Decentralized Planning Using Probabilistic Hyperproperties

Francesco Pontiggia
TU Wien
Vienna, Austria
francesco.pontiggia@tuwien.ac.at

Filip Macák
Brno University of Technology
Brno, Czech Republic
imacak@fit.vut.cz

Roman Andriushchenko
Brno University of Technology
Brno, Czech Republic
iandri@fit.vut.cz

Michele Chiari
TU Wien
Vienna, Austria
michele.chiari@tuwien.ac.at

Milan Češka
Brno University of Technology
Brno, Czech Republic
ceskam@fit.vutbr.cz

ABSTRACT

Multi-agent planning under stochastic dynamics is usually formalised using decentralized (partially observable) Markov decision processes (MDPs) and reachability or expected reward specifications. In this paper, we propose a different approach: we use an MDP describing how a single agent operates in an environment and probabilistic hyperproperties to capture desired temporal objectives for a set of decentralized agents operating in the environment. We extend existing approaches for model checking probabilistic hyperproperties to handle temporal formulae relating paths of different agents, thus requiring the self-composition between multiple MDPs. Using several case studies, we demonstrate that our approach provides a flexible and expressive framework to broaden the specification capabilities with respect to existing planning techniques. Additionally, we establish a close connection between a subclass of probabilistic hyperproperties and planning for a particular type of Dec-MDPs, for both of which we show undecidability. This lays the ground for the use of existing decentralized planning tools in the field of probabilistic hyperproperty verification.

KEYWORDS

Probabilistic Hyperproperties; Decentralized Planning; Markov Decision Processes; Abstraction Refinement; Self-composition.

ACM Reference Format:
Francesco Pontiggia, Filip Macák, Roman Andriushchenko, Michele Chiari, and Milan Češka. 2025. Decentralized Planning Using Probabilistic Hyperproperties. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

1 INTRODUCTION

Decentralized planning under uncertainty. Markov decision processes (MDPs) are the ubiquitous model to describe sequential decision making (or planning) of a single agent operating in an uncertain environment: the outcomes of the agent's actions are determined by a probability distribution over the successor states. Decentralised MDPs (Dec-MDPs) naturally extend MDPs to the situation in which multiple agents are deployed in the same environment, each state of the environment generating a unique set of observations for the agents. Dec-MDPs can be seen as a special case of decentralised partially observable MDPs (Dec-POMDPs) [15], a classical model for decentralized planning under state uncertainty where the full information of the current state is not available to the agents. A classical synthesis task in Dec-(PO)MDPs is to compute for each agent a policy that maximises a given joint objective. Existing tool-supported approaches for Dec-(PO)MDPs typically handle only simple objectives such as time-bounded reachability or reward [38] or infinite-horizon (discounted) rewards [50].

Planning of multi-agent systems using temporal logics. In order to effectively formalise more complicated tasks and constraints, various temporal logics have been proposed in the context of multi-agent systems (MAS). The most prominent are *Alternating Temporal Logic* (ATL) [2] and *Strategy Logic* [21, 41]. Extensions deal with partial observability [13, 14, 16, 19], stochastic systems [3, 11, 22, 36, 37], and hyperproperties for MAS [17]. Belardinelli et al. [12] study model checking probabilistic variants of ATL on stochastic concurrent games with imperfect information. Despite their expressive power, most notably the ability to express coalitions or adversarial behaviours, these logics do not specialize to the particular type of partial observability arising in *decentralized* MAS.

Linear Temporal Logic (LTL) [44] has too been used to specify objectives in decentralized planning under uncertainty [47, 52], but only via manually splitting the global goal into separate, agent-local goals.

Probabilistic hyperproperties. In this paper, we propose a different approach for decentralized planning under uncertainty. The key idea is to use Probabilistic Hyperproperties (PHs) [1, 26] as the specification language. Hyperproperties [24] extend the traditional notion of trace properties by specifying requirements that put in relation multiple execution traces of a system at once. To this extent, they consider the *self-composition* of several copies of the same system, where the synchronised (step-wise paired) execution traces are analysed. For probabilistic systems, hyperproperties either relate the probability measures of different sets of traces (probabilistic computation trees) or constrain the joint probability of sets of computation trees [30]. PHs fit perfectly in a decentralized planning setting, where each different execution is interpreted as a different agent, and the hyperformula specifies a shared goal.

Our approach is inspired by pioneering studies showing that PHs have novel applications in robotics and planning [4, 26, 48]. In order

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

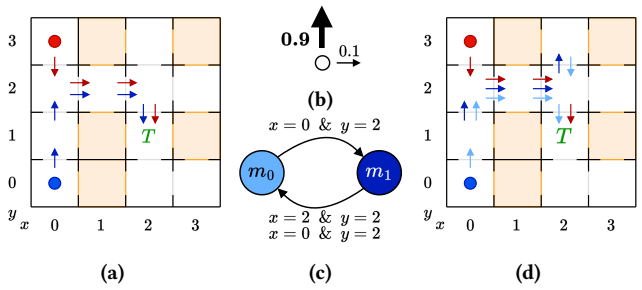

Figure 1: Running example on a 4x4 grid with 2 agents and a target cell T. The goal is to ensure that the red agent reaches T first. (a) An illustration of the optimal memoryless policies. (b) The considered slipping factor. (c) A 2-node controller representing the optimal policy for the blue agent (the labels describes the transitions between the nodes). (d) An illustration including the optimal policy for the blue agent.

to formalise a decentralized multi-agent planning problem, we use an MDP to describe how a single agent operates in the environment and a PH to capture relative temporal constraints on the agents. We exemplify our approach using the following planning problem.

Motivating example. Consider the 4x4 grid depicted in Fig. 1a, where obstacles are in orange. In every white cell, an agent can move in any of the four cardinal directions and will slip right from the selected course with a probability of 0.1 (see Fig. 1b). We also assume that each action can cause with some probability an irrecoverable failure. Consider two agents, a_0 and a_1 that start in the bottom left and the upper left corners, respectively, and seek to reach the target location $T : x = 2 \,\&\, y = 1$ with maximal probability (i.e. as soon as possible). Additionally, we require agent a_1 to reach the target first. The agents cannot communicate or observe the location of each other. We can see this problem as a more complex variant of the *decentralized meeting* problem [33]. The specification can be encoded as a PH as follows: $\exists\,(\hat{\sigma}_0\,\hat{\sigma}_1)\,.\,\forall \hat{s}_0 \in \{s_0\}(\hat{\sigma}_0)\,\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1)\,.\,\mathbb{P}_{\max}\big((\Diamond\, T_{\hat{s}_0}) \wedge (\Diamond\, T_{\hat{s}_1}) \wedge \Box\,(T_{\hat{s}_0} \implies T_{\hat{s}_1})\big)$ where $s_0 : x = 0 \,\&\, y = 0$ and $s_1 : x = 0 \,\&\, y = 3$. Intuitively, this formula requires two independent (i.e. decentralized) policies $\hat{\sigma}_1$ and $\hat{\sigma}_2$ for the MDP environment, one for agent $\hat{s}_0$ that starts in s_0, and one for agent $\hat{s}_1$ that starts in s_1. The inner part requires maximising the probability that both agents reach the target, but agent a_1 arrives first. Such decentralized requirement is not expressible in classical logics (i.e. PCTL or PLTL) because they do not allow for declaring multiple agents (via state variables $\hat{s}_i$) nor multiple policies (via policy variables $\hat{\sigma}_i$). Fig. 1a shows the optimal memoryless policies. There is, however, a better memoryful strategy depicted in Fig. 1c as a 2-node controller for agent a_0. The corresponding policy is illustrated in Fig. 1d. In this case, agent a_0 takes a detour in cell $x = 2 \,\&\, y = 2$ to give a_1 more time to reach the meeting cell. This seems straightforward, however, we still want the agents to reach the target, and therefore, agent a_0 only takes the detour if they have not slipped in the cell $x = 0 \,\&\, y = 2$, otherwise, they go straight to the target.

Contributions. We propose a new logic, PHyperLTL, as a specification language for planning in decentralized, uncertain environments. We propose two approaches to construct decentralized policies satisfying PHyperLTL specifications that both rely on an automata-based synchronised product construction. For the full logic, we present an abstraction-refinement algorithm that constructs the optimising policy for the synchronised product MDP and attempts to factorise it into individual policies for each agent. If factorisation is not possible, we refine the abstraction to remove such a policy from the design space and continue the search.

For a fragment of PHyperLTL, the self-composition allows for translating it to a simple reachability problem for Dec-MDPs solvable by any off-the-shelf Dec-(PO)MDP solver. As a result, we establish a close connection between a subclass of probabilistic hyperproperties and decentralized planning. Using a novel undecidability proof for our logic, we obtain undecidability of infinite-horizon planning in locally fully observable Dec-MDPs and thus strengthen existing undecidability results. In the experimental evaluation, we demonstrate on a broad set of benchmarks that (i) PHyperLTL can succinctly capture complex temporal constraints over policies and their executions, (ii) the proposed synthesis algorithm can solve challenging planning problems, and (iii) the algorithm, in many cases, outperforms the state-of-the-art solver on Dec-MDPs.

