Prophecies All the Way: Game-Based Model-Checking for HyperQPTL Beyond $\forall^*\exists^*$

Sarah Winter
IRIF, Université Paris Cité, Paris, France

Martin Zimmermann
Aalborg University, Denmark

Abstract

Model-checking HyperLTL, a temporal logic expressing properties of sets of traces with applications to information-flow based security and privacy, has a decidable, but TOWER-complete, model-checking problem. While the classical model-checking algorithm for full HyperLTL is automata-theoretic, more recently, a game-based alternative for the $\forall^*\exists^*$-fragment has been presented.

Here, we employ imperfect information-games to extend the game-based approach to full HyperQPTL, which features arbitrary quantifier prefixes and quantification over propositions and can express every ω-regular hyperproperty. As a byproduct of our game-based algorithm, we obtain finite-state implementations of Skolem functions via transducers with lookahead that explain satisfaction or violation of HyperQPTL properties.

2012 ACM Subject Classification Theory of computation $\to$ Logic and verification; Theory of computation $\to$ Modal and temporal logics; Theory of computation $\to$ Verification by model checking

Keywords and phrases HyperLTL, HyperQPTL, model-checking games, prophecies

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.37

Related Version *Full Version*: https://arxiv.org/abs/2504.08575 [25]

Funding *Martin Zimmermann*: Supported by DIREC - Digital Research Centre Denmark.

1 Introduction

Hyperlogics like HyperLTL and HyperCTL* [6] extend their classical counterparts LTL [20] and CTL* [10] by trace quantification and are thereby able to express so-called hyperproperties [7], properties which relate multiple execution traces of a system. These are crucial for expressing information-flow properties capturing security and privacy requirements [6]. For example, generalized noninterference [19] is captured by the HyperLTL formula

$$\varphi_{\mathrm{GNI}} = \forall\pi.\ \forall\pi'.\ \exists\pi''.\ \mathbf{G}\left(\bigwedge_{\mathrm{p}\in L_{\mathrm{in}}\cup L_{\mathrm{out}}} \mathrm{p}_\pi \leftrightarrow \mathrm{p}_{\pi''}\right) \wedge \mathbf{G}\left(\bigwedge_{\mathrm{p}\in H_{\mathrm{in}}} \mathrm{p}_{\pi'} \leftrightarrow \mathrm{p}_{\pi''}\right)$$

expressing that for all traces π and π' there exists a trace π'' that agrees with the low-security inputs (propositions in L_{in}) and low-security outputs (propositions in L_{out}) of π and the high-security inputs (propositions in H_{in}) of π'. Intuitively, it is satisfied if every input-output behavior observable by a low-security user of a system is compatible with any sequence of high-security inputs, i.e., the low security input-output behavior does not leak information about the high-security inputs, which should indeed be unobservable (directly and indirectly) by a low-security user.

These expressive logics offer a uniform approach to the specification, analysis, and verification of hyperproperties with intuitive syntax and semantics, and a decidable [6], albeit TOWER-complete [22, 18], model-checking problem. The classical model-checking algorithm [6] is automata-based and even the case of $\forall^*\exists^*$-formulas like φ_{GNI} requires the complementation of ω-automata, a famously challenging construction to implement.

As an alternative, it is beneficial to draw upon the deep connections between logic and games. Consider a HyperLTL formula of the form $\forall\pi.\ \exists\pi'.\psi$ with quantifier-free ψ. It is straightforward to capture the semantics of HyperLTL by a model-checking game between two players called Verifier (trying to prove $\mathfrak{T} \models \varphi$) and Falsifier (trying to prove $\mathfrak{T} \not\models \varphi$). In the model-checking game, Falsifier picks a trace of $\mathfrak{T}$ to interpret π and then Verifier picks a trace of $\mathfrak{T}$ to interpret π'. Verifier wins if these two traces satisfy the formula ψ, otherwise Falsifier wins. The game can easily be shown sound (if Verifier wins, then $\mathfrak{T} \models \varphi$) and complete (if $\mathfrak{T} \models \varphi$ then Verifier wins), as winning strategies for Verifier are Skolem functions for π' and vice versa. But the game has one major drawback: both players have in general uncountably many possible moves as they are picking traces.

