Formal Methods in Computer-Aided Design 2025

On Hyperproperty Verification, Quantifier Alternations, and Games under Partial Information

Raven Beutner
CISPA Helmholtz Center for Information Security Germany

Bernd Finkbeiner
CISPA Helmholtz Center for Information Security Germany

***Abstract*—Hyperproperties generalize traditional trace properties by relating multiple execution traces rather than reasoning about individual runs in isolation. They provide a unified way to express important requirements such as information flow and robustness properties. Temporal logics like HyperLTL capture these properties by explicitly quantifying over executions of a system. However, many practically relevant hyperproperties involve *quantifier alternations*, a feature that poses substantial challenges for automated verification. Complete verification methods require a system complementation for each quantifier alternation, making it infeasible in practice. A cheaper (but incomplete) method interprets the verification of a HyperLTL formula as a two-player game between universal and existential quantifiers. The game-based approach is significantly cheaper, facilitates interactive proofs, and allows for easy-to-check certificates of satisfaction. It is, however, limited to $\forall^*\exists^*$ properties, leaving important properties out of reach. In this paper, we show that we can use games to verify hyperproperties with arbitrary quantifier alternations by utilizing multiplayer games under *partial information*. While games under partial information are, in general, undecidable, we show that our game is played under hierarchical information and thus falls in a decidable class of games. We discuss the completeness of the game and study prophecy variables in the setting of partial information.**

I. Introduction

In 2008, Clarkson and Schneider [1] coined the term *hyperproperties* for the rich class of system requirements that relate multiple executions. In contrast to *trace properties* – i.e., properties over individual executions, expressed, e.g., in linear-time temporal logics (LTL) [2] – hyperproperties can express important properties related to information flow, knowledge, and robustness. As an example, consider a system with secret input h, public input l, and public output o, and assume we want to express that the public behavior does not leak any information about the secret input. We cannot express such an information-flow requirement as a trace property in, e.g., LTL; we need to compare *multiple* executions to see if (and how) the secret input impacts the output. Instead, we can express it as a hyperproperty in HyperLTL [3], an extension of LTL with explicit quantification over execution traces. For example,

$$\forall \pi_1.\forall \pi_2. \square(l_{\pi_1} \leftrightarrow l_{\pi_2}) \to \square(o_{\pi_1} \leftrightarrow o_{\pi_2}). \tag{OD}$$

requires that any pair of executions π_1, π_2 with identical public input also has the same output, i.e., the output is fully determined by the public input and thus cannot possibly leak the secret input (cf. *observational determinism* [4]).

While originating in the study of information flow, hyperproperties have since been established as a much more general framework that captures properties from many different areas, including, e.g., knowledge properties in multi-agent systems (MAS) [5], [6]. As an example, consider some MAS with agents $1, \ldots, n$, and some LTL property ψ, and assume that we want to verify that there exists at least one execution of the MAS such that agent i knows that ψ holds (e.g., some adversary knowing some secret). Formally, *knowing* that ψ holds on some execution trace π_1 means that ψ must hold on all traces π_2 that are indistinguishable from π_1 for agent i (cf. [7]), which *is* a hyperproperty. We can easily express this property in HyperLTL, as follows

$$\exists \pi_1.\forall \pi_2. \, \pi_1 \equiv_i \pi_2 \to \psi[\pi_2], \tag{K$_1$}$$

where we write $\psi[\pi_2]$ to indicate that ψ holds on trace π_2, and $\pi_1 \equiv_i \pi_2$ denotes that executions π_1 and π_2 appear indistinguishable under agent i's observations. Using the flexibility of quantification, we can also express nested knowledge properties. For example, we can express that, on some execution, agent i knows that agent j does *not* know whether ψ holds:

$$\exists \pi_1.\forall \pi_2.\exists \pi_3.\exists \pi_4. \, \pi_1 \equiv_i \pi_2 \to \big(\pi_2 \equiv_j \pi_3 \land \pi_2 \equiv_j \pi_4 \land \psi[\pi_3] \land \neg\psi[\pi_4]\big), \tag{K$_2$}$$

i.e., for every trace π_2 that agent i cannot distinguish from π_1, there exist two traces π_3, π_4 that agent j cannot distinguish from π_2, one of which satisfies ψ and one violates ψ.

Verification of HyperLTL: In this paper, we study the model-checking problem for HyperLTL (or, more generally, of hyperproperties that can be expressed using HyperLTL-style quantification over system paths or traces [8]–[10]). Unsurprisingly, the quantifier prefix of the HyperLTL formula directly impacts the complexity of this verification problem. For alternation-free formulas (e.g., OD), verification can be reduced to the verification of an LTL property on the self-composition of the system [11], [12], which is very efficient. Verification gets much more challenging when the formula includes quantifier alternations as used in K$_1$, K$_2$, and many other HyperLTL formulas studied in the literature [13]–[15]. Complete approaches require one system complementation for each alternation in the formula [3], [12], making it infeasible in practice.

https://doi.org/10.34727/2025/isbn.978-3-85448-084-6_8

Game-based Verification: A cheaper (but incomplete) verification method for $\forall^*\exists^*$ formulas (i.e., formulas where an arbitrary number of universal quantifiers is followed by an arbitrary number of existential quantifiers) is based on a strategy-based interpretation of existential quantification [16], [17]. The key idea is to interpret the verification of a HyperLTL formula of the form $\forall \pi_1 . \exists \pi_2 . \psi$ (where ψ is the LTL body) as a *game* between two players. A refuter controls the universally quantified trace by moving through a copy of the underlying system, thereby constructing a concrete trace for π_1. The verifier reacts to the moves by the refuter and moves through a separate copy of the system, thereby producing a concrete trace for π_2. The goal of the verifier is to ensure that π_1 and π_2, together, satisfy ψ. We can think of the verifier's strategy as providing a step-wise construction of a concrete witness trace for π_2 (akin to a Skolem function). This game-based approach is sound (i.e., if the verifier wins, the hyperproperty is satisfied by the system) and computationally cheaper than complementation-based approaches. Moreover, the game-based approach also allows for interactive proofs and witnesses of satisfaction. For example, we can use the game-based framework to let the user construct a strategy *interactively* [18], allowing verification even in situations where automated techniques do not scale. Likewise, we can use a winning strategy for the verifier as an (easy-to-check) certificate that the property is satisfied [19].

Unsoundness Beyond $\forall^*\exists^*$: The game-based verification approach of Coenen et al. [16] and its descendants have proven themselves in many situations (cf. Section VII). However, since its inception, the approach has been limited to $\forall^*\exists^*$ properties. Intuitively, as soon as we consider properties beyond $\forall^*\exists^*$, the step-wise selection of the traces leads to unsoundness, i.e., cases where a winning strategy exists even though the property is violated; we give a concrete instance in Example 1. Consequently, for properties outside the $\forall^*\exists^*$ fragment, no effective verification approximation exists (cf. Section VII), nor does there exist any approach that allows interactive proofs or satisfaction certificates.

Partial Information: In this paper, we present a novel game-based method that allows us to soundly verify *arbitrary* quantifier structures. Our key contribution is the observation that we need to reason about *partial information*. In our game-based encoding, we consider multiple players, each of whom controls a unique trace in the HyperLTL formula. We then carefully design an observation model for each player to ensure soundness. Intuitively, our observations ensure that each player controlling some trace π can only observe the state sequence from traces that are quantified *before* π. We thus obtain a multiplayer game played under partial information that, if won by a certain group of players, ensures that the formula holds on the given system (Section IV).

Hierarchical Information: In general, multiplayer games under incomplete information are undecidable. We show that our verification game falls in a well-known class of games that can be solved effectively. Namely, games where the information of the players is *hierarchical* [20]–[26].

Completeness and Prophecy Variables: Similar to the $\forall^*\exists^*$ game [16], [17], our game-based approach using partial information is incomplete: In some cases, the property holds, but no winning strategy exists. While incomplete for $\forall^*\exists^*$ properties, we show that our approach is complete for $\exists^*\forall^*$ properties (Section V). This gives rise to a sound-and-complete verification method for all properties with at most one quantifier alternation (via complementation), at the cost of introducing partial information. More generally, we study the use of prophecy variables [17], [27] in our verification game, allowing a player to peek at the future temporal behavior of other players (Section VI).

Applications: The core contribution that partial information enables game-based verification of arbitrary hyperproperties creates a plethora of new possibilities for hyperproperty verification: Our game-based view **(1)** leverages techniques for solving partial information games (cf. Section VII) for automated verification; **(2)** facilitates interactive proof of hyperproperties beyond $\forall^*\exists^*$ by letting the user construct strategies; **(3)** enables strategies as easy-to-check certificates for satisfaction; and **(4)** supports prophecies to soundly strengthen the approach, both in automated and interactive verification.

Full Version: Proofs of all results can be found in the full version [28].