Related work

Decentralized planning with complex shared objectives. There have been several attempts at finding formalisms for expressing objectives more general than reachability in decentralized planning.

Neary et al. [43] use Mealy machines to encode a shared objective which is split into subtasks assigned to each agent, enabling decentralized reinforcement learning. They do not use logics, but rather automata, to express requirements.

Schuppe and Tumova [47] use LTL over finite traces (LTLf) formulae to capture safety and liveness specifications for agents modelled as MDPs. They assume the global objective to be factored into individual agent tasks, plus a smaller task requiring their interaction. They use stochastic games to synthesize local policies separately. Such policies are not completely decentralized: the planner assumes that agents can share some limited information (called *advisers*) on their current state, violating local observability.

LTL specifications have been considered in the context of reinforcement learning for Dec-POMDPs: Zhu et al. [52] propose a hierarchical approach that computes temporal equilibrium strategies via a parity game and encode the individual reward machines [43] used in the learning process.

These works propose different approaches to decompose the planning problem to tackle its high complexity. On the contrary, we perform centralized planning but tackle its complexity through abstraction refinement [5, 20]. Moreover, these approaches use logics to specify objectives, but aspects related to the multi-agent nature of the problem are expressed externally or in the underlying operational model because LTL is not sufficiently expressive.

Verification of Hyperproperties. The deterministic hyperlogics HyperLTL and HyperCTL* [23] extend LTL and CTL* with *path quantifiers* that relate multiple execution traces of the same system. They can specify important security properties such as *noninterference* [40]. Beutner and Finkbeiner [18] show that model checking of a HyperLTL fragment can be cast as a *nondeterministic planning problem* for Qualitative Dec-POMDPs. HyperSL [17] is a

variant of Strategy Logic that can relate multiple execution paths and was used for expressing planning objectives in MAS. In contrast to PHyperLTL, it does not support probabilistic reasoning and assumes perfect information.

PHyperLTL can be seen as a probabilistic variant of HyperLTL, that replaces path quantifiers with probability operators and quantifies over policies. There are two other temporal logics for PHs: HyperPCTL [1] and PHL [26]. They have been devised mainly with the purpose of verification, while we designed PHyperLTL to conveniently specify planning problems. We compare them in Sec. 3.3.

HyperProb [27] is an SMT-based tool for model checking probabilistic hyperproperties with limited scalability [4]. Two approaches tackle the high intractability of the problem: [28] proposes a statistical method for bounded HyperLTL model checking on MDPs based on sampling policies and traces, and [4] introduces an abstraction refinement algorithm for the synthesis of formulae involving a comparison of probability measures for different computation trees and additional qualitative structural constraints.

2 BACKGROUND

A *distribution* over a countable set A is a function $\mu\colon A \to [0,1]$ s.t. $\sum_{a\in A}\mu(a)$=1. The set $Distr(A)$ contains all distributions on A. Single-agent planning problems in a fully observable stochastic environment can be represented as MDPs.

Definition 2.1 (MDP). A *Markov decision process (MDP)* is a tuple $M = (S, Act, \mathcal{P}, L)$ with a finite set S of states, a finite set Act of actions, a transition function $\mathcal{P}\colon S\times Act \to Distr(S)$, and a labelling function $L\colon S \to 2^{AP}$ where AP is a finite set of atomic propositions. We denote $\mathcal{P}(s,a,s') \coloneqq \mathcal{P}(s,a)(s')$. A *Markov chain (MC)* is an MDP with $|Act| = 1$, denoted as $(S, \mathcal{P}, L)$ with $\mathcal{P} : S \to Distr(S)$.

Note that we do not specify initial states: these will be derived from the specification. A finite *path* of an MDP M is a sequence $\pi = s^0a^0s^1a^1\dots s^n$ where $\mathcal{P}(s^i,a^i,s^{i+1}) > 0$ for $0 \le i < n$. $Paths^M$ denotes the set of all finite paths of M.

A deterministic *policy* (or *controller*, *scheduler*) is a function $\sigma\colon Paths^M \to Act$. Policy σ is *memoryless* if the action selection only depends on the last state of the path, i.e. σ maps each state $s \in S$ to an action $\sigma(s)$. Policy σ for MDP $M = (S, Act, \mathcal{P}, L)$ *induces* an MC $M^\sigma = (Paths^M, \mathcal{P}^\sigma, L^\sigma)$ where for all paths $\pi \in Paths^M$ ending with state s we set $\mathcal{P}^\sigma(\pi, \pi a s') = \mathcal{P}(s,a,s')$ if $a = \sigma(\pi)$ and 0 otherwise, and $L^\sigma(\pi) = L(s)$.

For an MDP M and a policy σ, $\mathbb{P}_{M^\sigma}(s \models \Diamond T)$ denotes the probability of reaching from state $s \in S$ some state in a set $T \subseteq S$ of target states. Standard techniques [8] can efficiently compute policy $\sigma_{\max}$ that maximises this probability[1].

To represent several agents acting in the same environment defined as an MDP, we introduce the MDP self-composition. In the following, given a set A, we denote its k-ary set product as $A^k = \bigtimes_{i=1}^k A$, and its element tuples in bold face as $\mathbf{a} = (a_1, \dots, a_k)$.

Definition 2.2 (MDP self-composition). Given a positive integer m and an MDP $M = (S, Act, \mathcal{P}, L)$, its *$m$-self-composition* is an MDP $M^m = (S^m, Act^m, \mathcal{P}^m, L^m)$ where $\mathcal{P}^m\colon S^m \times Act^m \to Distr(S^m)$ is s.t. $\mathcal{P}^m(\mathbf{s},\mathbf{a},\mathbf{s'}) = \prod_{i=1}^m \mathcal{P}(s_i,a_i,s_i')$, and $L^m\colon S^m \to (2^{AP})^m$ where $L^m(\mathbf{s}) = (L(s_1),\dots,L(s_m))$.

[1]We do not consider rewards here but our implementation supports them.

Intuitively, m-self-composition M^m describes m independent instances (so-called *replicas*) of M that evolve simultaneously. The notions of paths, policies, and reachability probability trivially derive from those defined for MDPs. On the other hand, stochastic decentralized planning problems are usually modeled as *decentralized POMDPs (Dec-POMDPs)*, where several agents jointly interact with a stochastic environment that they only partially observe. In this paper, we assume agents can fully observe their state but not the state of other agents. Thus, we introduce *decentralized MDPs*: the definition is derived from a standard Dec-POMDP definition (cf. [10, 15]) stripped of observations and rewards.

Definition 2.3. A *factored, observation-independent, locally fully observable Decentralised MDP (Dec-MDP)* with m agents is a tuple $\mathcal{M} = (S, Act, \mathcal{P}, L)$ where the state space $S = \bigtimes_{i=1}^m S_i$ can be factored into m sets of agent-local states, $Act = \bigtimes_{i=1}^m Act_i$ is the set of *joint actions*, where each Act_i is a set of actions available to agent i, and $\mathcal{P}$ and L are as in MDPs.

The m-self-composition M^m is a Dec-MDP. A (deterministic, memoryless) policy for agent i is a mapping $\sigma_i\colon S_i \to Act_i$, and a *joint policy* σ is a tuple $\sigma = (\sigma_1, \dots \sigma_m)$ of individual policies. For general (i.e., not locally fully observable) Dec-MDPs, the problem of finding a joint policy that maximises the probability of reaching a target state is undecidable [15]. We will strengthen this undecidability result to the class of locally fully observable Dec-MDPs in Sec. 5.

2.1 Temporal Logic

The syntax of LTL [9] is given as

$$\varphi \coloneqq a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi \mathbin{\mathsf{U}} \varphi \qquad \text{where } a \in AP$$

for a finite set of atomic propositions AP. We consider semantics over *infinite traces*, i.e., infinite sequences of subsets of AP. Intuitively, a holds in the current trace position if a is part of the subset of AP labeling it, $\bigcirc\varphi$ if φ holds in the next position, and $\varphi_1 \mathbin{\mathsf{U}} \varphi_2$ holds if φ_1 holds until φ_2 does. Propositional operators $\wedge$ and $\neg$ have their usual semantics. We also use the standard abbreviations $\Diamond\varphi = \top \mathbin{\mathsf{U}} \varphi$, and $\Box\varphi = \neg\Diamond\neg\varphi$ meaning that φ will resp. *eventually* and *always* hold. An LTL formula φ can be checked on a MC by building a deterministic Rabin automaton $\mathcal{A}_\varphi$ that accepts exactly traces that satisfy φ [9].