To obtain a game that can be handled algorithmically, Coenen et al. introduced the following variation of the model-checking game [8] (which we will call the alternating game): Instead of picking the traces of $\mathfrak{T}$ in one move, the players construct them alternatingly one vertex at a time. Together with a deterministic ω-automaton that is equivalent to ψ, this yields a finite game with ω-regular winning condition that is sound, but possibly incomplete ($\mathfrak{T} \models \varphi$ does not always imply that Verifier wins the alternating game).

The problem boils down to the informedness of Verifier: In the model-checking game, she has knowledge about the full trace assigned to π when picking π'. On the other hand, in the alternating game, Verifier only has access to the first vertex of the trace assigned to π when picking the first vertex of the trace assigned to π', which puts her at a disadvantage. That the alternating game can be incomplete is witnessed by the formula

$$\varphi_{\text{inc}} = \forall\pi.\ \exists\pi'.\ \mathtt{p}_{\pi'} \leftrightarrow \mathbf{F}\,\mathtt{p}_{\pi}$$

expressing that for every trace π in $\mathfrak{T}$ there is a trace π' in $\mathfrak{T}$ such that $\mathtt{p}$ holds at the first position of π' if and only if there is a $\mathtt{p}$ somewhere in π, i.e., the first letter of π' depends (possibly) on every letter of π. So, by not picking a $\mathtt{p}$ in the first round, Falsifier forces Verifier to make a prediction about whether Falsifier will ever pick a $\mathtt{p}$ or not in the future. However, Falsifier can easily contradict that prediction and thereby win, e.g., for a transition system $\mathfrak{T}_{\text{all}}$ having all traces over $\mathtt{p}$. Thus, we indeed have $\mathfrak{T}_{\text{all}} \models \varphi$, but Verifier does not win the alternating game.

However, a single bit of lookahead allows Verifier to win the alternating game induced by $\mathfrak{T}_{\text{all}}$ and φ_{inc}, i.e., the answer to the query "will the trace picked by Falsifier contain a $\mathtt{p}$". If Falsifier has to provide this information with his first move, then Verifier can make her first move accordingly. For correctness, we additionally have to adapt the rules of the alternating game so that Falsifier loses when contradicting his answer made during the first round, e.g., if he commits to there being no $\mathtt{p}$'s in his trace, but then picking one.

In general, it is not sufficient to ask a single query at the beginning of a play, but one needs to ask queries at every move of Falsifier. To see this, let us consider another example, this time with the formula

$$\varphi'_{\text{inc}} = \forall\pi.\ \exists\pi'.\ \mathbf{G}(\mathtt{p}_{\pi'} \leftrightarrow \mathbf{X}\,\mathtt{p}_{\pi}).$$

Here, Verifier has to always pick a $\mathtt{p}$ in her move if and only if Falsifier will pick a $\mathtt{p}$ in his next move (which Verifier does not have yet access to). Thus, Verifier wins if she gets the (truthful) answer to the query "does the next move by Falsifier contain a $\mathtt{p}$", but loses without the ability to query Falsifier. Note that in both cases, the queries are used to obtain a (binding) commitment about the future moves that Falsifier will make.

Coenen et al. [8] and Beutner and Finkbeiner [5] showed how to formalize this intuition using so-called prophecies [1]. A single prophecy is a language P of traces and is associated with a (Boolean) prophecy variable. Now, in addition to picking, vertex by vertex, a trace of

$\mathfrak{T}$, Falsifier also picks a truth value for the prophecy variable with the interpretation that a value of 1 corresponds to the suffix of the trace picked by him from now on being in P and that a value of of 0 corresponds to the suffix of the trace picked by him from now on not being in P. If the language P is ω-regular, then one can employ an ω-automaton to check whether the predictions made by Falsifier are actually truthful (and make him lose if they are not truthful). We call the resulting game the alternating game with prophecies. Intuitively, it constitutes a middle-ground between the naive game where both players pick traces and the alternating game (without prophecies). The former is sound and complete, but not finite-state while the latter is sound and finite-state, but not complete.