II. Preliminaries

Kripke Structures: As the basic system model, we use finite-state Kripke structures. We assume that AP is a fixed set of *atomic propositions* (AP). A Kripke structure (KS) is a tuple $\mathcal{K} = (S, s_{init}, \mathbb{D}, \kappa, \ell)$, where S is a finite set of states, $s_{init} \notin S$ is a dedicated initial state (*not* part of S), $\mathbb{D}$ is a finite set of directions, $\kappa : (S \uplus \{s_{init}\}) \times \mathbb{D} \to S$ is the transition function, and $\ell : (S \uplus \{s_{init}\}) \to 2^{AP}$ labels each state with an evaluation of the APs.[1] A path in $\mathcal{K}$ is an infinite sequence $\tau \in (S \uplus \{s_{init}\})^\omega$ such that $\tau(0) = s_{init}$, and for every $i \in \mathbb{N}$, there exists some $d \in \mathbb{D}$ such that $\tau(i+1) = \kappa(\tau(i), d)$. We define $\mathit{Paths}(\mathcal{K})$ as the set of all paths in $\mathcal{K}$. Each path τ denotes an associated trace $\ell(\tau) := \ell(\tau(0))\ell(\tau(1))\ell(\tau(2)) \cdots \in (2^{AP})^\omega$ defined by applying ℓ pointwise. We define $\mathit{Traces}(\mathcal{K}) := \{\ell(\tau) \mid \tau \in \mathit{Paths}(\mathcal{K})\} \subseteq (2^{AP})^\omega$ as the set of all traces generated by $\mathcal{K}$.

Linear-Time Temporal Logic: Linear-time temporal logic (LTL) [2] formulas are defined as follows

$$\psi := a \mid \psi \land \psi \mid \neg \psi \mid \bigcirc \psi \mid \psi \mathcal{U} \psi,$$

where $a \in AP$ is an atomic proposition. The basic formula a requires that the AP a holds in the current state, $\bigcirc \psi$ requires that ψ holds in the *next* step, and $\psi_1 \mathcal{U} \psi_2$ requires that ψ_1 holds *until* ψ_2 eventually holds. We use the usual derived constants and connectives $\mathit{true}, \mathit{false}, \lor, \to, \leftrightarrow$, and

[1]We use slightly unconventional notation in two places: Firstly, we assume a dedicated initial state s_{init}, simplifying the addition of prophecies. Secondly, we use *directions* to uniquely identify successor states, simplifying our game construction. Traditionally, transition functions in Kripke structures are functions $S \to 2^S \setminus \{\emptyset\}$ that map each state to a non-empty set of successor states. We can easily transform such a transition function into a directed function $S \times \mathbb{D} \to S$ by using sufficiently many directions.

the temporal operators *eventually* $\Diamond\psi := \mathit{true}\,\mathcal{U}\,\psi$, and *globally* $\Box\psi := \neg\Diamond\neg\psi$. Given a trace $t \in (2^{AP})^\omega$, we define the semantics of LTL for each time point $i \in \mathbb{N}$ as follows:

$$\begin{aligned}
&t, i \models a && \text{iff} \quad a \in t(i)\\
&t, i \models \psi_1 \land \psi_2 && \text{iff} \quad t, i \models \psi_1 \text{ and } t, i \models \psi_2\\
&t, i \models \neg\psi && \text{iff} \quad t, i \not\models \psi\\
&t, i \models \bigcirc\psi && \text{iff} \quad t, i+1 \models \psi\\
&t, i \models \psi_1 \,\mathcal{U}\, \psi_2 && \text{iff} \quad \exists k \geq i.\, t, k \models \psi_2 \text{ and}\\
&&& \qquad\qquad \forall i \leq j < k.\, t, j \models \psi_1
\end{aligned}$$

We write $t \models_{LTL} \psi$ if t satisfies ψ, i.e., $t, 0 \models \psi$.

Deterministic Parity Automata: A deterministic parity automaton (DPA) is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, c)$ where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, and $c : Q \to \mathbb{N}$ colors each state with a natural number. For an infinite word $u \in \Sigma^\omega$, we define $run(\mathcal{A}, u) \in Q^\omega$ as the unique run of $\mathcal{A}$ on u. Formally, $run(\mathcal{A}, u)$ is the unique word in Q^ω such that $run(\mathcal{A}, u)(0) = q_0$ and for every $i \in \mathbb{N}$, $run(\mathcal{A}, u)(i+1) = \delta(run(\mathcal{A}, u)(i), u(i))$. That is, we start the run in the initial state q_0 and progress by following $\mathcal{A}$'s transition function using the letters from u. The acceptance condition in DPAs is based on the color of the states (as given by c). An infinite run in Q^ω is accepting if the minimal color that occurs infinitely often is even. We write $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ for the language of the automaton, which consists of all words where the unique run is accepting. In this paper, we use a parity acceptance condition as DPAs capture every ω-regular property and thus every LTL-expressible property:

Lemma 1 ([29], [30]). *For every LTL formula ψ, we can effectively construct a DPA $\mathcal{A}_\psi = (2^{AP}, Q_\psi, q_{0,\psi}, \delta_\psi, c_\psi)$ such that $\mathcal{L}(\mathcal{A}_\psi) = \{t \in (2^{AP})^\omega \mid t \models_{LTL} \psi\}$.*

We emphasize that our construction also applies to other automaton types. For example, if the LTL body of our hyperproperty can be expressed as a *deterministic* Büchi (resp. safety) automaton, our later construction in Section IV yields a Büchi (resp. safety) game.

HyperLTL: HyperLTL [3] extends LTL with explicit quantification over (execution) traces of the system. Let $\mathcal{V} = \{\pi_1, \pi_2, \ldots\}$ be a set of *trace variables*. HyperLTL formulas are generated by the following grammar

$$\begin{aligned}
\psi &:= a_\pi \mid \psi \land \psi \mid \neg\psi \mid \bigcirc\psi \mid \psi\,\mathcal{U}\,\psi\\
\varphi &:= \forall\pi.\,\varphi \mid \exists\pi.\,\varphi \mid \psi
\end{aligned}$$

where $a \in AP$ is an atomic proposition, and $\pi \in \mathcal{V}$ is a trace variable. Each HyperLTL formula thus has the form $\varphi = \mathbb{Q}_1\pi_1 \ldots \mathbb{Q}_n\pi_n.\,\psi$, where $\mathbb{Q}_1, \ldots, \mathbb{Q}_n \in \{\forall, \exists\}$ are quantifiers, $\pi_1, \ldots, \pi_n \in \mathcal{V}$ are trace variables, and ψ is an LTL formula over trace-variable-indexed APs. The formula quantifies over traces $\pi_1, \ldots, \pi_n$ in the system (in typical first-order semantics) and requires that the resulting combination of n traces satisfies the temporal requirement expressed by the trace-variable-indexed LTL formula ψ.

A trace assignment is a partial function $\Pi : \mathcal{V} \rightharpoonup (2^{AP})^\omega$ that maps trace variables to traces. Given Π we can evaluate the LTL body ψ in each time point $i \in \mathbb{N}$:

$$\begin{aligned}
&\Pi, i \models a_\pi && \text{iff } a \in \Pi(\pi)(i)\\
&\Pi, i \models \psi_1 \land \psi_2 && \text{iff } \Pi, i \models \psi_1 \text{ and } \Pi, i \models \psi_2\\
&\Pi, i \models \neg\psi && \text{iff } \Pi, i \not\models \psi\\
&\Pi, i \models \bigcirc\psi && \text{iff } \Pi, i+1 \models \psi\\
&\Pi, i \models \psi_1 \,\mathcal{U}\, \psi_2 && \text{iff } \exists k \geq i.\, \Pi, k \models \psi_2 \text{ and}\\
&&& \qquad\qquad \forall i \leq j < k.\, \Pi, j \models \psi_1
\end{aligned}$$

Temporal and Boolean operators are evaluated as for LTL. Whenever we evaluate an indexed AP a_π, we look at the trace bound to π and check if a currently holds on this trace. Given a KS $\mathcal{K}$, the quantifier prefix in HyperLTL then adds traces to the trace assignment in typical first-order fashion:

$$\begin{aligned}
&\Pi \models_\mathcal{K} \psi && \text{iff} \quad \Pi, 0 \models \psi\\
&\Pi \models_\mathcal{K} \forall\pi.\,\varphi && \text{iff} \quad \forall t \in \mathit{Traces}(\mathcal{K}).\, \Pi[\pi \mapsto t] \models_\mathcal{K} \varphi\\
&\Pi \models_\mathcal{K} \exists\pi.\,\varphi && \text{iff} \quad \exists t \in \mathit{Traces}(\mathcal{K}).\, \Pi[\pi \mapsto t] \models_\mathcal{K} \varphi
\end{aligned}$$

We say $\mathcal{K}$ satisfies φ, written $\mathcal{K} \models \varphi$, if $\emptyset \models_\mathcal{K} \varphi$, where $\emptyset$ denotes the trace assignment with empty domain.

III. Game-Based Verification of $\forall^*\exists^*$

Model-checking a HyperLTL formula with k quantifier alternations is k-fold exponential [3], [8], and complete methods typically utilize expensive operations like automata complementation [12] and inclusion checking [31]. For $\forall^*\exists^*$ properties, we can soundly (but incompletely) *approximate* the expensive model-checking problem by, instead, constructing a game and searching for a *strategy* that defines witness paths for existentially quantified traces [16], [17].

A. Parity Games

To model the dynamics of the verification game, we use a turn-based game played between a verifier and a refuter (as done by [17]).