Definition 2.4 (DRA). A deterministic Rabin automaton (DRA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where Q is a finite set of states, Σ is a finite input alphabet ($\Sigma = 2^{AP}$ when checking LTL formulae), $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $Acc \subseteq 2^Q \times 2^Q$ is the *Rabin acceptance condition*.

A *run* of $\mathcal{A}$ reading the infinite word $a_0a_1\dots \in \Sigma^\omega$ is a sequence of states $\rho = q_0q_1\dots \in Q^\omega$ such that $\delta(q_i,a_i) = q_{i+1}$ for all $i \ge 0$. Run ρ is accepting if there exists a pair $(L,K) \in Acc$ such that states in L appear only finitely often in ρ, while those in K appear infinitely often. $\mathcal{A}$ accepts the language of all and only words whose run is accepting. Let $\mathbb{P}_M(s \models \varphi)$ be the probability that the traces generated by labels of a MC M starting from a state s satisfy an LTL formula φ. Computing $\mathbb{P}_M(s \models \varphi)$ reduces to the reachability problem of a set of states U_{Acc} in the *synchronised product* between M and $\mathcal{A}_\varphi$. We refer to [9] for more details.

3 PHyperLTL

In this section, we introduce the syntax and semantics of Probabilistic HyperLTL (PHyperLTL). The logic allows for expressing probabilistic hyperproperties on MDPs so that their model-checking problem can be easily interpreted as a planning problem.

3.1 PHyperLTL Syntax

Let $\mathcal{V}_{pol}$ and $\mathcal{V}_{state}$ be sets of policy and state variables, respectively, and let $M = (S, Act, \mathcal{P}, L)$ be an MDP. Formulae in PHyperLTL are constructed with the syntax

$$\exists\,(\hat{\sigma}_1 \dots \hat{\sigma}_n)\,.\,Q_1\hat{s}_1 \in I_1(\hat{\sigma}_i) \dots\, Q_m\hat{s}_m \in I_m(\hat{\sigma}_j)\,.\,\phi$$
$$\phi \coloneqq \phi \wedge \phi \;\mid \neg\phi \;\mid \mathbb{P}(\varphi) \bowtie c$$

where $n \leq m$ and each $\hat{\sigma}_i$ ($\hat{s}_i$) is a policy (state) variable from $\mathcal{V}_{pol}$ ($\mathcal{V}_{state}$); $Q \in \{\exists, \forall\}$, $\mathbb{P}(\cdot)$ is the *probabilistic operator*; $c \in [0, 1]$; each $I_i \subseteq S$ is a subset of states; and φ is an LTL formula over the set $AP_{\mathcal{V}_{state}} = \{a_{\hat{s}} \mid a \in AP, \hat{s} \in \mathcal{V}_{state}\}$ of atomic propositions tagged with state variables (cf. Sec. 2.1).

A PHyperLTL formula starts with a prefix that asserts the existence of n policies, $\hat{\sigma}_1, \dots, \hat{\sigma}_n$, that will be the interest of the planning problem. Then, we have m agent quantifiers: $Q_i\hat{s}_i \in I_i(\hat{\sigma}_j)$ states that agent i starts in state $\hat{s}_i$ and is controlled by policy $\hat{\sigma}_j$, where a possible initial state is quantified existentially or universally from some subset $I_i \subseteq S$. Multiple initial states are useful to express e.g. uncertainty regarding the initial state. Note that $m > n$ implies that some agents share the same policy. A Boolean combination ϕ of *probability constraints* describes the desired behaviour of agents. A probability constraint $\mathbb{P}(\varphi) \bowtie c$ requires the probability of the set of paths that satisfy φ *in the m-self-composition* of the MDP as seen by each agent to meet $\bowtie c$. Atomic propositions in φ are indexed with state variables identifying the agent for which to evaluate them. Thus, φ can be seen as a LTL formula over the self-composition.

A formula is *well-formed* if all policy variables are bound by a policy quantifier—i.e., for every occurrence of $\hat{\sigma}_j$ in e.g. $Q_i\hat{s}_i{\in}I_i(\hat{\sigma}_j)$ there is a corresponding $\exists\hat{\sigma}_j$—and each state variable $\hat{s}_j$ that occurs in φ is bound to some state quantifier $Q\hat{s}_j$.

PHyperLTL can be straightforwardly extended with a reward operator [29]. Also, as common in PCTL planning, we consider maxi/minimization objectives $\mathbb{P}_{max}$ ($\mathbb{P}_{min}$) instead of the threshold.

3.2 Formal Semantics

Let $n = |\mathcal{V}_{pol}|$ be the number of policy variables. A *policy assignment* for $\mathcal{V}_{pol}$ is a vector $\Sigma = ((\hat{\sigma}_1, \sigma_1), \dots, (\hat{\sigma}_n, \sigma_n))$, that assigns policies $(\sigma_1 \dots \sigma_n)$ to all variables $(\hat{\sigma}_1 \dots \hat{\sigma}_n)$. We denote with $\Sigma(\hat{\sigma})$ the policy assigned to variable $\hat{\sigma}$. Probability constraints are evaluated in the context of MDP M, a policy assignment Σ, a sequence of m policies η, and a state $s = (s_1, \dots, s_m)$ of the m-self-composition. We use $()$ to denote the empty sequence and $\circ$ for the concatenation operator. M satisfies formula $\exists\,(\hat{\sigma}_1 \dots \hat{\sigma}_n)\,.\,\Phi$ iff there exists a policy assignment Σ such that $M, \Sigma, (), () \vDash \Phi$, where the satisfaction relation $\vDash$ is defined as follows (we omit M, Σ for clarity):

$$\eta, s \vDash \exists\hat{s} \in I(\hat{\sigma})\,.\,\Phi \quad \text{iff} \bigvee_{s' \in I} \eta \circ \Sigma(\hat{\sigma}), s \circ s' \vDash \Phi[m+1/\hat{s}]$$
$$\eta, s \vDash \forall\hat{s} \in I(\hat{\sigma})\,.\,\Phi \quad \text{iff} \bigwedge_{s' \in I} \eta \circ \Sigma(\hat{\sigma}), s \circ s' \vDash \Phi[m+1/\hat{s}]$$
$$\eta, s \vDash \phi_1 \wedge \phi_2 \quad \text{iff } \eta, s \vDash \phi_1 \text{ and } \eta, s \vDash \phi_2$$
$$\eta, s \vDash \neg\phi \quad \text{iff } \eta, s \nvDash \phi$$
$$\eta, s \vDash \mathbb{P}(\varphi) \bowtie c \quad \text{iff } \Pr\big(\{\pi \in \omega Paths(M_\eta^m, s) \mid M_\eta^m, \pi \vDash \varphi\}\big) \bowtie c$$

where $m = |\eta|$, and $\Phi[m+1/\hat{s}]$ is the formula obtained by replacing every occurrence of a tagged proposition $p_{\hat{s}}$ with the index $m+1$; M_η^m is the MC generated by applying the m-tuple of policies η to the m-self-composition of M, and $\omega Paths(M_\eta^m, s)$ is the set of its infinite paths starting from state s. Given $\pi = s^0 s^1 \dots \in \omega Paths(M_\eta^m, s)$, we set $\pi^i = s^i$, and $\pi[i] = s^i s^{i+1} \dots$. It is well-known that sets $\{\pi \in \omega Paths(M_\eta^m, s) \mid M_\eta^m, \pi \vDash \varphi\}$, where φ is an LTL formula, are measurable [9]. The φ-formulae follow the usual LTL trace semantics, which we report below:

$$M_\eta^m, \pi \vDash p_j \quad \text{iff } p \in L^m(\pi^0)(j)$$
$$M_\eta^m, \pi \vDash \varphi_1 \wedge \varphi_2 \quad \text{iff } M_\eta^m, \pi \vDash \varphi_1 \text{ and } M_\eta^m, \pi \vDash \varphi_2$$
$$M_\eta^m, \pi \vDash \neg\varphi \quad \text{iff } M_\eta^m, \pi \nvDash \varphi$$
$$M_\eta^m, \pi \vDash \bigcirc\varphi \quad \text{iff } M_\eta^m, \pi[1] \vDash \varphi$$
$$M_\eta^m, \pi \vDash \varphi_1 \,\mathsf{U}\, \varphi_2 \quad \text{iff } \exists j \geq 0\,.\,M_\eta^m, \pi[j] \vDash \varphi_2 \wedge \forall i \in [0, j)\,.\,M_\eta^m, \pi[i] \vDash \varphi_1$$

3.3 Related Logics

Policy quantification and $\mathbb{P}$-operators are common to PHL [26], HyperPCTL [1], and PHyperLTL. PHL and HyperPCTL also offer alternation of policy quantifiers and comparison of $\mathbb{P}$-operators. Quantifier alternation is useful for specifying adversarial problems, but in this paper we assume a cooperative environment. Moreover, it significantly complicates planning. Comparison of $\mathbb{P}$-operators could be added to PHyperLTL by integrating the abstraction refinement approach by [4]. PHyperLTL and PHL allow for full LTL formulae in $\mathbb{P}$-operators, a feature particularly suitable for planning objectives. HyperPCTL supports nested $\mathbb{P}$-operators. PHyperLTL and PHL exclude them since they further complicate planning without offering significant benefits. PHyperLTL and HyperPCTL support state quantification, which is useful to express uncertainty in initial states of the agents. State quantification can be expressed in Probabilistic Hyper Logic (PHL) with nonprobabilistic constraints: PHyperLTL is thus strictly included in PHL.