The alternating game with (finitely many) ω-regular prophecies is sound and finite-state and Beutner and Finkbeiner showed that for every $\forall^*\exists^*$-formula there is a finite and effectively computable set of ω-regular prophecies such that Verifier wins the alternating game with these prophecies if and only if $\mathfrak{T} \models \varphi$, i.e., there are always prophecies that make the alternating game-based approach to model-checking also complete. Furthermore, the resulting game with prophecies has a ω-regular winning condition, and can therefore be solved effectively. Note that this construction is still automata-based, as the prophecies are derived from a (deterministic) ω-automaton for the quantifier-free part of φ. Beutner and Finkbeiner implemented their prophecy-based model-checking algorithm and presented encouraging results on small instances, e.g., their prototype implementation is in some cases the first tool that can prove $\mathfrak{T} \not\models \varphi$, a result that was out of reach for existing model-checking tools [5]. These results shows the potential of the game-based approach to HyperLTL model-checking.

However, one question remains open: can the alternating game-based approach be extended to full HyperLTL, i.e., beyond a single quantifier alternation. There is precedent for such a game-based analysis of full HyperLTL: Recently, Winter and Zimmermann showed that the existence of (Turing) computable Skolem functions for existentially quantified variables can be characterized by a game [24]. For example, $\mathfrak{T}_{\text{all}} \models \varphi'_{\text{inc}}$ is witnessed by computable Skolem functions, as reading a prefix of length n of π allows to compute the prefix of length $n-1$ of π' so that for every π, π and the resulting π' satisfy $\mathbf{G}(\mathtt{p}_{\pi'} \leftrightarrow \mathbf{X}\,\mathtt{p}_{\pi})$. On the other hand, $\mathfrak{T}_{\text{all}} \models \varphi_{\text{inc}}$ is not witnessed by computable Skolem functions, as the choice of the first letter of π' depends, as explained above, on all letters of π'. Such a Skolem function is not computable by a Turing machine, as it is not continuous.

The game characterizing the existence of computable Skolem functions, say for a formula

$$\varphi = \forall\pi_0.\ \exists\pi_1.\ \ldots\forall\pi_{k-2}.\ \exists\pi_{k-1}.\ \psi$$

with quantifier-free ψ, is a multi-player game played between a player in charge of selecting, vertex by vertex, a trace for each universally quantified variable (i.e., he has the role that Falsifier has in the previous games), and a coalition of players, one for each existentially quantified variable (i.e., the coalition has the role that Verifier has in the previous games). Furthermore, the game must be of imperfect information in order to capture the semantics of HyperLTL, where the choice of π_i may only depend on the choice of the π_j with $j < i$. Thus, in the game, the player in charge of an existentially quantified π_i only has access to the choices made so far for the π_j for $j < i$. Furthermore, the game needs to incorporate a delay [16, 17, 12] between the moves of the different players to capture the fact that a choice by one of the existential players may depend on future moves by the universal player (see, e.g., the formula φ'_{inc} above). The main insight then is that a bounded delay is always sufficient, if there are computable Skolem functions at all (see, again, the difference between φ_{inc} (which has no computable Skolem functions over $\mathfrak{T}_{\text{all}}$) and φ'_{inc} (which has computable Skolem functions over $\mathfrak{T}_{\text{all}}$)).

Our Contribution. We present the first effective game-based characterization of model-checking for full HyperLTL (and even HyperQPTL, which allows to express all ω-regular hyperproperties [22, 13]), yielding a sound and complete imperfect information finite-state game with ω-regular winning condition. This result generalizes both the alternating game with prophecies from $\forall^*\exists^*$-formulas to formulas with arbitrary quantifier prefixes and the game of Winter and Zimmermann from characterizing the existence of computable Skolem functions witnessing $\mathfrak{T} \models \varphi$ to characterizing $\mathfrak{T} \models \varphi$.

