{s} \in \widetilde{S}$, (ii) $x_{\widetilde{s}} \geq \sum_{i\in I} c_i$ for all $I \subseteq \{1, \ldots, k\}$ and $\widetilde{s} \in U_I$ and (iii) $x_{\widetilde{s}} \geq \sum_{\widetilde{t}\in\widetilde{S}} \mathbf{P}(\widetilde{s}, a, \widetilde{t}) \cdot x_{\widetilde{t}}$ for all $\widetilde{s} \in \widetilde{S}$ and $a \in \mathit{Act}^n$. If $(x^*_{\widetilde{s}})_{s\in\widetilde{S}}$ is the optimal solution of the linear program, let $c^* = \sum_{\widetilde{s}\in\widetilde{S}} \widetilde{\iota}(\widetilde{s}) \cdot x^*_{\widetilde{s}}$.

If $c^* \leq c$, then for all tuples of schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ we have that if $M_{\mathfrak{S}_1\parallel\ldots\parallel\mathfrak{S}_n} \models \chi$, then for their product $\overline{\mathfrak{S}} = \mathfrak{S}_1 \parallel \ldots \parallel \mathfrak{S}_n$ it holds that $\sum_{i=1}^{k}(c_i \cdot \mathit{Prob}_{M^n,\overline{\mathfrak{S}}}(\varphi_i)) \leq c$, and we conclude that $M \models \Phi$. If, on the other hand, we have $c^* > c$, then the result is inconclusive. When this is the case, we can use bounded model checking to search for counterexamples to formulas of the form (1). For the procedure above, we establish the following result.

Theorem 3 (Complexity). *Given an MDP $M = (S, \mathit{Act}, \mathbf{P}, \iota, \mathsf{AP}, L)$ and a PHL formula Φ of the form (1) the model checking procedure above runs in time polynomial in the size of M and doubly exponential in the size of Φ.*

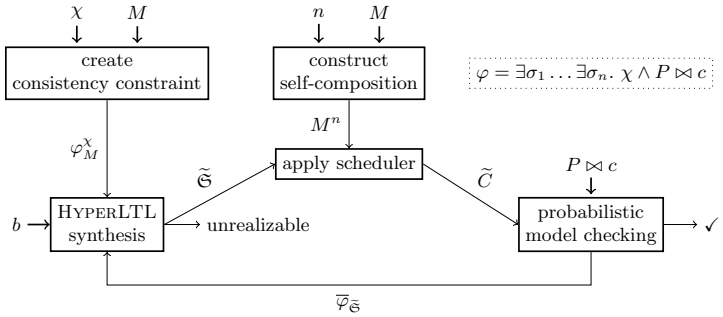

Figure 2. Bounded model checking of MDPs against PHL formulas for the form (2).

5 Bounded Model Checking

We present a bounded model-checking procedure for PHL formulas of the form

$$\Phi = \exists \sigma_1 \ldots \exists \sigma_n.\ \big(\chi \wedge c_1 \cdot \mathbb{P}(\varphi_1) + \ldots + c_k \cdot \mathbb{P}(\varphi_k) \bowtie c\big) \tag{2}$$

where $\chi = \forall \pi_1 : \sigma_1 \ldots \forall \pi_n : \sigma_n.\ \psi$ is in the $\forall^*$ fragment of HyperLTL [14]. An example of a formula in this fragment is the formula in Example 1. By finding a scheduler assignment that is a witness for a PHL formula of the form (2) we can find counterexamples to PHL formulas of the form (1).

Given an MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$, a bound $b \in \mathbb{N}$, and a PHL formula $\Phi = \exists \sigma_1 \ldots \exists \sigma_n.\ \big(\chi \wedge c_1 \cdot \mathbb{P}(\varphi_1) + \ldots + c_k \cdot \mathbb{P}(\varphi_k) \bowtie c\big)$, the *bounded model checking problem for M, b and Φ* is to determine whether there exists a *deterministic finite-memory scheduler* $\widetilde{\mathfrak{S}} = \mathfrak{S}_1 || \ldots || \mathfrak{S}_n$ for M^n composed of deterministic finite-memory schedulers $\mathfrak{S}_i = (Q^i, \delta^i, q_0^i, act_i)$ for M for $i \in \{1, \ldots, n\}$, with $|\mathfrak{S}| = b$ such that $M^n_{\widetilde{\mathfrak{S}}} \models \chi \wedge \sum_{i=1}^{k} (c_i \cdot \mathbb{P}(\varphi_i)) \bowtie c$.

Our bounded model checking procedure employs bounded synthesis for the logic HyperLTL [14] and model checking of Markov chains [19]. The flow of our procedure is depicted in Figure 2. The procedure starts by checking whether there is a scheduler $\widetilde{\mathfrak{S}}$ for M^n composed of schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ for M that satisfies the constraint given by the hyperproperty χ. This is done by synthesizing a scheduler of size b for the HyperLTL formula φ^{χ}_{M} composed of the formula χ, an encoding of M, which ensures that the schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ defining $\widetilde{\mathfrak{S}}$ follow the structure of M, and an additional consistency constraint that requires $\widetilde{\mathfrak{S}}$ to be a composition of n schedulers $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ for M.