3.4 Undecidability

Undecidability was proved for HyperPCTL [30] and PHL [26] by encoding the emptiness problem for probabilistic Büchi automata, which is undecidable [7]. The proof for HyperPCTL does not apply to PHyperLTL, because it employs a formula containing nested probability operators. The formula built in the proof for PHL contains some HyperCTL* parts, but it can be rewritten as a PHyperLTL formula containing, however, at least *two* probability constraints. In this section, we sketch an alternative proof that uses a PHyperLTL formula with just *one* probability constraint. We thus prove the undecidability of a smaller logic fragment, allowing us to obtain an undecidability result for locally fully observable Dec-MDPs in Sec. 5, which could not be easily obtained from the other proofs.

Theorem 3.1. PHyperLTL *model checking is undecidable.*

The proof is based on a reduction of the emptiness problem for Probabilistic Finite Automata (PFA), which is undecidable [25, 32, 42], to model checking of a PHyperLTL formula on an MDP. A PFA can be seen as an MDP whose actions are input symbols, and it accepts finite strings (i.e., sequences of actions) for which it has a probability $> \lambda \in \mathbb{Q}$ to reach an accepting state. The emptiness problem consists of deciding whether such a string exists.

We build an MDP that is the union of two disjoint MDPs M_1, that encodes the PFA, and M_2, that is a deterministic finite-state automaton (FSA) with the same input alphabet as the PFA. A PHyperLTL formula Φ states the existence of two policies σ_1 and σ_2, respectively controlling M_1 and M_2, which are synchronised on the actions they take. Moreover, Φ constrains σ_1 to reach an accepting state of the PFA. One single $\mathbb{P}$-expression states that these constraints must be true with probability $\geq \lambda$. If two such policies exist, since M_2 is a deterministic FSA, we can extract from σ_2 a string that is accepted by the PFA. The full proof can be found in the technical report [45, App. A].

4 PLANNING BY ABSTRACTION REFINEMENT

Assume an MDP M and a well-formed PHyperLTL formula containing n policy variables, m state variables and a single probability expression $\mathbb{P}_{\max}(\varphi)$. In other words, we are interested in computing an n-tuple of policies that maximises the probability of satisfying an LTL formula φ containing m state variables. The definition below presents the key device used to recast the satisfaction probability of φ into a reachability probability.

Definition 4.1 (Synchronised product). Assume a Markov decision process $M = (S, Act, \mathcal{P}, L)$, an LTL formula φ with m state variables and a DRA $\mathcal{A}_\varphi = (Q, (2^{AP})^m, \delta, q_0, Acc)$ accepting traces that satisfy φ. The *synchronised product* of M and $\mathcal{A}_\varphi$ is $\mathcal{D}(M, \varphi) = (S^m \times Q, Act^m \times \{\alpha\}, \mathcal{P}_\mathcal{D}, L_\mathcal{D})$, where α is a fresh action for transitions of $\mathcal{A}_\varphi$ and for all $\mathbf{s}, \mathbf{s}' \in S^m$, $\mathbf{a} \in Act^m$, $\mathcal{P}_\mathcal{D}((\mathbf{s}, q), (\mathbf{a}, \alpha), (\mathbf{s}', q')) = \mathcal{P}^m(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ where $\delta(q, L^m(\mathbf{s})) = q'$. $L_\mathcal{D}(\mathbf{s}, q) = L^m(\mathbf{s})$ for all $\mathbf{s} \in S^m$ and $q \in Q$.

We will denote $\mathcal{D}(M, \varphi)$ simply as $\mathcal{D}$ whenever M, φ are clear from the context. Intuitively, $\mathcal{D}$ is the product of the m-self-composition M^m with $\mathcal{A}_\varphi$. Each state tuple of $\mathcal{D}$ encodes a state of each of the m replicas of M modelling the m agents, and a state of $\mathcal{A}_\varphi$. The actions of $\mathcal{D}$ synchronise m actions of M with the (deterministic) update of $\mathcal{A}_\varphi$ on m-tuples of labels of M. The theorem below asserts that computing the satisfaction probability of φ on M is equivalent to computing a reachability probability on $\mathcal{D}$. The proof follows directly from Def. 2.2, Def. 4.1 and [9].

Theorem 4.2. *Assume $\mathbf{s} \in S^m$ and a policy $\sigma \in \Sigma^{M^m}$. Then*

$$\mathbb{P}_{(M^m)^\sigma}(\mathbf{s} \vDash \varphi) = \mathbb{P}_{\mathcal{D}^{\sigma'}}((\mathbf{s}, \delta(q_0, L^m(\mathbf{s}))) \models \Diamond U_{\mathcal{A}_\varphi})$$

where $\sigma'(\pi) = (\sigma(\pi), \alpha)$ for any path π, and $U_{\mathcal{A}_\varphi}$ is the success set of $\mathcal{A}_\varphi$'s acceptance condition.

4.1 Synchronised Product as an Abstraction

Assume the PHyperLTL specification from above where we seek an n-tuple of policies that maximises the probability of satisfying an LTL formula φ containing m state variables. With standard methods for MDP model checking, we can compute policy $\overline{\sigma}_{\max}$ maximising $\mathbb{P}(\Diamond U_{\mathcal{A}_\varphi})$ on $\mathcal{D}$. $\overline{\sigma}_{\max}$ maps state tuples $(s_1, \ldots, s_m, q)$ of $\mathcal{D}$ to action tuples $(a_1, \ldots, a_m, \alpha)$. Under two conditions, $\overline{\sigma}_{\max}$ can be *factorised* into an n-tuple $(\sigma_1, \ldots, \sigma_n)$ of policies for individual agents. Intuitively, a policy $\overline{\sigma}$ for $\mathcal{D}$ is a centralised policy that makes decisions while considering the states of all agents and thus may violate the local observability assumption. Additionally, $\overline{\sigma}$ ignores the fact that m might be larger than n, i.e. the case where multiple agents (replicas) must adhere to the same policy. The following definition formalises these concepts.

Definition 4.3 (Policy consistency). Let $\overline{\sigma}$ be a policy for $\mathcal{D}$. We say that $\overline{\sigma}$ *violates local observability* if for some agent i there exist two states $\mathbf{s}$ and $\mathbf{s}'$ s.t. $s_i = s'_i$ and $\overline{\sigma}(\mathbf{s})(i) \neq \overline{\sigma}(\mathbf{s}')(i)$. We say that $\overline{\sigma}$ *violates policy bindings* if for two agents i and j, $i \neq j$, bound to the same policy variable, there exist two (not necessarily different) states $\mathbf{s}$ and $\mathbf{s}'$ s.t. $s_i = s'_j$ and $\overline{\sigma}(\mathbf{s})(i) \neq \overline{\sigma}(\mathbf{s}')(j)$. We say that $\overline{\sigma}$ is *inconsistent* if it violates local observability or policy bindings.

We remark that every n-tuple of policies for M can be mapped to a policy for $\mathcal{D}$, but only a consistent policy $\overline{\sigma}$ can be factorised into an n-tuple of policies $(\sigma_1, \ldots, \sigma_n)$ as follows: $\sigma_i(s) = \overline{\sigma}(\mathbf{s})(j)$ where j is the index of an agent bound to i-th policy $\hat{\sigma}_i$ and $\mathbf{s}_j = s$. This makes $\mathcal{D}$ a proper abstraction for M and φ.

4.2 Abstraction Refinement

Assume we require[2] $\mathbb{P}(\varphi) > \lambda$. Since $\mathcal{D}$ is an abstraction, i.e. $\Sigma^\mathcal{D}$ encompasses all policy tuples for M and φ, the value $V(\overline{\sigma}_{\max})$ of the maximizing policy for $\mathcal{D}$ is an over-approximation of the maximal value achievable by a policy n-tuple. If $V(\overline{\sigma}_{\max}) > \lambda$ and $\overline{\sigma}_{\max}$ is consistent, then it can be factorized into an n-tuple of policies that satisfy the property. If $V(\overline{\sigma}_{\max}) \leq \lambda$, then no n-tuple of policies can satisfy the property. If $V(\overline{\sigma}_{\max} > \lambda)$ and $\overline{\sigma}_{\max}$ is inconsistent, we optimistically obtain a policy by randomly resolving the inconsistencies in $\overline{\sigma}_{\max}$, and check if the resulting policy n-tuple satisfies the threshold. If we fail, no conclusion can be deduced. Inspired by the abstraction refinement approach of [5, 20], we *split* $\mathcal{D}$ into sub-models, refining the abstraction and ensuring that the inconsistencies obtained for $\mathcal{D}$ are excluded from the refined model.