A parity game (PG) is a tuple $\mathcal{G} = (V_\mathfrak{V}, V_\mathfrak{R}, v_{init}, \mathbb{D}, E, c)$, where $V := V_\mathfrak{V} \uplus V_\mathfrak{R}$ is the set of game vertices, partitioned into vertices controlled by the verifier ($V_\mathfrak{V}$) and refuter ($V_\mathfrak{R}$), $v_{init} \in V$ is the initial vertex of the game, $\mathbb{D}$ is a set of directions, $E : V \times \mathbb{D} \to V$ is the transition function of the game, and $c : V \to \mathbb{N}$ assigns each state a color used for the parity acceptance condition.

The game is played on an underlying graph whose vertices are controlled by the verifier ($\mathfrak{V}$) or refuter ($\mathfrak{R}$). Whenever the game is in a vertex controlled by a player $p \in \{\mathfrak{V}, \mathfrak{R}\}$, the respective player can determine to which vertex the game should progress by choosing some direction from $\mathbb{D}$. A *strategy* for the verifier is a function $\sigma : V^* \cdot V_\mathfrak{V} \to \mathbb{D}$. The strategy reads a sequence of vertices $v_1 \cdots v_n$ (ending in a vertex $v_n \in V_\mathfrak{V}$ controlled by the verifier) and determines a direction $\sigma(v_1 \cdots v_n) \in \mathbb{D}$ in which the game should progress. A play $\rho \in V^\omega$ is compatible with a strategy σ for $\mathfrak{V}$ if $\rho(0) = v_{init}$ (i.e., the play starts in $\mathcal{G}$'s initial vertex), and for every $i \in \mathbb{N}$, with $\rho(i) \in V_{\mathfrak{V}}$, we have $\rho(i+1) = E\big(\rho(i), \sigma(\rho[0,i])\big)$. That is, we construct the play iteratively; Whenever the game is in a vertex controlled by the verifier, we query strategy σ on the current prefix to obtain a direction and update the vertex based on $\mathcal{G}$'s transition function. We say a play $\rho \in V^\omega$ is *even* if the minimal color that appears infinitely often on ρ (according to coloring c) is even (similar to the acceptance condition used for DPAs). The verifier wins $\mathcal{G}$ if there exists a strategy σ for the verifier such that every play compatible with σ is even.

B. The Verification Game for $\forall^*\exists^*$

Assume a fixed system $\mathcal{K} = (S, s_{init}, \mathbb{D}, \kappa, \ell)$ and a $\forall\exists$ HyperLTL formula $\forall\pi_1.\exists\pi_2.\psi$. For our game construction, we represent the temporal requirement expressed by ψ – the LTL body of φ – as a deterministic automaton. Recall that the atomic propositions in ψ are indexed with trace variables, i.e., ψ is an LTL formula over

$$AP_\psi := \big\{a_\pi \mid a \in AP, \pi \in \{\pi_1, \pi_2\}\big\}.$$

We assume that $\mathcal{A}_\psi = (2^{AP_\psi}, Q_\psi, q_{0,\psi}, \delta_\psi, c_\psi)$ is a DPA over alphabet 2^{AP_ψ} that recognizes ψ, i.e., $\mathcal{L}(\mathcal{A}_\psi) = \{t \in (2^{AP_\psi})^\omega \mid t \models_{LTL} \psi\}$ (cf. Lemma 1). We can then define a parity game, denoted $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$, that captures the iterative construction of traces [17]:

Definition 1 ($\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$, [17]). *Define the parity game $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ by $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi} := (V_{\mathfrak{V}}, V_{\mathfrak{R}}, v_{init}, \mathbb{D}, E, c)$, where*

- $V_{\mathfrak{V}} := \big\{\langle s_1, s_2, q, \mathfrak{V}\rangle \mid s_1, s_2 \in S \uplus \{s_{init}\} \land q \in Q_\psi\big\}$,
- $V_{\mathfrak{R}} := \big\{\langle s_1, s_2, q, \mathfrak{R}\rangle \mid s_1, s_2 \in S \uplus \{s_{init}\} \land q \in Q_\psi\big\}$,
- $v_{init} := \langle s_{init}, s_{init}, q_{0,\psi}, \mathfrak{R}\rangle$,
- *the set of direction $\mathbb{D}$ is the same as in $\mathcal{K}$,*
- *the transition function $E : V \times \mathbb{D} \to V$ is defined by*

$$E\big(\langle s_1, s_2, q, \mathfrak{V}\rangle, d\big) := \langle s_1, \kappa(s_2, d), q, \mathfrak{R}\rangle$$
$$E\big(\langle s_1, s_2, q, \mathfrak{R}\rangle, d\big) :=$$
$$\Big\langle \kappa(s_1, d), s_2, \delta_\psi\Big(q, \bigcup_{i=1}^{2} \{a_{\pi_i} \mid a \in \ell(s_i)\}\Big), \mathfrak{V}\Big\rangle,$$

- $c\big(\langle s_1, s_2, q, p\rangle\big) := c_\psi(q)$.

In our game, each vertex $\langle s_1, s_2, q, p\rangle$ tracks a state s_1 for π_1 (called the π_1-copy), a state s_2 for π_2 (called the π_2-copy), the current state q of $\mathcal{A}_\psi$, and the current player $p \in \{\mathfrak{V}, \mathfrak{R}\}$. In the initial vertex v_{init}, every system copy starts in the initial state, and $\mathcal{A}_\psi$ begins tracking in its initial state $q_{0,\psi}$. Intuitively, in each round of the game, the refuter can update the π_1-copy by moving along some transition in $\mathcal{K}$, followed by the verifier updating the π_2-copy; afterward, the game repeats. Formally, whenever in a vertex $\langle s_1, s_2, q, \mathfrak{V}\rangle$, the verifier can choose a direction $d \in \mathbb{D}$, and the π_2-copy is updated to $\kappa(s_2, d)$ (the first case in the definition of E). Analogously, when in vertex $\langle s_1, s_2, q, \mathfrak{R}\rangle$ the refuter can update the π_1-copy along some direction (the second case in the definition of E). When the refuter moves a round of the game has concluded, so we update the state of $\mathcal{A}_\psi$. For each $i \in \{1, 2\}$, we thus read of the APs that currently hold in state s_i ($\ell(s_i) \subseteq AP$) and index all these APs with π_i to obtain a letter $\bigcup_{i=1}^{2} \{a_{\pi_i} \mid a \in \ell(s_i)\} \in 2^{AP_\psi}$, which we feed to $\mathcal{A}_\psi$'s transition function.

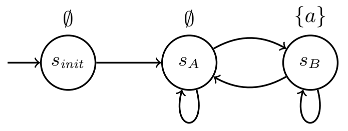

Fig. 1: Simple Kripke structure over $AP = \{a\}$

As we track separate states for π_1 and π_2, every infinite play in $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ defines two paths in $\mathcal{K}$; one for π_1 (where each step is controlled by the refuter), and one for π_2 (controlled by the verifier). In $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$, each vertex $\langle s_1, s_2, q, p\rangle$ is assigned color $c_\psi(q)$ using $\mathcal{A}_\psi$'s coloring function, so an infinite play in $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ is won by the verifier iff the paths constructed for π_1, π_2 during the gameplay are accepted by $\mathcal{A}_\psi$ and thus satisfy ψ. Any winning strategy for $\mathfrak{V}$ thus step-wise constructs a witness trace for π_2, no matter how the refuter constructs π_1. It is not hard to see that the existence of a winning strategy for the verifier thus implies that we can always find a witness trace for π_2 in the HyperLTL semantics:

Lemma 2 ([17]). *If the verifier wins $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$, then $\mathcal{K} \models \varphi$.*

Remark 1. *The game-based approach can be used for automated verification by constructing the game and solving it via an off-the-shelf parity solver (the original motivation of [16], [17]). However, the appeal of Lemma 2 is much broader. For example, the user can construct a strategy by using domain knowledge, which enables* interactive *verification even in situations where automated model-checking does not scale (see, e.g., [18]). Likewise, checking if a given strategy for the verifier wins $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ is often easier than computing a strategy from scratch; strategies are easy-to-check certificates.*

IV. Game-Based Verification Beyond $\forall^*\exists^*$

The game-based approach from the previous section soundly approximates the semantics of $\forall^*\exists^*$ formulas. Unfortunately, the approach is limited to $\forall^*\exists^*$ properties and becomes unsound when considering properties beyond $\forall^*\exists^*$.