However, since $\mathfrak{T} \models \varphi_{\text{inc}}$ holds, but does not have computable Skolem functions, the games of Winter and Zimmermann are *not* a special case of the games we construct here: Our games here are still multi-player games of imperfect information (to capture the semantics of quantification) and use prophecies (for completeness), but do not require delayed moves, as prophecies can be seen as a (restricted) form of infinite lookahead. And while the existence of computable Skolem functions is concerned with bounded lookahead, here we do indeed need infinite lookahead as witnessed by the formula φ_{inc}.

Our main result shows that there is again a finite and effectively computable set of ω-regular prophecies so that the coalition of players in charge of the existentially quantified variables has a winning strategy in the game with these prophecies if and only if $\mathfrak{T} \models \varphi$. One challenge to overcome here is a careful definition of the prophecies, so that they are not leaking any information about choices for variables π_j that a player in charge of π_i with $i < j$ must not have access to.

Our result can also be framed in terms of Skolem function implementable by letter-to-letter transducers with (ω-regular) lookahead: such a transducer computes a Skolem function for an existentially quantified variable π while reading values for the variables universally quantified before π while also being able to get a regular lookahead on the trace for those universally quantified variable (much like prophecies). The use of regular lookahead is well-studied in automata theory, see e.g., [9, 11, 3].

All proofs omitted due to space restrictions can be found in the full version [25].

2 Preliminaries

For convenience, technical terms and notations in the electronic version of this manuscript are hyper-linked to their definitions (cf. `https://ctan.org/pkg/knowledge`).

Hereafter, we denote the set of nonnegative integers by $\mathbb{N}$.

Traces, Transition Systems, and Automata. An alphabet is a nonempty finite set. The sets of finite and infinite words over an alphabet Σ are denoted by Σ^* and Σ^ω, respectively. The length of finite or infinite word w is denoted by $|w| \in \mathbb{N} \cup \{\infty\}$. For a word w of length at least n, we write $w[0, n)$ for the prefix of w of length n. Given n infinite words $w_0, \ldots, w_{n-1}$, let their *merge* (also known as zip), which is an infinite word over Σ^n, be defined as

$$\mathrm{mrg}(w_0, \ldots, w_{n-1}) = (w_0(0), \ldots, w_{n-1}(0))(w_0(1), \ldots, w_{n-1}(1))(w_0(2), \ldots, w_{n-1}(2)) \cdots .$$

We define $\mathrm{mrg}(w_0, \ldots, w_{n-1})$ for finite words $w_0, \ldots, w_{n-1}$ of the same length analogously.

Let AP be a nonempty finite set of atomic propositions. A *trace* over AP is an infinite word over the alphabet 2^{AP}. Given a subset $\mathrm{AP}' \subseteq \mathrm{AP}$, the AP'-projection of a trace $t(0)t(1)t(2)\cdots$ over AP is the trace $(t(0) \cap \mathrm{AP}')(t(1) \cap \mathrm{AP}')(t(2) \cap \mathrm{AP}') \cdots \in (2^{\mathrm{AP}'})^\omega$. Now, let AP and AP' be two disjoint sets, let t be a trace over AP, and let t' be a trace over AP'. Then, we define $t^\frown t'$ as the pointwise union of t and t', i.e., $t^\frown t'$ is the trace over $\mathrm{AP} \cup \mathrm{AP}'$ defined as $(t(0) \cup t'(0))(t(1) \cup t'(1))(t(2) \cup t'(2)) \cdots$.

A *transition system* $\mathfrak{T} = (V, E, V_I, \lambda)$ consists of a finite set V of vertices, a set $E \subseteq V \times V$ of (directed) edges, a nonempty set $V_I \subseteq V$ of initial vertices, and a labelling $\lambda\colon V \to 2^{\mathrm{AP}}$ of the vertices by sets of atomic propositions. We assume that every vertex has at least one outgoing edge. For $v \in V$, we denote by $\mathrm{S}(v)$ the set of its successors. A *path* ρ through $\mathfrak{T}$ is an infinite sequence $\rho = v_0v_1v_2\cdots$ of vertices with $v_0 \in V_I$ and $(v_n, v_{n+1}) \in E$ for every $n \geq 0$. The *trace of ρ* is defined as $\lambda(\rho) = \lambda(v_0)\lambda(v_1)\lambda(v_2)\cdots \in (2^{\mathrm{AP}})^\omega$. The *set of traces* of $\mathfrak{T}$ is $\mathrm{Tr}(\mathfrak{T}) = \{\lambda(\rho) \mid \rho \text{ is a path of } \mathfrak{T}\}$. For $V' \subseteq V$, we write $\mathfrak{T}_{V'}$ to denote the transition system (V, E, V', λ) obtained from $\mathfrak{T}$ by making V' the set of initial states, and use $\mathfrak{T}_v$ as shorthand for $\mathfrak{T}_{\{v\}}$ for $v \in V$.