To split $\mathcal{D}$, we inspect the inconsistent policy $\overline{\sigma}_{\max}$. From Def. 4.3 it follows that there must exist two states $\mathbf{s}, \mathbf{s}'$ and two (not necessarily different) agents i, j, bound to the same policy variable, for which $\mathbf{s}_i = \mathbf{s}'_j$ and such that action selections $a \coloneqq \overline{\sigma}_{\max}(\mathbf{s})(i)$ and $b \coloneqq \overline{\sigma}_{\max}(\mathbf{s}')(j)$, $a \neq b$, violate local observability or policy binding. Let $\mathcal{H}$ be the set of all agents bound to the same policy variable as i and j. Splitting generates 3 submodels, where, for an arbitrary state $\mathbf{s}$ of $\mathcal{D}$ and agent $k \in \mathcal{H}$: $\mathcal{D}_1$ with $Act(\mathbf{s})(k) = a$; $\mathcal{D}_2$ with $Act(\mathbf{s})(k) = b$; and $\mathcal{D}_3$ with $Act(\mathbf{s})(k) = Act(\mathbf{s}) \setminus \{a, b\}$.

Upon splitting and obtaining a refined abstraction $\mathcal{D}_i$, its analysis proceeds as that of $\mathcal{D}$. For optimality objectives, this yields a recursive anytime algorithm tracking the currently best result.

Correctness of the algorithm follows from the construction of $\mathcal{D}$ and, in particular, the synchronised product with DRA $\mathcal{A}_\varphi$.

Theorem 4.4. *Abstraction refinement terminates and either returns a consistent tuple of memoryless policies $\sigma_1 \ldots \sigma_n$ witnessing the specification or proves that no such tuple exists.*

[2] λ can be the value of the current optimum if we seek $\mathbb{P}_{\max}(\varphi)$.

Complexity. $|\mathcal{D}|$ is doubly exponential in the length of φ and singly exponential in m. Moreover, the number of (memoryless) policies is exponential in both n and the number of states in M and thus naive enumeration of policies is intractable. Indeed, Dec-MDP problems are NEXPTIME-hard [15], even for a finite horizon.

Beyond memoryless policies. The presented algorithm analysing the synchronised product $\mathcal{D}$ can only find tuples of memoryless policies witnessing the specification. As we demonstrated in the motivating example, using memory might allow agents to satisfy specifications where simple behaviour described by memoryless policies is not enough, or to achieve better values when searching for the optimal behaviour. To enable the search for non-memoryless policies, we can unfold memory into the MDP M itself, allowing each agent to perform original actions while simultaneously updating their memory, if necessary.

Multiple probability constraints. An extension to handle multiple probability constraints derives straightforwardly from [4]. We analyse each probability constraint in the abstraction $\mathcal{D}$ as above, and, by computing also a minimizing policy $\overline{\sigma}_{\min}$, we mark it as 'SAT' (all policies satisfy the constraint, $V(\overline{\sigma}_{\min}) > \lambda$), 'UNSAT' (no policy, $V(\overline{\sigma}_{\max}) < \lambda$), or 'ambiguous' (some policy may, $V(\overline{\sigma}_{\min}) < \lambda < V(\overline{\sigma}_{\max})$). These results are then combined according to the boolean operators. If the combination generates a 'SAT' result, we return any policy. If 'UNSAT', we discard $\mathcal{D}$. Otherwise, if it is ambiguous and optimistically generated policies cannot be merged or are unsat, we split $\mathcal{D}$ w.r.t. an ambiguous constraint, and recursively analyse submodels.

5 FROM PHyperLTL TO DEC-MDP PLANNING

We introduce $\mathsf{PHyperLTL}_{\mathsf{DEC}}$, a fragment of PHyperLTL that can be reduced to Dec-MDP planning. It consists of all formulae in which i) each policy is associated with exactly one agent, ii) there is a single probability operator, and iii) all $I_1 \dots I_m$ are singleton sets. $\mathsf{PHyperLTL}_{\mathsf{DEC}}$ formulae have the following syntax:

$$\exists(\hat{\sigma}_1 \dots \hat{\sigma}_m).\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1) \dots \forall \hat{s}_m \in \{s_m\}(\hat{\sigma}_m).\mathbb{P}(\varphi) \bowtie \lambda$$

Given a $\mathsf{PHyperLTL}_{\mathsf{DEC}}$ formula Φ over an MDP M, it is easy to see that the synchronised product $\mathcal{D}$ described in Def. 4.1 is a Dec-MDP with $m+1$ agents—one for each replica of M in the self-product, plus one for the DRA. Given $\mathbf{s} = (s_1, \dots, s_m)$, the tuple of initial states associated to each policy variable in Φ, $\mathbb{P}(\varphi)$ is equivalent to the probability of reaching the acceptance set of the DRA from s in the Dec-MDP. Despite the apparent simplicity of this connection, we are, to the best of our knowledge, the first to relate a class of probabilistic hyperproperties to Dec-MDP planning.

This fact has two main consequences. First, we can exploit existing planning algorithms for Dec-MDPs to model check the fragment. In Sec. 6, we compare our abstraction refinement algorithm from Sec. 4 to a state-of-the-art Dec-MDP planning tool.

Second, we derive a new undecidability result for a class of Dec-MDPs. Since a POMDP is a special case of Dec-MDP [15], the planning problem for general Dec-MDPs is known to be undecidable. The construction of $\mathcal{D}$ turns the verification of a $\mathsf{PHyperLTL}_{\mathsf{DEC}}$ formula on M into a planning problem for a *locally fully observable* Dec-MDP. So from Theorem 3.1 follows that planning is undecidable for Dec-MDPs even if they are locally fully observable:

Corollary 5.1. *The planning problem for locally fully observable Dec-MDPs under the infinite-horizon total reward criterion is undecidable.*

6 EXPERIMENTAL EVALUATION

We implemented the proposed abstraction refinement approach in the tool PHSynt. The tool is built on top of PAYNT [6], a tool for verification approaches for families of MCs, using STORM [35] to solve MDP model-checking queries and SPOT [31] to generate Rabin automata for LTL formulae. The implementation and all the considered benchmarks are publicly available[3]. The evaluation focuses on the following questions:

- Can PHyperLTL capture relevant planning specifications?
- Can PHSynt adequately solve such problems?
- Can PHSynt find even better solutions by searching for non-memoryless policies?
- How does the performance of PHSynt fare against state-of-the-art Dec-PMDP solver Inf-JESP [50] for $\mathsf{PHyperLTL}_{\mathsf{DEC}}$?

We could not compare with the model-checking tool HyperProb [27] because it does not support the self-composition.

6.1 Experimental setting

Benchmarks. We consider decentralized planning problems over a grid-world environment [34] with various obstacles. Agents move in the four cardinal directions (via four actions), and each action has a small probability of transitioning into an unintended neighbouring cell. We consider four variants of different dimensions and varying complexity of the specification. The particular specifications are described case by case later. In all models, we equip each action with a non-zero probability that the agent transitions to a *trap state*. This allows us to explicitly model *discounting* and thus conduct a fair comparison with Inf-JESP that currently does not support undiscounted specifications.

Main tables. For every model, we report experimental results in Tables 1 and 2. Here $|\mathcal{M}|$ denotes the size of the underlying MDP, i.e. the state space of one agent. We report *bounds* on the values that can be achieved for the respective planning problem. A baseline is the value of a randomized policy (in each state, all agents uniformly pick one of the available actions), and an upper bound is given by solving the product MDP $\mathcal{D}$, i.e. assuming a centralized planning problem where each agent has perfect information about other agents. The table further shows the size of the product $|\mathcal{D}|$ for memoryless policies ($mem = 0$) and for policies using 1-bit memory ($mem = 1$). For PHSynt (and Inf-JESP in Table 1), we report the runtimes (in seconds) and the best values achieved. † indicates that PHSynt was not able to explore the whole search space: we report the best value found within a time limit of 1 hour, and the time when the best value was found. When running the experiment with $mem = 1$, a value obtained for $mem = 0$ is given to the synthesiser as a reference point. The goal here is to see whether adding memory can help the synthesiser find a better value: † without the number in parentheses indicates that no better policy has been found.

[3] https://github.com/francescopont/hyper-synthesis

Using Inf-JESP. Inf-JESP leverages point-based belief exploration methods, in particular, it repeatedly calls SARSOP [39] on sub-problems given as (single agent) POMDPs. Inf-JESP builds on randomised initialization and individual runs of the tool can vary massively. Inspired by the experimental evaluation of Inf-JESP presented in [50], we use 10 restarts and run the tool 30 times with an overall time limit of 2 hours. It means that we get at most 300 individual runs per experiment. We report the overall runtime of the experiment[4] and the best value achieved.