Example 1. *Consider the Kripke structure $\mathcal{K}$ over $AP = \{a\}$ in Figure 1 and the $\exists^1\forall^1$ HyperLTL formula*

$$\varphi := \exists\pi_1.\forall\pi_2. (\bigcirc\bigcirc\bigcirc a_{\pi_1}) \leftrightarrow (\bigcirc\bigcirc a_{\pi_2}),$$

where the LTL body expresses that AP a should hold in the third step on π_1 iff it holds in the second step on π_2. Clearly, $\mathcal{K} \not\models \varphi$; no matter what fixed *trace we choose for π_1, we can always find a trace for π_2 that violates the LTL body.*

Now let us naïvely adopt the game used in Section III to this $\exists\forall$ property. That is, we, again, maintain states for all trace variables and let the verifier and refuter iteratively update the existentially and universally quantified system copies, respectively. The resulting game is won by the verifier: During the gameplay, the refuter (who controls the state for the universally quantified π_2) has to decide in the second *step* *whether or not AP a should hold (by moving to s_B or s_A). Only later in the game (in the third round) does the verifier choose if a holds on π_1. By that time, the verifier can thus react to what the refuter has done in the previous step and set a on π_1 appropriately, ensuring that $(\bigcirc\bigcirc\bigcirc a_{\pi_1}) \leftrightarrow (\bigcirc\bigcirc a_{\pi_2})$ is satisfied. The game is won by the verifier, even though the property does not hold.*

A. Multiplayer Games and Partial Information

In this paper, we propose a novel game-based approximation that applies to arbitrary quantifier structures. Our simple yet powerful observation is that the unsoundness is directly linked to the *knowledge* of the player. In Example 1, the verifier could win the game as it can observe what the refuter did in the previous steps. In contrast, in the semantics of an $\exists\forall$ property, any witness for the existential quantifier must be *independent* of the choice for the universal quantifier. By identifying knowledge as the core reason for unsoundness, we can extend the game-based approach to *arbitrary* HyperLTL formulas by utilizing *partial information*. The technical challenge is then to link the knowledge/information of a player to the first-order semantics of HyperLTL.

As our underlying game formalism, we use (sequential) multiplayer parity games [32], which extend parity games with multiple players and partial information. A multiplayer parity game under incomplete information (MPG$_{ii}$) is a tuple

$$\mathcal{G} = (\mathbb{P}, \{V_p\}_{p\in\mathbb{P}}, v_{init}, \mathbb{D}, E, \{\sim_p\}_{p\in\mathbb{P}}, c),$$

where $\mathbb{P}$ is a finite set of players; for each $p \in \mathbb{P}$, V_p is a finite set of vertices controlled by p. We write $V := \biguplus_{p\in\mathbb{P}} V_p$ for the set of all vertices in the game (which we assume to be disjoint). $v_{init} \in V$ is the initial vertex of the game, $\mathbb{D}$ is a set of directions, and $E : V \times \mathbb{D} \to V$ is the transition function of the game. For each player $p \in \mathbb{P}$, $\sim_p \subseteq V \times V$ is an equivalence relation on the set of vertices. Lastly, $c : V \to \mathbb{N}$ assigns each vertex a color.

An MPG$_{ii}$ models the joint behavior of the players in $\mathbb{P}$, where each player $p \in \mathbb{P}$ controls a disjoint set of vertices V_p. If the game is currently in a vertex in V_p, player p can determine where the game should move next by choosing some direction from $\mathbb{D}$. We obtain a standard two-player game (cf. Section III) by setting $\mathbb{P} = \{\mathfrak{V}, \mathfrak{R}\}$. Moreover, each player $p \in \mathbb{P}$ is assigned an indistinguishability relation $\sim_p$, i.e., if $v \sim_p v'$, player p cannot distinguish between v and v'. As usual, we assume that each player is at least able to distinguish whether or not it controls the current vertex. That is, for every $v \sim_p v'$, we either have $v, v' \in V_p$ or $v, v' \in V \setminus V_p$.

Strategies and Plays: Two finite plays $\rho_1, \rho_2 \in V^*$ are indistinguishable for player $p \in \mathbb{P}$, written $\rho_1 \sim_p \rho_2$, if $|\rho_1| = |\rho_2|$ and for every $0 \leq i < |\rho_1|$, $\rho_1(i) \sim_p \rho_2(i)$. A strategy for player $p \in \mathbb{P}$ is a function $\sigma_p : V^* \cdot V_p \to \mathbb{D}$, such that $\sigma_p(\rho_1) = \sigma_p(\rho_2)$ whenever $\rho_1 \sim_p \rho_2$. The strategy reads a sequence of vertices $v_1 \cdots v_n$ (ending in a vertex $v_n \in V_p$) and determines a direction $\sigma_p(v_1 \cdots v_n) \in \mathbb{D}$ in which the game should progress. This decision must conform to the agent's observations, i.e., if two finite plays appear indistinguishable for p, strategy σ_p must choose the same direction on both plays. Within the game, we obtain infinite plays in $\mathcal{G}$ by letting strategies for all players interact. A *global strategy* $\{\sigma_p\}_{p\in\mathbb{P}}$ assigns each player $p \in \mathbb{P}$ a strategy σ_p. Every global strategy $\{\sigma_p\}_{p\in\mathbb{P}}$ defines a unique infinite play $\rho \in V^\omega$, where $\rho(0) = v_{init}$ (i.e., the play starts in $\mathcal{G}$'s initial vertex), and for every $i \in \mathbb{N}$, we have $\rho(i+1) = E\big(\rho(i), \sigma_p(\rho[0,i])\big)$ where $p \in \mathbb{P}$ is the unique player with $\rho(i) \in V_p$. That is, whenever the game is currently in a vertex controlled by player p, we query p's strategy on the current prefix $\rho[0,i]$ to obtain a direction and update the game's vertex along that direction. As before, we say a play ρ is *even* if the minimal color that appears infinitely often on ρ is even. We are interested in checking if a given *coalition* of players $A \subseteq \mathbb{P}$ can win the game. A coalition $A \subseteq \mathbb{P}$ wins $\mathcal{G}$, written $wins(A, \mathcal{G})$, if there exists strategies $\{\sigma_p\}_{p\in A}$ for the players in A, such that, no matter what strategies other players use, i.e., for every possible $\{\sigma_p\}_{p\in\mathbb{P}\setminus A}$, the play resulting from the combined global strategy $\{\sigma_p\}_{p\in\mathbb{P}}$ is even. That is, if the players in A follow their strategies in $\{\sigma_p\}_{p\in A}$, the resulting play satisfies the parity winning condition no matter how the other players behave.

B. HyperLTL Verification as an MPG$_{ii}$

We now present the core contribution of this paper: MPG$_{ii}$s allow for the sound verification of *arbitrary* HyperLTL formulas. For this, assume

$$\varphi = \mathbb{Q}_1\pi_1 \dots \mathbb{Q}_n\pi_n.\, \psi$$

is a fixed HyperLTL formula over trace variables $\pi_1, \dots, \pi_n$ with *arbitrary* quantifier structure. As in Section III, we assume that $\mathcal{A}_\psi = (2^{AP_\psi}, Q_\psi, q_{0,\psi}, \delta_\psi, c_\psi)$ is a DPA over alphabet $AP_\psi := \{a_\pi \mid a \in AP, \pi \in \{\pi_1, \dots, \pi_n\}\}$ that recognizes ψ (cf. Lemma 1).

Definition 2 ($\mathcal{G}_{\mathcal{K},\varphi}$). *Define the MPG$_{ii}$ $\mathcal{G}_{\mathcal{K},\varphi}$ by*

$$\mathcal{G}_{\mathcal{K},\varphi} := (\mathbb{P}, \{V_p\}_{p\in\mathbb{P}}, v_{init}, \mathbb{D}, E, \{\sim_p\}_{p\in\mathbb{P}}, c),$$

where

- $\mathbb{P} := \{1, \dots, n\}$,
- *for each* $p \in \mathbb{P}$,

$$V_p := \{\langle s_1, \dots, s_n, q, p\rangle \mid s_1, \dots, s_n \in S \uplus \{s_{init}\}, q \in Q_\psi\},$$

- $v_{init} := \langle s_{init}, \dots, s_{init}, q_{0,\psi}, 1\rangle$,
- *the set of direction $\mathbb{D}$ is the same as in $\mathcal{K}$,*
- *the transition function $E : V \times \mathbb{D} \to V$ is defined by*

$$E\Big(\big\langle s_1, \dots, s_n, q, p\big\rangle, d\Big) := \Big\langle s_1, \dots, s_{p-1}, \kappa(s_p, d), s_{p+1}, \dots, s_n, q, nxt(p)\Big\rangle,$$

for $p > 1$ and for $p = 1$ define

$$E\Big(\big\langle s_1, \dots, s_n, q, 1\big\rangle, d\Big) := \Big\langle \kappa(s_1, d), s_2, \dots, s_n, \delta_\psi\Big(q, \bigcup_{i=1}^{n} \big\{a_{\pi_i} \mid a \in \ell(s_i)\big\}\Big), nxt(1)\Big\rangle,$$

where $nxt(p) := p+1$ if $p < n$ and $nxt(n) := 1$,

- *for each $p \in \mathbb{P}$,*

$$\sim_p := \Big\{ \big(\langle s_1, \ldots, s_n, q, p' \rangle, \langle s'_1, \ldots, s'_n, q', p'' \rangle\big) \mid p' = p'' \land \forall 1 \leq j \leq p.\, s_j = s'_j \Big\},$$