A (deterministic) *parity automaton*[1] $\mathcal{A} = (Q, \Sigma, q_I, \delta, \Omega)$ consists of a finite set Q of states containing the initial state $q_I \in Q$, an alphabet Σ, a transition function $\delta\colon Q \times \Sigma \to Q$, and a coloring $\Omega\colon Q \to \mathbb{N}$ of its states by natural numbers. Let $w = w(0)w(1)w(2)\cdots \in \Sigma^\omega$. The run of $\mathcal{A}$ on w is the sequence $q_0q_1q_2\cdots$ with $q_0 = q_I$ and $q_{n+1} = \delta(q_n, w(n))$ for all $n \geq 0$. A run $q_0q_1q_2\cdots$ is (parity) accepting if the maximal color appearing infinitely often in the sequence $\Omega(q_0)\Omega(q_1)\Omega(q_2)\cdots$ is even. The language (parity) recognized by $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of infinite words over Σ such that the run of $\mathcal{A}$ on w is accepting.

▶ Remark 1. Deterministic parity automata accept exactly the ω-regular languages (see, e.g., [15] for definitions).

HyperQPTL, HyperLTL and QPTL. Let $\mathcal{V}$ be a countable set of *trace variables*. The formulas of *HyperQPTL* are given by the grammar

$$\varphi ::= \exists \pi.\ \varphi \mid \forall \pi.\ \varphi \mid \psi \qquad \psi ::= \tilde{\exists} \mathtt{q}.\ \psi \mid \tilde{\forall} \mathtt{q}.\ \psi \mid \psi \mid \mathtt{p}_\pi \mid \mathtt{q} \mid \neg \psi \mid \psi \vee \psi \mid \mathbf{X}\,\psi \mid \psi\,\mathbf{U}\,\psi$$

where $\mathtt{p}$ and $\mathtt{q}$ range over AP and where π ranges over $\mathcal{V}$. Here, we use a tilde to decorate propositional quantifiers to distinguish them from trace quantifiers.

Note that there are two types of atomic formulas, i.e., propositions labeled by trace variables on which they are evaluated ($\mathtt{p}_\pi$ with $\mathtt{p} \in \mathrm{AP}$ and $\pi \in \mathcal{V}$) and unlabeled propositions ($\mathtt{q} \in \mathrm{AP}$).[2] A formula is a sentence, if every occurrence of an atomic formula $\mathtt{p}_\pi$ is in the scope of a quantifier binding π (otherwise π is said to be a free trace variable) and every occurrence of an atomic formula $\mathtt{q}$ is in the scope of a quantifier binding $\mathtt{q}$ (otherwise $\mathtt{q}$ is said to be a free propositional variable). Note that the proposition $\mathtt{p}$ in an atomic formula $\mathtt{p}_\pi$ is not considered free. Finally, we use the usual syntactic sugar like conjunction ($\wedge$), implication ($\to$), equivalence ($\leftrightarrow$), eventually ($\mathbf{F}$), and always ($\mathbf{G}$).