If φ^{χ}_{M} is realizable, then the procedure proceeds by applying the synthesized scheduler $\widetilde{\mathfrak{S}}$ to the n-self-composition of the MDP M, which results in a Markov chain $\widetilde{C} = M^n_{\widetilde{\mathfrak{S}}}$. To check whether the synthesized scheduler also satisfies the probabilistic constraint $P \bowtie c$, we apply a probabilistic model checker to the Markov chain $\widetilde{C}$ to compute for each φ_i the probability $Prob_{\widetilde{C}}(\varphi_i)$, and then we evaluate the probabilistic predicate $P \bowtie c$. If $\widetilde{C}$ satisfies $P \bowtie c$, then $M^n_{\widetilde{\mathfrak{S}}} \models \chi \wedge \sum_{i=1}^{k} (c_i \cdot \mathbb{P}(\varphi_i)) \bowtie c$, implying that $M \models \Phi$. If not, we return back to the synthesizer to construct a new scheduler. In order to exclude the scheduler $\widetilde{\mathfrak{S}}$

Table 1. Experimental results from model checking plan non-interference.

Benchmark	MDP Size	# Iterations.	Synthesis time (s)	Model checking time (s)
Arena 3	16	6	12.04	2.68
Arena 4	36	5	17.23	2.19
Arena 5	81	5	18.49	2.76
Arena 7	256	5	19.46	3.01
Arena 9	625	7	168.27	4.72
3-Robots Arena 4	36	9	556.02	4.5

from the subsequent search, a new constraint $\overline{\varphi}_{\widetilde{\mathfrak{S}}}$ is added to φ_M^{χ}. The formula $\varphi_{\widetilde{\mathfrak{S}}}$ imposes the requirement that the synthesized scheduler should be different from $\widetilde{\mathfrak{S}}$. This process is iterated until a scheduler that is a witness for Φ is found, or all schedulers within the given bound b have been checked. The complexity of the procedure is given in the next theorem and follows from complexity of probabilistic model checking [19] and that of synthesis for HyperLTL [14].

Theorem 4 (Complexity). *Given an MDP $M = (S, Act, \mathbf{P}, \iota, \mathsf{AP}, L)$, a bound $b \in \mathbb{N}$, and a* PHL *formula $\Phi = \exists \sigma_1 \ldots \exists \sigma_n . \chi \wedge c_1 \cdot \mathbb{P}(\varphi_1) + \ldots + c_k \cdot \mathbb{P}(\varphi_k) \bowtie c$, the bounded model checking problem for M, b and Φ can be solved in time polynomial in the size of M, exponential in b, and exponential in the size of Φ.*

5.1 Evaluation

We developed a proof-of-concept implementation of the approach in Figure 2. We used the tool BoSyHyper [14] for the scheduler synthesis and the tool PRISM [20] to model check the Markov chains resulting from applying the synthesized scheduler to the self-composition of the input MDP. For our experiments, we used a machine with 3.3 GHz dual-core Intel Core i5 and 16 GB of memory.

Table 1 shows the results of model checking the "plan non-interference" specification introduced in Section 3.1 against MDP's representing two robots that try to reach a designated cell on grid arenas of different sizes ranging from 3-grid to a 9-grid arena. In the last instance, we increased the number of robots to three to raise the number of possible schedulers. The specification thus checks whether the probability for a robot to reach the designated area changes with the movements the other robots in the arena. In the initial state, every robot is positioned on a different end of the grid, i.e., the farthest point from the designated cell.

In all instances in Table 1, the specification with $\varepsilon = 0.25$ is violated. We give the number of iterations, i.e., the number of schedulers synthesized, until a counterexample was found. The synthesis and model checking time represent the total time for synthesizing and model checking all schedulers. Table 1 shows the feasibility of approach, however, it also demonstrates the inherent difficulty of the synthesis problem for hyperproperties.

Table 2. Detailed experimental results for the 3-Robots Arena 4 benchmark.

Iteration	Synthesis (s)	Model checking (s)
1	3.723	0.504
2	3.621	0.478
3	3.589	0.469
4	3.690	0.495
5	3.934	0.514
6	4.898	0.528
7	11.429	0.535
8	60.830	0.466
9	460.310	0.611

Table 2 shows that the time needed for the overall model checking approach is dominated by the time needed for synthesis: The time for synthesizing a scheduler quickly increases in the last iterations, while the time for model checking the resulting Markov chains remains stable for each scheduler. Despite recent advances on the synthesis from hyperproperties [14], synthesis tools for hyperproperties are still in their infancy. PHL model checking will directly profit from future progress on this problem.