6.2 Results

We present 6 planning problems, demonstrate their specification via probabilistic hyperproperties, and discuss the solutions found by our approach. The first three problems are within the fragment $\mathsf{PHyperLTL}_{\mathsf{DEC}}$, so we will also evaluate these using Inf-JESP. All the experiments are run on a device equipped with AMD Ryzen 5 6600U @2.9GHz and 64GB RAM.

Meeting. We consider a classical decentralized planning problem [33] where two agents, each starting from different initial locations s_1 and s_2, seek to meet at a given target location T. Note that the two agents do not have the option to sit still upon reaching the target, and therefore the probability of both agents residing at the same location is not the product of individual reachability probabilities. In the following PHyperLTL formula, we require to maximise the probability of meeting, thus to meet as soon as possible (recall the transitions to the trap state)

$$\exists\,(\hat{\sigma}_1\,\hat{\sigma}_2)\,.\,\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1)\,\forall \hat{s}_2 \in \{s_2\}(\hat{\sigma}_2)\,.\,\mathbb{P}_{\max}(\Diamond\ (T_{\hat{s}_1} \wedge T_{\hat{s}_2}))$$

We observe that, for smaller grids, memoryless controllers obtained with PHSynt often suffice to get a close-to-maximal probability: often simply heading to the meeting location is good enough. Inf-JESP achieves similar results, even marginally outperforming PHSynt at meet-12x7. For larger grids, however, the value achieved by the memoryless policy gets further from the upper bound: memory is needed to navigate different paths that facilitate the meeting. In those cases, PHSynt manages to improve the achieved probability when policies are equipped with 1 bit of memory. On the other hand, Inf-JESP struggles to find solutions that outperform memoryless policies. Due to the exponential blow-up of the search space (and the increased size of the model), PHSynt does not fully explore the design space of policies within the timeout, although the improved solution is usually obtained very quickly, e.g. in 47 seconds for meet-19x11.

We also consider a variant meet^R where the goal is for the two agents to meet at a designated area while minimising costs associated with the movement of one of the agents. This variant was designed to include in our benchmark suite also problems with non-sparse rewards. We observe that even memoryless policies are sufficient to achieve close-to-optimal behaviour. PHSynt again often produces close-to-optimal solutions. This does not mean that these problems are trivial, cf. meet^R-19x11, where Inf-JESP cannot find adequate policies even after 2 hours.

Race. We consider the specification from the motivating example (see the introduction) that can be seen as a probabilistic variant of the *shortest path* hyperproperty [48]. The race-3 variant reported in Table 1 considers three agents with a specific desired order of arrival. The obtained policies indicate again that often memoryless policies are good enough, depending on the initial locations of the agents that may already favour the intended winner of the race. For the bigger variants of race-3, PHSynt fails to build the model for $mem = 1$ due to its size. On the other hand, the publicly available version of Inf-JESP cannot handle Dec-(PO)MDPs with more than two agents, thus no values are reported in the table. For the variants available to both tools, one can see that the performance of Inf-JESP significantly lags behind PHSynt both in the quality of the produced policies as well as the overall runtime.

Current state opacity. Opacity with respect to the current state is a well-studied security/privacy planning requirement both in the deterministic [48, 49, 51] and in the probabilistic setting [4, 28]. It ensures that an intruder with limited observation capability is not able to fully guess an agent's current state. We consider the variant of [28]. In our setting, two agents start from two close initial states and need to reach a target location via two different plans such that, in their synchronised runs, the agents always find themselves in the same region of the grid. This way, the intruder cannot distinguish between the two. In PHyperLTL:

$$\exists\,(\hat{\sigma}_1\,\hat{\sigma}_2)\,.\,\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1)\,\forall \hat{s}_2 \in \{s_2\}(\hat{\sigma}_2)\,.$$
$$\mathbb{P}_{\max}\big(\neg\Box\,(act_{\hat{s}_1} = act_{\hat{s}_2}) \wedge \Box\,(reg_{\hat{s}_1} = reg_{\hat{s}_2}) \wedge \Diamond\,(T_{\hat{s}_1}) \wedge \Diamond\,(T_{\hat{s}_2})\big)$$

where *act* represents the action chosen by an agent in a state, and *reg* is the current region of the agent (intruder's observation). We observe that, for smaller variants, PHSynt can quickly synthesise the best memoryless policy vector and can improve upon the obtained value when considering non-memoryless policies. This is the only model in our benchmark set that seems to be favoured by Inf-JESP that can compute better solutions, although it fails to build a meaningful tuple of policies for the larger variant. However, a high upper bound indicates that the policies computed by both tools may be far from the optimal one.

Up until now, the considered specifications belonged to the $\mathsf{PHyperLTL}_{\mathsf{DEC}}$ fragment. We now consider problems that cannot be translated into a Dec-MDP formulation.

Initial state opacity (ISO). A variation of the opacity requirement is initial state opacity (ISO) [46, 48]. Consider a variant where each action has some (different) probability of failing and going to a sink state *S*. We seek one policy that is opaque with respect to the initial state, i.e. if the two agents execute the policy from two different initial states, they either simultaneously reach the target location or simultaneously end up in the sink state. In PHyperLTL:

$$\exists\,(\hat{\sigma}_1)\,.\,\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1)\,\forall \hat{s}_2 \in \{s_2\}(\hat{\sigma}_1)\,.$$
$$\mathbb{P}_{\max}\big(\,(\neg T_{\hat{s}_1} \wedge \neg T_{\hat{s}_2})\,\mathsf{U}\,(T_{\hat{s}_1} \wedge T_{\hat{s}_2})\ \vee$$
$$(\neg S_{\hat{s}_1} \wedge \neg S_{\hat{s}_2} \wedge \neg T_{\hat{s}_1} \wedge \neg T_{\hat{s}_2})\,\mathsf{U}\,(S_{\hat{s}_1} \wedge S_{\hat{s}_2})\big)$$

PHSynt can find the optimal memoryless policy for all variants but one (ISO-15x9) and, for the smallest variant (ISO-4x4), can even find the optimal policy with 1-bit memory.

Action failure robustness. Robustness of the synthesized plan is a major concern in uncertain environments. Here we consider action-failure robustness [48]. We seek one robust policy for the initial state that, even in the case of a slip (*sl*)—the agent fails to execute

[4] The runtime can exceed 2 hours since we do not terminate runs that already started.

Table 1: Results for decentralised planning problems that can be encoded as Dec-MDPs. All models maximise, meet^R minimises, hence its bound is actually a lower bound. Values in bold indicate the best results. Individual columns are described in Sec. 6.1.

model	$\|\mathcal{M}\|$	bounds		product		PHSynt		Inf-JESP		model	$\|\mathcal{M}\|$	bounds		product		PHSynt		Inf-JESP	
		rand.	upp.	mem	$\|\mathcal{D}\|$	time	value	time	value			rand.	upp.	mem	$\|\mathcal{D}\|$	time	value	time	value
meet-4x4	22	0.1	0.66	0	122	<1	0.63	15	0.63	meet-12x7	78	0.04	0.66	0	2k	347	0.63	1839	**0.64**
				1	485	†	0.63							1	6k	†	0.63		
meet-15x9	160	0	0.47	0	6k	† (41)	0.32	351	0	meet-19x11	216	0.01	0.57	0	12k	† (9)	0.29	7298	0.17
				1	26k	†(370)	**0.33**							1	47k	† (47)	**0.56**		
meet^R-4x4	22	322	106	0	122	<1	117	1262	118	meet^R-12x7	78	347	104	0	2k	9	**109**	4572	128
				1	485	† (355)	**114**							1	6k	†	109		
meet^R-15x9	160	43	21	0	6k	† (498)	**22**	509	25	meet^R-19x11	216	655	240	0	12k	421	**245**	9730	606
				1	26k	†	22							1	47k	†	245		
race-2-4x4	18	0.2	0.8	0	76	<1	0.7	9	0.72	race-2-12x7	64	0.07	0.7	0	236	1	**0.65**	1457	0.62
				1	295	2777	**0.76**							1	935	†	0.65		
race-2-15x9	160	0.02	0.58	0	6k	26	**0.58**	195	0.51	race-2-19x11	168	0.01	0.61	0	2k	19	**0.57**	2968	0.53
				1	26k	†	0.58							1	7k	†	0.57		
race-3-4x4	18	0.17	0.76	0	228	0	0.65	-	-	race-3-12x7	64	0.02	0.61	0	2k	†	0.42	-	-
				1	2k	† (346)	0.71							1	†	†	†		
race-3-15x9	160	0	0.39	0	513k	† (528)	0.27	-	-	opac-4x4	42	0	0.37	0	529	5	0.09	80	**0.15**
				1	†	†	†							1	2k	† (1000)	0.14		
opac-12x7	130	0	0.45	0	6k	25	0.11	3730	**0.19**	opac-15x9	540	0	0.29	0	94k	† (3)	**0.01**	2234	0
				1	23k	† (349)	0.15							1	376k	†	0.01		