A (trace) *variable assignment* is a partial mapping $\Pi\colon \mathcal{V} \to (2^{\mathrm{AP}})^\omega$. Given a variable $\pi \in \mathcal{V}$, and a trace t, we denote by $\Pi[\pi \mapsto t]$ the assignment that coincides with Π on all variables but π, which is mapped to t. Let $\mathtt{q} \in \mathrm{AP}$, let $t \in (2^{\mathrm{AP}})^\omega$ be a trace over AP, and let $t_\mathtt{q} \in (2^{\{\mathtt{q}\}})^\omega$ be a trace over $\{\mathtt{q}\}$. We define the trace $t[\mathtt{q} \mapsto t_\mathtt{q}] = t' {}^\frown t_\mathtt{q}$, where t' is the $(\mathrm{AP} \setminus \{\mathtt{q}\})$-projection of t: Intuitively, the occurrences of $\mathtt{q}$ in t are replaced according to $t_\mathtt{q}$. We lift this to sets T of traces by defining $T[\mathtt{q} \mapsto t_\mathtt{q}] = \{t[\mathtt{q} \mapsto t_\mathtt{q}] \mid t \in T\}$. Note that all traces in $T[\mathtt{q} \mapsto t_\mathtt{q}]$ have the same $\{\mathtt{q}\}$-projection, which is $t_\mathtt{q}$.

Now, for a trace assignment Π, a position $i \in \mathbb{N}$, and a nonempty set T of traces, i.e., we disregard the empty set of traces as model, we define

- $T, \Pi, i \models \mathtt{p}_\pi$ if $\mathtt{p} \in \Pi(\pi)(i)$,
- $T, \Pi, i \models \mathtt{q}$ if for all $t \in T$ we have $\mathtt{q} \in t(i)$,

[1] Note that we use parity acceptance, as we need deterministic automata for our proofs.
[2] We use different letters in these cases, but let us stress again that both $\mathtt{p}$ and $\mathtt{q}$ are propositions in AP.

- $T, \Pi, i \models \neg\psi$ if $T, \Pi, i \not\models \psi$,
- $T, \Pi, i \models \psi_1 \vee \psi_2$ if $T, \Pi, i \models \psi_1$ or $T, \Pi, i \models \psi_2$,
- $T, \Pi, i \models \mathbf{X}\,\psi$ if $T, \Pi, i+1 \models \psi$,
- $T, \Pi, i \models \psi_1 \mathbf{U}\, \psi_2$ if there is a $j \geq i$ such that $T, \Pi, j \models \psi_2$ and $T, \Pi, j' \models \psi_1$ for all $i \leq j' < j$,
- $T, \Pi, i \models \tilde{\exists}\mathsf{q}.\ \psi$ if there exists a trace $t_\mathsf{q} \in (2^{\{\mathsf{q}\}})^\omega$ such that $T[\mathsf{q} \mapsto t_\mathsf{q}], \Pi, i \models \psi$,
- $T, \Pi, i \models \tilde{\forall}\mathsf{q}.\ \psi$ if for all traces $t_\mathsf{q} \in (2^{\{\mathsf{q}\}})^\omega$ we have $T[\mathsf{q} \mapsto t_\mathsf{q}], \Pi, i \models \psi$,
- $T, \Pi, i \models \exists\pi.\ \varphi$ if there exists a trace $t \in T$ such that $T, \Pi[\pi \mapsto t], i \models \varphi$, and
- $T, \Pi, i \models \forall\pi.\ \varphi$ if for all traces $t \in T$ we have $T, \Pi[\pi \mapsto t], i \models \varphi$.

We say that an (again nonempty) set T of traces satisfies a sentence φ, written $T \models \varphi$, if $T, \Pi_\emptyset, 0 \models \varphi$ where $\Pi_\emptyset$ is the variable assignment with empty domain. We then also say that T is a model of φ. A transition system $\mathfrak{T}$ satisfies φ, written $\mathfrak{T} \models \varphi$, if $\mathrm{Tr}(\mathfrak{T}) \models \varphi$. Let ψ be a trace quantifier-free formula that has no free (unlabeled) propositions. A labeled proposition of the form p_π in ψ is evaluated on the traces assigned to π and an unlabeled proposition of the form q is evaluated on a trace "selected" by the quantifier binding q. Hence, $T, \Pi, i \models \psi$ is independent of T and we often write $\Pi \models \psi$ instead of $T, \Pi, 0 \models \psi$.

Some comments on the definition of HyperQPTL are due.