- *$c\big(\langle s_1, \ldots, s_n, q, p\rangle\big) := c_\psi(q)$.*

The first key idea is to represent each trace variable in the formula as a separate player, so $\mathbb{P} = \{1, \ldots, n\}$. The vertices in $\mathcal{G}_{\mathcal{K},\varphi}$ are of the form $\langle s_1, \ldots, s_n, q, p\rangle$, where $s_1, \ldots, s_n$ track the current state of the system copies for $\pi_1, \ldots, \pi_n$, respectively,, $q \in Q_\psi$ tracks ψ, and $p \in \mathbb{P}$ determines which player controls this vertex. Each infinite play in $\mathcal{G}_{\mathcal{K},\varphi}$, therefore, defines n concrete paths (and thus traces) for $\pi_1, \ldots, \pi_n$. As expected, we start each system in $\mathcal{K}$'s initial state s_{init}, start $\mathcal{A}_\psi$ in $q_{0,\psi}$, and let player 1 begin. Similar to Section III, a vertex $\langle s_1, \ldots, s_n, q, p\rangle$ is assigned color $c_\psi(q)$. The transition function then allows the players to update their system copy. Whenever in a vertex $\langle s_1, \ldots, s_n, q, p\rangle$ where $p > 1$ (the first case in the definition of E), the direction d (which is chosen by player p controlling this vertex) updates the π_p-copy by moving along the chosen direction $d \in \mathbb{D}$ to $\kappa(s_p, d)$. Afterward, it is the next player's turn: $nxt(p)$ increases p by 1 and cycles back to player 1 once the last player (player n) has acted. The players thus take turns updating their system state; first, player 1 updates the state of the π_1-copy, then player 2 updates the state of the π_2-copy, and so forth, until, finally, player n updates the state of the π_n-copy, and the game repeats with player 1. When it is player 1's turn (the second case in the definition of E), a game round has just concluded. In this case, the direction chosen by player 1 is used to update the π_1 copy (similar to the other rounds). At the same time, we update the automaton state of $\mathcal{A}_\psi$ by reading the AP evaluation of $s_1, \ldots, s_n$, obtaining a letter $\bigcup_{i=1}^{n} \{a_{\pi_i} \mid a \in \ell(s_i)\} \in 2^{AP_\psi}$.

The last key ingredient is the partial information of each player. The core problem of the game from Section III was that the players could observe the global state of the game, leading to unsound behavior. MPG$_{ii}$s allow us to precisely determine the information that each player can act on. Once we have observed that knowledge is the key to a sound verification game, we can align the player's information with the HyperLTL semantics: In a HyperLTL formula $\mathbb{Q}\pi_1 \ldots \mathbb{Q}\pi_{i-1}.\exists\pi_i.\mathbb{Q}\pi_{i+1} \ldots \mathbb{Q}\pi_n.\, \psi$, the choice for π_i is only based on the traces $\pi_1, \ldots, \pi_{i-1}$ as those are the traces that are already added to the trace assignment in the semantics of HyperLTL. We can directly express this in our game definition: For a player p (that controls π_p), two vertices $\langle s_1, \ldots, s_n, q, p'\rangle$ and $\langle s'_1, \ldots, s'_n, q', p''\rangle$ appear indistinguishable if it is the same player's turn ($p' = p''$) and $s_j = s'_j$ for all $j \leq p$, i.e., the state of all traces quantified *before* π_p agrees.

C. Soundness

To use our game as a verification method, we are interested in the strategic ability of all players controlling existentially quantified system copies. That is, we define

$$\mathbb{P}_\exists := \{i \in \mathbb{P} \mid \mathbb{Q}_i = \exists\}.$$

We can then show that our game under partial information constitutes a sound verification approach:

Theorem 1. *If $wins(\mathbb{P}_\exists, \mathcal{G}_{\mathcal{K},\varphi})$, then $\mathcal{K} \models \varphi$.*

Example 2. *Consider some formula $\forall\pi_1.\exists\pi_2.\forall\pi_3.\psi$. In our game definition, the player controlling π_2 can observe the current state of the π_1-copy and thus react to the behavior of player 1. However, it cannot base its decision on the behavior of player 3. In the special case of $\exists\pi_1.\forall\pi_2$ properties (like in Example 1), player 1 can only observe its own state. It must thus use the same sequence of directions (and thus define the* same *witness paths), no matter how player 2 behaves (cf. [33]). In Example 1, player 1 would thus lose this game; there is no winning behavior in the third step without observing player 2's behavior in the second step.*

It is not hard to see that in the special case of $\forall^*\exists^*$ properties, the MPG$_{ii}$ $\mathcal{G}_{\mathcal{K},\varphi}$ coincides with the game $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ from Section III.

Lemma 3. *Let $\varphi = \forall\pi_1.\,\exists\pi_2.\,\psi$. Then $wins(\{2\}, \mathcal{G}_{\mathcal{K},\varphi})$ if and only if the verifier wins $\mathcal{G}^{\forall\exists}_{\mathcal{K},\varphi}$ (cf. Section III).*

Similar to the $\forall^*\exists^*$ game from Section III, we can use $\mathcal{G}_{\mathcal{K},\varphi}$ not only for automated verification but also as a foundation for interactive verification and certificates (cf. Remark 1).

D. Hierarchical Information

Multiplayer games under imperfect information are often undecidable [20], [34], i.e., there exists no general algorithm to check if a given group of players can win the game. Fortunately, there exists a well-known class of imperfect information games that we can solve effectively. An MPG$_{ii}$ $(\mathbb{P}, \{V_p\}_{p\in\mathbb{P}}, v_{init}, \mathbb{D}, E, \{\sim_p\}_{p\in\mathbb{P}}, c)$ is played under hierarchical information if there exists a total order $\prec$ on $\mathbb{P}$ such that for every $p' \prec p$, we have $\sim_p \subseteq \sim_{p'}$. That is, we can order the players such that the information is hierarchical, i.e., player p observes at least as much as all smaller players (w.r.t. $\prec$). By adopting standard techniques, we can show that we can decide if a given group of players can win a game played under hierarchical information [35]–[37].

Lemma 4. *The MPG$_{ii}$ $\mathcal{G}_{\mathcal{K},\varphi}$ from Definition 2 is played under hierarchical information. Consequently, it is decidable if $wins(\mathbb{P}_\exists, \mathcal{G}_{\mathcal{K},\varphi})$.*

V. Completeness for $\exists^*\forall^*$

In our game-based view, we let players construct witness traces for existentially quantified traces. Compared to the HyperLTL semantics, this limits the power of existential quantification. For example, in the semantics of a $\forall\pi_1.\,\exists\pi_2.\,\psi$ formula, the choice for π_2 can be based on the *entire* trace assigned to π_1. In the game-based view, π_2 is constructed step-wise by a player, so the decision in the ith round of the game can only depend on π_1's prefix of length i. This leads to incompleteness, i.e., situations where a property holds, but the game is not won by $\mathbb{P}_\exists$.

Example 3 ([17]). *Consider the Kripke structure $\mathcal{K}$ in Figure 1 and the HyperLTL formula $\varphi = \forall \pi_1. \exists \pi_2. \Box\Diamond(a_{\pi_2} \leftrightarrow \bigcirc a_{\pi_1})$ which requires that π_2 predicts the next value of π_1 infinitely often. Clearly, $\mathcal{K} \models \varphi$, but $\neg wins(\{2\}, \mathcal{G}_{\mathcal{K},\varphi})$: In $\mathcal{G}_{\mathcal{K},\varphi}$, player 2 has to decide in each step whether AP a should hold but does not know what player 1 will do on trace π_1 in the future. No matter what player 2 does, player 1 can thus always ensure that $a_{\pi_2} \not\leftrightarrow \bigcirc a_{\pi_1}$.*

Our game is thus incomplete; already on $\forall^*\exists^*$ properties where our game coincides with the full-information games from [17], cf. Lemma 3. While incomplete for $\forall^*\exists^*$, our game is, perhaps surprisingly, complete for $\exists^*\forall^*$ properties.

Theorem 2. *Assume φ is a $\exists^*\forall^*$ HyperLTL formula. Then $wins(\mathbb{P}_\exists, \mathcal{G}_{\mathcal{K},\varphi})$ if and only if $\mathcal{K} \models \varphi$.*

Note that we can check any $\forall^*\exists^*$ property by checking the negated property (which is $\exists^*\forall^*$). Our game-based approach, therefore, constitutes a sound-and-complete model-checking technique for all HyperLTL formulas with *at most one* quantifier alternation. The cost of this completeness manifests itself in the additional complexity; our game-based approach for $\forall^*\exists^*$ properties (which coincides with [17]) is incomplete but results in a standard game under full information. If we, instead, check the negated $\exists^*\forall^*$ formula, our game is complete, but the game uses partial information, making automated game-solving more challenging.

VI. Prophecy Variables

For properties beyond $\exists^*\forall^*$, we need additional tools to counteract the incompleteness. In this section, we study the use of prophecy variables in our game-based framework; a technique originally studied in [16], [17] for the verification of $\forall^*\exists^*$ hyperproperties (building on the seminal work by Abadi and Lamport [27]). At a high level, a prophecy provides limited information about the future behavior of other players. For example, in the setting of a $\forall \pi_1. \exists \pi_2. \psi$ formula, the player controlling π_2 only observes the past behavior of the player controlling π_1. For such a formula, a prophecy is an LTL formula ξ over trace variable π_1 (cf. [17]). In each step of the game, the player controlling π_2 can then query an oracle that tells them whether or not the future behavior of π_1 will satisfy ξ, and base its decision on the additional information provided by the oracle.