Table 2: The results for the decentralised planning problems that can not be encoded as Dec-MDPs. All models maximise.

model	$\|\mathcal{M}\|$	bounds		product		PHSynt	
		rand.	upp.	mem	$\|\mathcal{D}\|$	time	value
ISO-4x4	22	0.01	0.47	0	1k	<1	0.08
				1	4k	1677	0.15
ISO-12x7	78	0.01	0.46	0	13k	6	0.2
				1	51k	†	0.2
ISO-15x9	160	0.02	0.29	0	74k	†(2244)	0.04
				1	297k	†	0.04
ISO-19x11	216	0.01	0.37	0	98k	1093	0.09
				1	390k	†	0.09
robust-4x4	35	0.23	0.7	0	281	<1	0.64
				1	1k	†	0.64
robust-12x7	128	0.12	0.63	0	375	<1	0.59
				1	1k	†	0.59
robust-15x9	640	0.04	0.59	0	23k	† (13)	0.53
				1	91k	†	0.53
robust-19x11	203	0.14	0.73	0	9k	15	0.64
				1	36k	†	0.64
noninter-4x4	22	0.25	0.71	0	51k	80	0.69

an action and goes instead to a neighbouring cell—still manages to reach the target location as if the accident had not happened. Here the executions start from the same initial state. In PHyperLTL:

$$\exists\,(\hat{\sigma}_1)\,.\,\forall \hat{s}_1 \in \{s_1\}(\hat{\sigma}_1)\,\forall \hat{s}_2 \in \{s_1\}(\hat{\sigma}_1)\,.\,\mathbb{P}_{\max}\Big(\big(\Diamond\, T_{\hat{s}_1} \wedge \Diamond\, T_{\hat{s}_2}\big) \wedge \big((\Diamond\, sl_{\hat{s}_1} \oplus \Diamond\, sl_{\hat{s}_2}) \implies (\neg T_{\hat{s}_1} \wedge \neg T_{\hat{s}_2})\,\mathsf{U}\,(T_{\hat{s}_1} \wedge T_{\hat{s}_2})\big)\Big)$$

where $\oplus$ is the *xor* operator. Regardless of the size, PHSynt quickly synthesises the optimal memoryless policy with a value close to the upper bound. Adding more memory seems to have no effect.

Plan noninterference. With PHyperLTL it is possible to express verification questions in path planning, such as *noninterference* [26, 28]. Again, two agents are moving in the grid towards a goal location from two initial locations far from each other. The second one, A2, is an intruder that wants to interfere with A1 by reaching the goal first. We want to verify whether changes in the policy of the intruder can affect A1. We seek one policy for A1 and two for A2 such that when the policy for A1 is executed together with one of the policies for A2, only in one execution the goal is accomplished. In PHyperLTL:

$$\exists\,(\hat{\sigma}_{A1}\ \hat{\sigma}_{A2a}\ \hat{\sigma}_{A2b})\ .\,\forall \hat{s}_{A1a} \in \{s_1\}(\hat{\sigma}_{A1})\ \forall \hat{s}_{A1b} \in \{s_1\}(\hat{\sigma}_{A1})\,\forall \hat{s}_{A2a} \in \{s_2\}(\hat{\sigma}_{A2a})\,\forall \hat{s}_{A2b} \in \{s_2\}(\hat{\sigma}_{A2b})\,.\ \mathbb{P}_{\max}\big((\neg goal_{\hat{s}_{A1a}}\,\mathsf{U}\,goal_{\hat{s}_{A2a}}) \oplus (\neg goal_{\hat{s}_{A1b}}\,\mathsf{U}\,goal_{\hat{s}_{A2b}})\big)$$

We consider only the 4x4 variant of this problem as it requires a 4-self-composition. PHSynt quickly computes memoryless policies that show that the intruder can indeed interfere with A1 with a very high probability.

7 CONCLUSION

We introduce PHyperLTL, a novel specification formalism for decentralized planning under uncertainty. It allows for specifying temporal constraints between policies and different executions of these policies by the agents. We use a synchronised product construction that allows us to (i) design an abstraction-refinement loop seeking the optimal joint policy and (ii) establish a close connection between a fragment of the logic and planning for Dec-MDPs. On a broad set of benchmarks, we demonstrate that the proposed abstraction-refinement approach can solve challenging problems in non-trivial environments and, in almost all cases, outperforms the state-of-the-art tool Inf-JESP on Dec-MDP problems.

A major limitation of the proposed approach is the exponential blow-up of the state space yielding specifications that are sometimes intractable even for small environments. In future work, we will investigate whether bottom-up approaches mitigating the state explosion [43, 47] can be applied in our setting.

ACKNOWLEDGMENTS

We are thankful to Tim Quatmann for some pointers to the Storm codebase. This work has been executed under the project VASSAL: "Verification and Analysis for Safety and Security of Applications in Life" funded by the European Union under Horizon Europe WIDERA Coordination and Support Action/Grant Agreement No. 101160022. This work has been partially funded by the EU Commission in the Horizon Europe research and innovation programme under grant agreement No. 101034440 (Marie Sklodowska-Curie Doctoral Network LogiCS@TU Wien) and 101107303 (MSCA Postdoctoral Fellowship CORPORA). Additionally, this work has been funded by the Czech Science Foundation grant GA23-06963S (VESCAA), the IGA VUT project FIT-S-23-8151, and the Vienna Science and Technology Fund project ICT22-023 (TAIGER).

REFERENCES

[1] Erika Ábrahám, Ezio Bartocci, Borzoo Bonakdarpour, and Oyendrila Dobe. 2019. Probabilistic Hyperproperties with Nondeterminism. In *ATVA'19*. 518–534. https://doi.org/10.1007/978-3-030-59152-6_29

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time temporal logic. *J. ACM* 49, 5 (2002), 672–713. https://doi.org/10.1145/585265.585270

[3] Benjamin Aminof, Marta Kwiatkowska, Bastien Maubert, Aniello Murano, and Sasha Rubin. 2019. Probabilistic Strategy Logic. In *IJCAI'19*. ijcai.org, 32–38. https://doi.org/10.24963/IJCAI.2019/5

[4] Roman Andriushchenko, Ezio Bartocci, Milan Ceska, Francesco Pontiggia, and Sarah Sallinger. 2023. Deductive Controller Synthesis for Probabilistic Hyperproperties. In *QEST'23*. 288–306. https://doi.org/10.1007/978-3-031-43835-6_20

[5] Roman Andriushchenko, Milan Ceska, Sebastian Junges, and Joost-Pieter Katoen. 2022. Inductive synthesis of finite-state controllers for POMDPs. In *UAI'22*. 85–95.

[6] Roman Andriushchenko, Milan Ceska, Sebastian Junges, Joost-Pieter Katoen, and Simon Stupinský. 2021. PAYNT: A Tool for Inductive Synthesis of Probabilistic Programs. In *CAV'21*. 856–869. https://doi.org/10.1007/978-3-030-81685-8_40

[7] Christel Baier, Nathalie Bertrand, and Marcus Größer. 2008. On Decision Problems for Probabilistic Büchi Automata. In *FOSSACS'08*. 287–301. https://doi.org/10.1007/978-3-540-78499-9_21

[8] Christel Baier, Luca de Alfaro, Vojtěch Forejt, and Marta Kwiatkowska. 2018. Model Checking Probabilistic Systems. In *Handbook of Model Checking*. 963–999. https://doi.org/10.1007/978-3-319-10575-8_28

[9] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*.

[10] Raphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. 2004. Solving Transition Independent Decentralized Markov Decision Processes. *J. Artif. Intell. Res.* 22 (2004), 423–455. https://doi.org/10.1613/JAIR.1497

[11] Francesco Belardinelli, Wojciech Jamroga, Munyque Mittelmann, and Aniello Murano. 2023. Strategic Abilities of Forgetful Agents in Stochastic Environments. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner (Eds.). 726–731. https://doi.org/10.24963/KR.2023/71

[12] Francesco Belardinelli, Wojtek Jamroga, Munyque Mittelmann, and Aniello Murano. 2024. Verification of Stochastic Multi-Agent Systems with Forgetful Strategies. In *AAMAS '24*, Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum (Eds.). International Foundation for Autonomous Agents and Multiagent Systems / ACM, 160–169. https://doi.org/10.5555/3635637.3662863

[13] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2017. Verification of Broadcasting Multi-Agent Systems against an Epistemic Strategy Logic. In *IJCAI '17*, Carles Sierra (Ed.). ijcai.org, 91–97. https://doi.org/10.24963/IJCAI.2017/14

[14] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2020. Verification of multi-agent systems with public actions against strategy logic. *Artif. Intell.* 285 (2020), 103302. https://doi.org/10.1016/J.ARTINT.2020.103302

[15] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.* 27, 4 (2002), 819–840. https://doi.org/10.1287/MOOR.27.4.819.297

[16] Raphaël Berthon, Bastien Maubert, Aniello Murano, Sasha Rubin, and Moshe Y. Vardi. 2021. Strategy logic with imperfect information. *ACM Trans. Comput. Log.* 22, 1 (2021), 1–51. https://doi.org/10.1145/3427955

[17] Raven Beutner and Bernd Finkbeiner. 2024. Hyper Strategy Logic. In *AAMAS'24*. 189–197.