▶ Remark 2. We have, for technical reasons, defined HyperQPTL so that every formula has a prefix of trace quantifiers followed by a formula (possibly) containing propositional quantifiers, but no more trace quantifiers. On the other hand, Finkbeiner et al. [13] require formulas to be in prenex normal form, but allow to mix trace and propositional quantification in the quantifier prefix. We refrained from doing so, as one can duplicate the vertices of the transition system one is interested in, so that it has enough paths to simulate propositional quantification by trace quantification, and can adapt the formula correspondingly. This construction has a linear blowup.

▶ Remark 3. *HyperLTL* is the fragment of HyperQPTL sentences that do not use the propositional quantifiers $\tilde{\exists}$ and $\tilde{\forall}$ (and thus also not use unlabeled propositions).

We say that a HyperQPTL formula φ is a *QPTL* sentence if it has no trace quantifiers and no free propositional variables, i.e., all unlabeled propositional variables are in the scope of a propositional quantifier. But a QPTL sentence may have free trace variables, say $\pi_0, \ldots, \pi_{k-1}$ for some $k \geq 0$. Typically, QPTL is defined without trace variables labelling propositions and QPTL formulas define languages over the alphabet 2^{AP} [23]. For technical necessity, we allow such labels, which implies that our formulas define languages over the alphabet $(2^{\mathrm{AP}})^k$, where k is the number of trace variables occurring in the formula. More formally, a QPTL sentence with trace variables $\pi_0, \pi_1, \ldots, \pi_{k-1}$ defines the language

$$L(\varphi) = \{\mathrm{mrg}(t_0, \ldots, t_{k-1}) \mid t_0, \ldots, t_{k-1} \in (2^{\mathrm{AP}})^\omega : \Pi_\emptyset[\pi_0 \mapsto t_0, \ldots, \pi_{k-1} \mapsto t_{k-1}] \models \varphi\}.$$

Now, let $\varphi = Q_0\pi_0 Q_1\pi_1 \cdots Q_{k-1}\pi_{k-1}.\ \psi$ with $Q_i \in \{\exists, \forall\}$ and trace quantifier-free ψ be a HyperQPTL sentence and define $\varphi_i = Q_{i+1}\pi_{i+1} Q_{i+2}\pi_{i+2} \cdots Q_{k-1}\pi_{k-1}.\ \psi$ for $i \in \{0, 1, \ldots, k-1\}$ and $\varphi_{-1} = \varphi$. Note that $\varphi_{k-1} = \psi$ and that the free trace variables of each φ_i with $i \in \{-1, 0, \ldots, k-1\}$ are exactly $\pi_0, \ldots, \pi_i$. The following results follows by combining and adapting automata constructions for classical QPTL and HyperLTL [6, 14, 23].

▶ **Proposition 4.**

1. *Let $\mathfrak{T}$ be a transition system. For every $i \in \{-1, 0, \dots, k-1\}$ there is an (effectively constructible) parity automaton $\mathcal{A}_i^{\mathfrak{T}}$ such that*

$$L(\mathcal{A}_i^{\mathfrak{T}}) = \{\mathrm{mrg}(\Pi(\pi_0), \dots, \Pi(\pi_i)) \mid \Pi(\pi_j) \in \mathrm{Tr}(\mathfrak{T}) \text{ for } 0 \leq j \leq i \text{ and } \mathrm{Tr}(\mathfrak{T}), \Pi, 0 \models \varphi_i\}.$$

2. *A language over $(2^{\mathrm{AP}})^k$ is ω-regular if and only if it is of the form $L(\varphi)$ for a QPTL-sentence φ over some $\mathrm{AP}' \supseteq \mathrm{AP}$ with k free trace variables.*[3]

3 Gamed-based Model-Checking for HyperQPTL

In this section, we present our game-based characterization of $\mathfrak{T} \models \varphi$ for finite transition systems $\mathfrak{T}$ and HyperQPTL sentences φ. To this end, we introduce multi-player games with hierarchical information in Section 3.1 and then present the construction of our game in Section 3.2, while we introduce prophecies and prove their soundness in Section 3.3. Then, in Section 4, we show how to construct complete prophecies for the special case of safety formulas (to be defined formally there). Finally, in Section 5, we show how to construct complete prophecies for arbitrary formulas. This allows us to first explain how to generalize the prophecy definition from $\forall^*\exists^*$ to arbitrary quantifier prefixes and then, in a second step, move from safety to ω-regular languages.