Example 4. *In Example 3, the player controlling π_2 does not have a winning strategy as it does not know if a will hold on π_1 in the* next *step. In this example, it suffices for the player controlling π_2 to have access to an oracle that, in each step, predicts whether prophecy $\xi := \bigcirc a_{\pi_1}$ holds (cf. [17]). If ξ holds (so $\bigcirc a_{\pi_1}$), the player can move the π_2-copy to state s_B (where a holds, cf. Figure 1), thus ensuring that $a_{\pi_2} \leftrightarrow \bigcirc a_{\pi_1}$.*

Prophecies essentially combat the problem of having *too little* information about the *future*, which already leads to *incompleteness* in the $\forall^*\exists^*$ setting. The main contribution of this paper is that we can use partial information to avoid players having *too much* information about the behavior of (certain) other players, which would lead to *unsoundness*. In this section, we extend the prophecy framework of [17] from $\forall^*\exists^*$ to arbitrary quantifier prefixes. Our prophecy construction combines two ideas: our observation model ensures that players do not observe too much (to ensure soundness), while prophecies provide missing information about future events.

A. Prophecies and Partial Information

To keep notation simple, we assume for the remainder of this section – and w.l.o.g. – that the hyperproperty in question is of the form $\varphi = \forall \pi_1. \exists \pi_2. \forall \pi_3 \dots \forall \pi_{2n-1}. \exists \pi_{2n}. \psi$, i.e., strictly alternates between universal (at odd indices) and existential (at even indices) quantification.

Example 5. *Consider the formula*

$$\varphi := \forall \pi_1. \exists \pi_2. \forall \pi_3. \exists \pi_4. \Box\Diamond(a_{\pi_2} \leftrightarrow \bigcirc a_{\pi_1}) \land \\ \bigcirc\bigcirc \big(a_{\pi_4} \leftrightarrow \Box(a_{\pi_1} \leftrightarrow a_{\pi_2} \land a_{\pi_2} \leftrightarrow a_{\pi_3})\big).$$

It requires π_2 to globally predict the next step of π_1, and π_4 should, in the second step, predict whether π_1, π_2, and π_3 agree on a. The KS $\mathcal{K}$ in Figure 1 satisfies φ, yet coalition $\{2, 4\}$ loses $\mathcal{G}_{\mathcal{K},\varphi}$. To win this game, the player controlling π_2 needs (in every step) information about the future of π_1, and the player controlling π_4 needs (in the second step) information about the joint future of π_1, π_2, and π_3.

B. Prophecies and Prophecy Variables

To ensure soundness, we need to ensure that each player i is (via the prophecies) only given information over traces quantified *before* π_i. For each existentially quantified trace π_{2i}, we therefore track a separate set of prophecies:

Definition 3. *A* prophecy family *is a collection $\vec{\Xi} = \{\Xi_{2i-1}\}_{i=1}^n$, where Ξ_{2i-1} is a finite set of LTL formulas over trace variables from $\{\pi_1, \dots, \pi_{2i-1}\}$.*

Intuitively, Ξ_{2i-1} contains all prophecies that provide information to the player controlling trace π_{2i}. Consequently, the formulas in Ξ_{2i-1} only reason about traces $\{\pi_1, \dots, \pi_{2i-1}\}$, which are exactly the traces player i can observe in the game.

Example 6. *Consider the property in Example 5. We can construct the prophecy family $\{\Xi_1, \Xi_3\}$, where $\Xi_1 := \{\bigcirc a_{\pi_1}\}$, and $\Xi_3 := \{\Box(a_{\pi_1} \leftrightarrow a_{\pi_2} \land a_{\pi_2} \leftrightarrow a_{\pi_3})\}$. These prophecies provide exactly the information needed by players 2 and 4. That is, if players 2 and 4 could,* in each step of the game*, determine if the prophecies in Ξ_1 and Ξ_3 hold, respectively, they can construct appropriate witness traces and win $\mathcal{G}_{\mathcal{K},\varphi}$.*

Now assume we have a fixed family of LTL formulas $\vec{\Xi} = \{\Xi_{2i-1}\}_{i=1}^n$. The intuition behind the prophecies is that each player π_{2i} is provided with an oracle that – in each step of the game– tells her which of the formulas in Ξ_{2i-1} hold. Following [17], we will use *prophecy variables* to formalize this oracle. A prophecy variable is essentially an AP that we add to the system, and we ensure that the value of this variable (AP) reflects the truth value of the prophecy formula. For this, we assume that P is a set of prophecy variables for $\vec{\Xi}$, i.e., for each $1 \leq i \leq n$ and $\xi \in \Xi_{2i-1}$, there exists a corresponding prophecy variable $p^\xi \in P$. A player can thus query an oracle on whether prophecy ξ currently holds, by simply reading the prophecy variable (AP) p^ξ. The key idea now is that we can attach these prophecy variables to *universally* quantified traces [17]. That is, we let the opposing players $\mathbb{P} \setminus \mathbb{P}_\exists$ (controlling the universally quantified traces) determine the truth value of the prophecy variables, and then ensure, within the HyperLTL formula, that the prophecy variables are set correctly: That is, we modify the HyperLTL formula to ensure that $\xi \in \Xi_{2i-1}$ holds iff the prophecy variable p^ξ is set to true on trace π_{2i-1}. This ensures that the player $2i$ controlling π_{2i} can query the prophecies in Ξ_{2i-1} by looking at the current state of π_{2i-1} (and the Boolean value of the prophecy variables, i.e., AP, in that state), but the players controlling $\pi_1, \ldots, \pi_{2i-2}$ cannot.

As a first step, we add the variables in P to the system, which can be set arbitrarily in each step:

Definition 4 ($\mathcal{K}^P$)**.** *Given a KS $\mathcal{K} = (S, s_{init}, \mathbb{D}, \kappa, \ell)$ over AP and a disjoint set of prophecy variables P ($AP \cap P = \emptyset$), define the modified KS $\mathcal{K}^P := (S \times 2^P, s_{init}, \mathbb{D} \times 2^P, \kappa^P, \ell^P)$ over $AP \uplus P$ where for each direction $(d, A) \in \mathbb{D} \times 2^P$, we define κ^P by*

$$\kappa^P(s_{init}, (d, A)) := (\kappa(s_{init}, d), A)$$
$$\kappa^P((s, A'), (d, A)) := (\kappa(s, d), A)$$

and $\ell^P(s_{init}) = \ell(s_{init})$ and $\ell^P(s, A) := \ell(s) \cup A$.

Intuitively, $\mathcal{K}^P$ generates all traces of $\mathcal{K}$ extended with all possible evaluations on the prophecy variables.

We then modify the body of the HyperLTL formula such that the original property is only required to hold if all prophecies are set correctly, i.e., a prophecy variable in p^ξ is set to true iff the future traces satisfy ξ.

Definition 5 ($\varphi^{P,\vec{\Xi}}$)**.** *Given a prophecy family $\vec{\Xi} = \{\Xi_{2i-1}\}_{i=1}^n$ and a corresponding set of prophecy variables P, define the modified HyperLTL formula $\varphi^{P,\vec{\Xi}}$ by*

$$\varphi^{P,\vec{\Xi}} := \forall \pi_1 . \exists \pi_2 . \forall \pi_3 \ldots \forall \pi_{2n-1} . \exists \pi_{2n} .$$
$$\left(\bigcirc \Box \bigwedge_{i=1}^{n} \Big(\bigwedge_{\xi \in \Xi_{2i-1}} \big((p^\xi)_{\pi_{2i-1}} \leftrightarrow \xi \big) \Big) \right) \to \psi.$$

That is, we require ψ (the original LTL body of φ) to hold, if in every step, for every $1 \leq i \leq n$, and every prophecy formula $\xi \in \Xi_{2i-1}$, ξ holds iff the corresponding prophecy variable p^ξ holds on trace π_{2i-1}. Note how $\varphi^{P,\vec{\Xi}}$ connects the prophecy variables with the underlying prophecy. If some prophecy variable p^ξ for $\xi \in \Xi_{2i-1}$ is set on (the universally quantified) trace π_{2i-1}, the player $2i \in \mathbb{P}_\exists$ controlling the existentially quantified trace π_{2i} can assume that ξ holds, and vice versa. If this is not the case, i.e., player $2i-1$ sets p^ξ incorrectly, the premise of $\varphi^{P,\vec{\Xi}}$ is violated, so the LTL body of $\varphi^{P,\vec{\Xi}}$ is vacuously true, and $\mathbb{P}_\exists$ wins the game in Definition 2. Note that trace π_{2i-1} is universally quantified, so we consider all possible valuations of the prophecy variables, including the unique valuation where the prophecy variables are set correctly in each step, i.e., $\bigcirc \Box \bigwedge_{\xi \in \Xi_{2i-1}} \big((p^\xi)_{\pi_{2i-1}} \leftrightarrow \xi \big)$. Observe that we only require the prophecies to be set correctly after the first step (using a single $\bigcirc$), as the unique initial state s_{init} of a KS does not record prophecy variables.

C. Prophecies and Games

We can use the information provided via prophecies in our game-based approach. Instead of checking if the players in $\mathbb{P}_\exists$ win $\mathcal{G}_{\mathcal{K},\varphi}$, we can then check if they win $\mathcal{G}_{\mathcal{K}^P,\varphi^{P,\vec{\Xi}}}$. In the latter game, the additional premise in $\varphi^{P,\vec{\Xi}}$ ensures that the players can use prophecies to peek at future moves of the universal traces.