[18] Raven Beutner and Bernd Finkbeiner. 2024. Non-deterministic Planning for Hyperproperty Verification. In *ICAPS'24*. 25–30. https://doi.org/10.1609/ICAPS.V34I1.31457

[19] Davide Catta, Jean Leneutre, Vadim Malvone, and Aniello Murano. 2024. Obstruction Alternating-time Temporal Logic: A Strategic Logic to Reason about Dynamic Models. In *AAMAS'24*. 271–280. https://doi.org/10.5555/3635637.3662875

[20] Milan Ceska, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. 2019. Shepherding Hordes of Markov Chains. In *TACAS'19*. 172–190. https://doi.org/10.1007/978-3-030-17465-1_10

[21] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. 2010. Strategy logic. *Inf. Comput.* 208, 6 (2010), 677–693. https://doi.org/10.1016/J.IC.2009.07.004

[22] Taolue Chen and Jian Lu. 2007. Probabilistic Alternating-time Temporal Logic and Model Checking Algorithm. In *FSKD '07*, J. Lei (Ed.). IEEE Computer Society, 35–39. https://doi.org/10.1109/FSKD.2007.458

[23] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. 2014. Temporal Logics for Hyperproperties. In *POST'14*. 265–284. https://doi.org/10.1007/978-3-642-54792-8_15

[24] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *J. Comput. Secur.* 18, 6 (2010), 1157–1210. https://doi.org/10.3233/JCS-2009-0393

[25] Anne Condon and Richard J. Lipton. 1989. On the complexity of space bounded interactive proofs. In *FOCS'89*, Vol. 89. 462–467. https://doi.org/10.1109/SFCS.1989.63519

[26] Rayna Dimitrova, Bernd Finkbeiner, and Hazem Torfah. 2020. Probabilistic Hyperproperties of Markov Decision Processes. In *ATVA'20*. 484–500. https://doi.org/10.1007/978-3-030-59152-6_27

[27] Oyendrila Dobe, Erika Ábrahám, Ezio Bartocci, and Borzoo Bonakdarpour. 2021. HyperProb: A Model Checker for Probabilistic Hyperproperties. In *FM'21*. 657–666. https://doi.org/10.1007/978-3-030-90870-6_35

[28] Oyendrila Dobe, Stefan Schupp, Ezio Bartocci, Borzoo Bonakdarpour, Axel Legay, Miroslav Pajic, and Yu Wang. 2023. Lightweight Verification of Hyperproperties. In *ATVA'23*. 3–25. https://doi.org/10.1007/978-3-031-45332-8_1

[29] Oyendrila Dobe, Lukas Wilke, Erika Ábrahám, Ezio Bartocci, and Borzoo Bonakdarpour. 2022. Probabilistic Hyperproperties with Rewards. In *NFM 2022 (LNCS, Vol. 13260)*. 656–673. https://doi.org/10.1007/978-3-319-11936-6_11

[30] Oyendrila Dobe, Erika Ábrahám, Ezio Bartocci, and Borzoo Bonakdarpour. 2022. Model checking hyperproperties for Markov decision processes. *Information and Computation* 289 (2022), 104978. https://doi.org/10.1016/j.ic.2022.104978

[31] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. 2022. From Spot 2.0 to Spot 2.10: What's New?. In *CAV'22*. 174–187. https://doi.org/10.1007/978-3-031-13188-2_9

[32] Rusins Freivalds. 1981. Probabilistic Two-Way Machines. In *MFCS'81*. 33–45. https://doi.org/10.1007/3-540-10856-4_72

[33] Claudia V. Goldman and Shlomo Zilberstein. 2004. Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. *J. Artif. Intell. Res.* 22 (2004), 143–174. https://doi.org/10.1613/JAIR.1427

[34] Milos Hauskrecht. 1997. Incremental methods for computing bounds in partially observable Markov decision processes. In *AAAI/IAAI'97*. 734–739.

[35] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.* 24, 4 (2022), 589–610. https://doi.org/10.1007/s10009-021-00633-z

[36] Xiaowei Huang and Cheng Luo. 2013. A logic of probabilistic knowledge and strategy. In *AAMAS '13*, Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker (Eds.). IFAAMAS, 845–852. http://dl.acm.org/citation.cfm?id=2485055

[37] Xiaowei Huang, Kaile Su, and Chenyi Zhang. 2012. Probabilistic Alternating-Time Temporal Logic of Incomplete Information and Synchronous Perfect Recall. In *AAAI '12*, Jörg Hoffmann and Bart Selman (Eds.). AAAI Press, 765–771. https://doi.org/10.1609/AAAI.V26I1.8214

[38] Wietze Koops, Nils Jansen, Sebastian Junges, and Thiago D. Simão. 2023. Recursive Small-Step Multi-Agent A* for Dec-POMDPs. In *IJCAI'23*. 5402–5410. https://doi.org/10.24963/IJCAI.2023/600

[39] Hanna Kurniawati, David Hsu, and Wee Sun Lee. 2008. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science and Systems IV*. https://doi.org/10.15607/RSS.2008.IV.009

[40] John McLean. 1994. A general theory of composition for trace sets closed under selective interleaving functions. In *S&P'94*. 79–93. https://doi.org/10.1109/RISP.1994.296590

[41] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comput. Log.* 15, 4 (2014), 34:1–34:47. https://doi.org/10.1145/2631917

[42] Masakazu Nasu and Namio Honda. 1969. Mappings Induced by PGSM-Mappings and Some Recursively Unsolvable Problems of Finite Probabilistic Automata. *Inf. Control.* 15, 3 (1969), 250–273. https://doi.org/10.1016/S0019-9958(69)90449-5

[43] Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. 2021. Reward Machines for Cooperative Multi-Agent Reinforcement Learning. In *AAMAS'21*. 934–942. https://doi.org/10.5555/3463952.3464063

[44] Amir Pnueli. 1977. The Temporal Logic of Programs. In *FOCS'77*. 46–57. https://doi.org/10.1109/SFCS.1977.32

[45] Francesco Pontiggia, Filip Macák, Roman Andriushchenko, Michele Chiari, and Milan Češka. 2025. Decentralized Planning Using Probabilistic Hyperproperties. https://arxiv.org/abs/2502.13621

[46] Anooshiravan Saboori and Christoforos N. Hadjicostis. 2013. Verification of initial-state opacity in security applications of discrete event systems. *Inf. Sci.* 246 (2013), 115–132. https://doi.org/10.1016/J.INS.2013.05.033

[47] Georg F. Schuppe and Jana Tumova. 2021. Decentralized Multi-Agent Strategy Synthesis under LTLf Specifications via Exchange of Least-Limiting Advisers. In *MRS'21*. https://doi.org/10.1109/MRS50823.2021.9620561

[48] Yu Wang, Siddhartha Nalluri, and Miroslav Pajic. 2020. Hyperproperties for Robotics: Planning via HyperLTL. In *ICRA'20*. 8462–8468. https://doi.org/10.1109/ICRA40945.2020.9196874

[49] Xiang Yin and Stéphane Lafortune. 2015. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In *ACC'15*. 377–383. https://doi.org/10.1109/ACC.2015.7170765

[50] Yang You, Vincent Thomas, Francis Colas, and Olivier Buffet. 2021. Solving infinite-horizon Dec-POMDPs using Finite State Controllers within JESP. In *ICTAI'21*. 427–434. https://doi.org/10.1109/ICTAI52525.2021.00069

[51] Kuize Zhang, Xiang Yin, and Majid Zamani. 2019. Opacity of Nondeterministic Transition Systems: A (Bi)Simulation Relation Approach. *IEEE Trans. Autom. Control.* 64, 12 (2019), 5116–5123. https://doi.org/10.1109/TAC.2019.2908726

[52] Chenyang Zhu, Wen Si, Jinyu Zhu, and Zhihao Jiang. 2024. Decomposing Temporal Equilibrium Strategy for Coordinated Distributed Multi-Agent Reinforcement Learning. In *AAAI'24*, Vol. 38. 17618–17627. https://doi.org/10.1609/AAAI.V38I16.29713