6 Conclusion

We presented a new logic, called PHL, for the specification of probabilistic temporal hyperproperties. The novel and distinguishing feature of our logic is the combination of quantification over both schedulers and paths, and a probabilistic operator. This makes PHL uniquely capable of specifying hyperproperties of MDPs. PHL is capable of expressing interesting properties both from the realm of security and from the planning and synthesis domains. While, unfortunately, the expressiveness of PHL comes at a price as the model checking problem for PHL is undecidable, we show how to approximate the model checking problem from two sides by providing sound but incomplete procedures for proving and for refuting universally quantified PHL formulas. We developed a proof-of-concept implementation of the refutation procedure and demonstrated its principle feasibility on an example from planning.

We believe that this work opens up a line of research on the verification of MDPs against probabilistic hyperproperties. One direction of future work is identifying fragments of the logic or classes of models that are practically amenable to verification. Furthermore, investigating the connections between PHL and simulation notions for MDPs, as well studying the different synthesis questions expressible in PHL are both interesting avenues for future work.

References

1. E. Abraham, E. Bartocci, B. Bonakdarpour, and O. Dobe. Probabilistic hyperproperties with nondeterminism. In *Automated Technology for Verification and Analysis, ATVA 2020, Proc.*, 2020.
2. E. Ábrahám and B. Bonakdarpour. HyperPCTL: A temporal logic for probabilistic hyperproperties. In *Quantitative Evaluation of Systems, QEST 2018, Proc.*, 2018.
3. B. Aminof, M. Kwiatkowska, B. Maubert, A. Murano, and S. Rubin. Probabilistic strategy logic. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 32–38, 2019.
4. C. Baier. On model checking techniques for randomized distributed systems. In *Integrated Formal Methods, IFM 2010, Proc.*, volume 6396 of *LNCS*, 2010.
5. C. Baier, N. Bertrand, and M. Größer. On decision problems for probabilistic büchi automata. In *FOSSACS 2008, Proc.*, volume 4962 of *LNCS*, 2008.
6. C. Baier, T. Brázdil, M. Größer, and A. Kucera. Stochastic game logic. *Acta Informatica*, 49(4):203–224, 2012.
7. C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.

8. T. Brázdil, V. Brozek, V. Forejt, and A. Kucera. Stochastic games with branching-time winning objectives. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), Proc.*, pages 349–358. IEEE Computer Society, 2006.
9. 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, POST 2014, Proc.*, volume 8414 of *LNCS*, pages 265–284, 2014.
10. M. R. Clarkson and F. B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010.
11. N. Coenen, B. Finkbeiner, C. Sánchez, and L. Tentrup. Verifying hyperliveness. In *Computer Aided Verification, CAV 2019, Proc.*, volume 11561 of *LNCS*, 2019.
12. R. Dimitrova, B. Finkbeiner, and H. Torfah. Probabilistic hyperproperties of markov decision processes. *CoRR*, abs/2005.03362, 2020.
13. C. Dwork. Differential privacy. In H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed*, pages 338–340. Springer, 2011.
14. B. Finkbeiner, C. Hahn, P. Lukert, M. Stenger, and L. Tentrup. Synthesizing reactive systems from hyperproperties. In *Computer Aided Verification, CAV 2018, Proc., Part I*, volume 10981 of *LNCS*, pages 289–306, 2018.
15. B. Finkbeiner, C. Hahn, and M. Stenger. Eahyper: Satisfiability, implication, and equivalence checking of hyperproperties. In *Computer Aided Verification, CAV 2017, Proc., Part II*, volume 10427 of *LNCS*, pages 564–570, 2017.
16. B. Finkbeiner, M. N. Rabe, and C. Sánchez. Algorithms for model checking HyperLTL and HyperCTL ˆ*. In *Computer Aided Verification - 27th International Conference, CAV 2015, Proc., Part I*, volume 9206 of *LNCS*, pages 30–48, 2015.
17. J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1982.
18. J. W. Gray. Toward a mathematical foundation for information flow security. *J. Comput. Secur.*, 1(3–4):255–294, May 1992.
19. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking: Advances and applications. In *Formal System Verification*. Springer, 2017.
20. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification, CAV 2011, Proc.*, volume 6806 of *LNCS*, pages 585–591, 2011.
21. K. R. O'Neill, M. R. Clarkson, and S. Chong. Information-flow security for interactive programs. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006)*, pages 190–201. IEEE Computer Society, 2006.
22. A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, 1977.
23. A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop, CSFW '00*, pages 200–214. IEEE Computer Society, 2000.
24. D. M. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. *J. Comput. Secur.*, 7(1), 1999.
25. Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic. Statistical verification of hyperproperties for cyber-physical systems. *ACM Trans. Embedded Comput. Syst.*, 18(5s):92:1–92:23, 2019.