Example 7. *We consider the example from Examples 5 and 6. We already argued that $\mathbb{P}_\exists = \{2, 4\}$ loses $\mathcal{G}_{\mathcal{K},\varphi}$. Now consider the prophecy family $\vec{\Xi}$ from Example 6, with corresponding prophecy variables $P := \{\{p\}, \{pp\}\}$. Using Definition 5, we construct*

$$\varphi^{P,\vec{\Xi}} = \forall \pi_1 . \exists \pi_2 . \forall \pi_3 . \exists \pi_4 . \Big[\bigcirc \Box \big(p_{\pi_1} \leftrightarrow \bigcirc a_{\pi_1} \big) \land$$
$$\bigcirc \Box \big(pp_{\pi_3} \leftrightarrow \Box (a_{\pi_1} \leftrightarrow a_{\pi_2} \land a_{\pi_2} \leftrightarrow a_{\pi_3}) \big) \Big] \to \psi,$$

where ψ is the original LTL body (cf. Example 5). It is easy to see that $\{2, 4\}$ wins $\mathcal{G}_{\mathcal{K}^P,\varphi^{P,\vec{\Xi}}}$: the prophecy variable p on π_1 hints at the next move of π_1. If, for example, player 1 sets p to true, player 2 can assume that $\bigcirc a_{\pi_1}$ holds (if it does not, the premise of $\varphi^{P,\vec{\Xi}}$'s LTL body is violated, and so the play is trivially won by $\{2, 4\}$). Likewise, the prophecy variable pp provides the necessary information for player 4.

D. Soundness

The key result we are left to prove is that the addition of prophecies does not change the HyperLTL semantics, even though the LTL body of $\varphi^{P,\vec{\Xi}}$ is weaker than the body of φ.

Theorem 3. *Assume a prophecy family $\vec{\Xi} = \{\Xi_{2i-1}\}_{i=1}^n$ and a corresponding set of prophecy variables P. Then $\mathcal{K} \models \varphi$ if and only if $\mathcal{K}^P \models \varphi^{P,\vec{\Xi}}$.*

Theorem 3 allows us to soundly combine prophecies with the game-based approach: If $wins(\mathbb{P}_\exists, \mathcal{G}_{\mathcal{K}^P,\varphi^{P,\vec{\Xi}}})$, then, by Theorem 1, we have $\mathcal{K}^P \models \varphi^{P,\vec{\Xi}}$, so, by Theorem 3, we have $\mathcal{K} \models \varphi$. Prophecy variables thus constitute a tool that can strengthen game-based verification in the presence of arbitrary quantifier alternations. This is particularly relevant when using our approach as an interactive proof technique. Once a suitable set of prophecies is found (i.e., a family $\vec{\Xi}$ s.t. $\mathbb{P}_\exists$ wins $\mathcal{G}_{\mathcal{K}^P,\varphi^{P,\vec{\Xi}}}$), the prophecies, together with the winning strategies for $\mathbb{P}_\exists$, are an easy-to-check certificate of satisfaction.

VII. Related Work

Logics for Hyperproperties: Most logics for expressing temporal hyperproperties use HyperLTL-style quantification over execution traces [8]–[10], [38]. In such logics, quantifier alternations are frequently used to, e.g., reason about non-determinism in the system. Our paper proposes a principled approach to deal with quantifier alternations that can be easily extended to other logics that feature HyperLTL-style quantification over system paths/traces.

HyperLTL Verification: Finite-state model-checking of HyperLTL is decidable [3], and complete algorithms rely on expensive automata complementation or language inclusion checks [12], [31]. To approximate this expensive problem, Hsu et al. [39], [40] propose a bounded model-checking approach for HyperLTL by unrolling the system and property into a QBF formula. The other prominent approximation for HyperLTL is the game-based approach [16], [17], which forms the foundation of the present paper. Both approximations are orthogonal to each other. In the game-based approach, we use strategies to resolve existential quantification and can thus reason about temporal behavior along *infinite* paths. In contrast, the QBF-based encoding features the same first-order semantics used in HyperLTL, but bounds the length of the traces, limiting the approach to properties that can be verified or refuted within a bounded timeframe.

Advantages of Game-based Verification: The game-based approach has multiple advantages over automata-complementation-based methods. Firstly, it allows verification in settings where complementation-based approaches fail. For example, the game-based approach can be used to verify *infinite-state* systems by constructing abstract games [41], [42], or utilizing infinite-state game solvers [43]–[47]. Secondly, the game-based approach allows for *interactive proofs*, i.e., the user can manually construct a proof by incrementally defining a winning strategy [18]. This facilitates proofs in situations where automated methods do not scale. And lastly, the game-based approach naturally yields *certificates* of satisfaction, i.e., the computed winning strategy (combined with a set of prophecies, if needed) can easily be checked by independent strategy checkers [19].

In this paper, we propose an extension of the game-based approach to arbitrary quantifier structures, based on the key conceptual contribution of leveraging *partial information*, extending our earlier extended abstract [48]. Based on this key conceptual idea, we provide a sound framework that allows us to utilize the benefits of game-based verification for arbitrary quantifier structures. While solving games under partial information is expensive, our paper also allows for cheaper approaches that build upon the game-based interpretation. For example, we can attempt to find *positional* strategies for $\mathbb{P}_\exists$. Finding positional strategies is much cheaper than finding strategies under perfect recall (i.e., unbounded memory), and, if winning positional strategies for $\mathbb{P}_\exists$ are found, our results allow us to soundly conclude that $\mathcal{K} \models \varphi$. This creates a spectrum of techniques with varying complexity and expressiveness (i.e., in some instances, positional strategies suffice, in others, we might need bounded memory or even perfect recall). In the case of $\exists^*\forall^*$ properties, our definition yields a game where the player acts without any information, a setting explored extensively within the planning community (cf. *conformant planning*) [33], [49].

Solving Games Under Incomplete Information: We believe that the primary use case of our approach lies in its ability to construct sound-by-design interactive proofs and certificates. Nevertheless, if paired with a solver for games under partial information, our approach could underpin a fully-automated verification pipeline. The study of (multiplayer) games under incomplete information has a long tradition, mostly relying on using a powerset construction or lattice framework to track belief states [24], [25], [35], [36], [50], [51]. Strategy-based logic can explicitly reason about the strategic abilities of players, and extensions to incomplete information exist [20], [23], [52]–[54]. Many of the frameworks used to study incomplete information can be used in our setting: for example, partially observable non-deterministic (POND) planning reasons about an agent that acts in a non-deterministic environment, essentially defining a game under partial information [55], which we can use for hyperproperty verification [33], [56]. Likewise, multi-agent planning [57], [58], partially observable Markov decision processes with multiple agents [59], [60], or multi-agent reinforcement learning [61] study strategy synthesis in partially observable domains.

VIII. Conclusion and Future Work

In this work, we have proposed a novel hyperproperty verification method using games. In contrast to previous game-based approaches, our method is not limited to $\forall^*\exists^*$ properties but soundly approximates *arbitrary* quantifier structures. Moreover, we designed a prophecy mechanism that aligns with the partial information of the player. Our work creates numerous avenues for future work, both in theory and practice. In theory, it is interesting to study the expressive power of our game-based approximation when using prophecies. Prophecies are a complete proof technique in the setting of $\forall^*\exists^*$ properties, i.e., whenever a property holds, there exists some finite set of ω-regular (not necessarily LTL-definable) prophecies such that the game is won by the verifier [17]. The high-level idea of this construction is to construct prophecies that directly determine successor states, i.e., prophecies $\xi_{s,s'}$ that hold iff, when in state s, moving to s' is the "optimal" move for the verifier (see [17] for details). We conjecture that completeness also holds in the presence of arbitrary quantifier alternations: For a property $\mathbb{Q}_1\pi_1 \ldots \mathbb{Q}_{i-1}\pi_{i-1}\exists\pi_i.\mathbb{Q}_{i+1}\pi_{i+1} \ldots \mathbb{Q}_n\pi_n.\psi$, we can design prophecies that precisely define the optimal behavior for player i (for a fixed system $\mathcal{K}$, $\mathbb{Q}_{i+1}\pi_{i+1} \ldots \mathbb{Q}_n\pi_n.\psi$ is just an ω-regular property over $\pi_1, \ldots, \pi_i$). In practice, we can utilize our games as an interactive proof technique, e.g., using the coinductive framework of [18].

Acknowledgments

This work was supported by the European Research Council (ERC) Grant HYPER (101055412), and by the German Research Foundation (DFG) as part of TRR 248 (389792660).

References

[1] M. R. Clarkson and F. B. Schneider, "Hyperproperties," in *Computer Security Foundations Symposium, CSF 2008*, 2008.

[2] A. Pnueli, "The temporal logic of programs," in *Symposium on Foundations of Computer Science, FOCS 1977*, 1977.

[3] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *International Conference on Principles of Security and Trust, POST 2014*, 2014.

[4] S. Zdancewic and A. C. Myers, "Observational determinism for concurrent program security," in *Computer Security Foundations Workshop, CSFW 2003*, 2003.

[5] W. van der Hoek and M. J. Wooldridge, "Model checking knowledge and time," in *International Workshop on Model Checking of Software, SPIN 2002*, 2002.

[6] R. Beutner, B. Finkbeiner, H. Frenkel, and N. Metzger, "Second-order hyperproperties," in *International Conference on Computer Aided Verification, CAV 2023*, 2023.

[7] J. Y. Halpern, "Reasoning about knowledge: An overview," in *Theoretical aspects of reasoning about knowledge*, 1986.

[8] M. N. Rabe, "A temporal logic approach to information-flow control," Ph.D. dissertation, Saarland University, 2016.

[9] J. O. Gutsfeld, M. Müller-Olm, and C. Ohrem, "Propositional dynamic logic for hyperproperties," in *International Conference on Concurrency Theory, CONCUR 2020*, 2020.

[10] N. Coenen, B. Finkbeiner, C. Hahn, and J. Hofmann, "The hierarchy of hyperlogics," in *Symposium on Logic in Computer Science, LICS 2019*, 2019.

[11] G. Barthe, P. R. D'argenio, and T. Rezk, "Secure information flow by self-composition," *Math. Struct. Comput. Sci.*, 2011.

[12] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL*," in *International Conference on Computer Aided Verification, CAV 2015*, 2015.

[13] Y. Wang, S. Nalluri, and M. Pajic, "Hyperproperties for robotics: Planning via hyperltl," in *International Conference on Robotics and Automation, ICRA 2020*, 2020.

[14] S. Biewer, R. Dimitrova, M. Fries, M. Gazda, T. Heinze, H. Hermanns, and M. R. Mousavi, "Conformance relations and hyperproperties for doping detection in time and space," *Log. Methods Comput. Sci.*, 2022.

[15] M. Anand, V. Murali, A. Trivedi, and M. Zamani, "Verification of hyperproperties for dynamical systems via barrier certificates," *IEEE Transactions on Automatic Control*, 2024.

[16] N. Coenen, B. Finkbeiner, C. Sánchez, and L. Tentrup, "Verifying hyperliveness," in *International Conference on Computer Aided Verification, CAV 2019*, 2019.

[17] R. Beutner and B. Finkbeiner, "Prophecy variables for hyperproperty verification," in *Computer Security Foundations Symposium, CSF 2022*, 2022.

[18] A. Correnson and B. Finkbeiner, "Coinductive proofs for temporal hyperliveness," *Proc. ACM Program. Lang.*, no. POPL, 2025.

[19] R. Beutner, B. Finkbeiner, and A. Göbl, "Visualizing game-based certificates for hyperproperty verification," in *International Symposium on Formal Methods, FM 2024*, 2024.

[20] R. Berthon, B. Maubert, and A. Murano, "Decidability results for ATL* with imperfect information and perfect recall," in *Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*, 2017.

[21] D. Berwanger, A. B. Mathew, and M. van den Bogaard, "Hierarchical information and the synthesis of distributed strategies," *Acta Informatica*, 2018.

[22] B. Maubert and A. Murano, "Reasoning about knowledge and strategies under hierarchical information," in *International Conference on Principles of Knowledge Representation and Reasoning, KR 2018*, 2018.

[23] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi, "Strategy logic with imperfect information," *ACM Trans. Comput. Log.*, 2021.

[24] D. Berwanger, K. Chatterjee, M. D. Wulf, L. Doyen, and T. A. Henzinger, "Strategy construction for parity games with imperfect information," *Inf. Comput.*, 2010.

[25] K. Chatterjee and L. Doyen, "The complexity of partial-observation parity games," in *International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2010*, 2010.

[26] P. Gastin, N. Sznajder, and M. Zeitoun, "Distributed synthesis for well-connected architectures," *Formal Methods Syst. Des.*, 2009.

[27] M. Abadi and L. Lamport, "The existence of refinement mappings," in *Symposium on Logic in Computer Science, LICS 1988*, 1988.

[28] R. Beutner and B. Finkbeiner, "On hyperproperty verification, quantifier alternations, and games under partial information," *CoRR*, 2025.

[29] J. Esparza, J. Kretínský, J. Raskin, and S. Sickert, "From LTL and limit-deterministic büchi automata to deterministic parity automata," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2017*, 2017.

[30] N. Piterman, "From nondeterministic büchi and streett automata to deterministic parity automata," *Log. Methods Comput. Sci.*, 2007.

[31] R. Beutner and B. Finkbeiner, "AutoHyper: Explicit-state model checking for HyperLTL," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*, 2023.

[32] V. Malvone, A. Murano, and L. Sorrentino, "Concurrent multi-player parity games," in *International Conference on Autonomous Agents & Multiagent Systems, AAMAS 2016*, 2016.

[33] R. Beutner and B. Finkbeiner, "On conformant planning and model-checking of $\exists^*\forall^*$ hyperproperties," in *European Conference on Artificial Intelligence, ECAI 2025*, 2025.

[34] A. Pnueli and R. Rosner, "Distributed reactive systems are hard to synthesize," in *Symposium on Foundations of Computer Science, FOCS 1990*, 1990.

[35] J. H. Reif, "The complexity of two-player games of incomplete information," *J. Comput. Syst. Sci.*, 1984.

[36] G. Peterson, J. Reif, and S. Azhar, "Decision algorithms for multiplayer noncooperative games of incomplete information," *Computers & Mathematics with Applications*, 2002.

[37] B. Finkbeiner and S. Schewe, "Uniform distributed synthesis," in *Symposium on Logic in Computer Science LICS 2005*, 2005.

[38] L. V. Nguyen, J. Kapinski, X. Jin, J. V. Deshmukh, and T. T. Johnson, "Hyperproperties of real-valued signals," in *International Conference on Formal Methods and Models for System Design, MEMOCODE 2017*, 2017.

[39] T. Hsu, C. Sánchez, and B. Bonakdarpour, "Bounded model checking for hyperproperties," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021*, 2021.

[40] T. Hsu, C. Sánchez, S. Sheinvald, and B. Bonakdarpour, "Efficient loop conditions for bounded model checking hyperproperties," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*, 2023.

[41] R. Beutner and B. Finkbeiner, "Software verification of hyperproperties beyond k-safety," in *International Conference on Computer Aided Verification, CAV 2022*, 2022.

[42] S. Itzhaky, S. Shoham, and Y. Vizel, "Hyperproperty verification as CHC satisfiability," in *European Symposium on Programming, ESOP 2024*, 2024.

[43] P. Battigalli, "Rationalizability in infinite, dynamic games with incomplete information," *Research in Economics*, 2003.

[44] L. de Alfaro, T. A. Henzinger, and R. Majumdar, "Symbolic algorithms for infinite-state games," in *International Conference on Concurrency Theory, CONCUR 2001*, 2001.

[45] P. Heim and R. Dimitrova, "Solving infinite-state games via acceleration," *Proc. ACM Program. Lang.*, no. POPL, 2024.

[46] M. Faella and G. Parlato, "Reachability games modulo theories with a bounded safety player," in *Conference on Artificial Intelligence, AAAI 2023*, 2023.

[47] C. Baier, N. Coenen, B. Finkbeiner, F. Funke, S. Jantsch, and J. Siber, "Causality-based game solving," in *International Conference on Computer Aided Verification, CAV 2021*, 2021.

[48] R. Beutner and B. Finkbeiner, "Multiplayer games with incomplete information for hyperproperty verification," in *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*, 2025.

[49] R. P. Goldman and M. S. Boddy, "Expressive planning and explicit knowledge," in *International Conference on Artificial Intelligence Planning Systems, AIPS 1996*, 1996.

[50] J. Raskin, K. Chatterjee, L. Doyen, and T. A. Henzinger, "Algorithms for omega-regular games with imperfect information," *Log. Methods Comput. Sci.*, 2007.

[51] M. D. Wulf, L. Doyen, and J. Raskin, "A lattice theory for solving games of imperfect information," in *International Workshop on Hybrid Systems: Computation and Control, HSCC 2006*, 2006.

[52] R. Beutner and B. Finkbeiner, "Hyper strategy logic," in *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024*, 2024.

[53] J. Pilecki, M. A. Bednarczyk, and W. Jamroga, "Model checking properties of multi-agent systems with imperfect information and imperfect recall," in *International Conference on Intelligent Systems*, 2014.

[54] R. Beutner and B. Finkbeiner, “Strategy logic, imperfect information, and hyperproperties,” in *International Conference on Principles of Knowledge Representation and Reasoning, KR 2025*, 2025.
[55] R. I. Brafman, G. Shani, and S. Zilberstein, “Qualitative planning under partial observability in multi-agent domains,” in *Conference on Artificial Intelligence, AAAI 2013*, 2013.
[56] R. Beutner and B. Finkbeiner, “Non-deterministic planning for hyperproperty verification,” in *International Conference on Automated Planning and Scheduling, ICAPS 2024*, 2024.
[57] P. J. Gmytrasiewicz and P. Doshi, “A framework for sequential planning in multi-agent settings,” *J. Artif. Intell. Res.*, 2005.
[58] M. F. L. Galesloot, T. D. Simão, S. Junges, and N. Jansen, “Factored online planning in many-agent POMDPs,” in *Conference on Artificial Intelligence, AAAI 2024*, 2024.
[59] C. Zhang and V. R. Lesser, “Coordinated multi-agent reinforcement learning in networked distributed POMDPs,” in *Conference on Artificial Intelligence, AAAI 2011*, 2011.
[60] C. Amato and F. A. Oliehoek, “Scalable planning and learning for multiagent POMDPs,” in *Conference on Artificial Intelligence, AAAI 2015*, 2015.
[61] L. Busoniu, R. Babuska, and B. D. Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Trans. Syst. Man Cybern. Part C*, 2008.