3.1 Multi-player Graph Games with Hierarchical Information

We develop a game-based characterization of model-checking for HyperQPTL via (multi-player) graph games with hierarchical information, using the notations of Berwanger et al. [4]. First, we introduce the necessary definitions and then present our game in Section 3.2. The games considered by Berwanger et al. are concurrent games (i.e., the players make their moves simultaneously), while for our purpose turn-based games (i.e., the players make their moves one after the other) are sufficient. Turn-based games are simpler versions of concurrent games. To avoid cumbersome notation, we introduce a turn-based variant of these games.

Fix some finite set C of players forming a coalition playing against a distinguished agent called Nature (which is *not* in C). For each player $i \in C$ we fix a finite set B^i of observations. A game graph $G = (V, E, v_I, (\beta^i)_{i \in C})$ consists of a finite set $V = \biguplus_{i \in C} V_i \uplus V_{\mathrm{Nat}}$ of positions partitioned into sets controlled by some player resp. Nature, an edge relation $E \subseteq V \times V$ representing moves, an initial position $v_I \in V$, and a collection $(\beta^i)_{i \in C}$ of observation functions $\beta^i \colon V \to B^i$ that label, for each player, the positions with observations. We require that E has no dead-ends, i.e., for every $v \in V$ there is a $v' \in V$ with $(v, v') \in E$.

A game graph $(V, E, v_I, (\beta^i)_{i \in C})$ yields hierarchical information if there exists a total order $\preceq$ over C such that if $i \preceq j$ then for all $v, v' \in V$, $\beta^i(v) = \beta^i(v')$ implies $\beta^j(v) = \beta^j(v')$, i.e., if Player i cannot distinguish v and v', then neither can Player j for $i \preceq j$.

Intuitively, a play starts at position $v_I \in V$. At position v, the player that controls this position chooses a successor position v' such that $(v, v') \in E$. Now, each player $i \in C$ receives the observation $\beta^i(v')$ and the play continues from position v'. Thus, a play of G is an infinite sequence $v_0 v_1 v_2 \cdots$ of vertices such that $v_0 = v_I$ and for all $r \geq 0$ we have $(v_r, v_{r+1}) \in E$.

[3] For the sake of readability, in the following, we do not distinguish between AP and AP′, i.e., we assume that AP always contains enough propositions not used in our languages to quantify over.

A history is a prefix $v_0v_1\cdots v_r$ of a play. We denote the set of all histories by $\mathrm{Hist}(G)$ and extend $\beta^i\colon V\to B^i$ to plays and histories by defining $\beta^i(v_0v_1v_2\cdots)=\beta^i(v_1)\beta^i(v_2)\beta^i(v_3)\cdots$. Note that the observation of the initial position is discarded for technical reasons [4]. We say two histories h and h' are indistinguishable to Player $i\in C$, denoted by $h\sim_i h'$, if $\beta^i(h)=\beta^i(h')$.

A strategy for Player $i\in C$ is a mapping $s^i\colon V^*\to V$ that satisfies $s^i(h)=s^i(h')$ for all histories h,h' with $h\sim_i h'$ (i.e., the move selected by the strategy only depends on the observations of the history). A play $v_0v_1v_2\cdots$ is consistent with s^i if for every $r\geq 0$, we have $v_{r+1}=s^i(v_0v_1\cdots v_r)$. A play is consistent with a collection of strategies $(s^i)_{i\in C}$ if it is consistent with each s^i. The set of possible outcomes of a collection of strategies is the set of all plays that are consistent with it. As usual, a strategy is finite-state, if it is implemented by some Moore machine.

Lastly, a game $\mathcal{G}$ consists of a game graph G and a winning condition $W\subseteq V^\omega$, where V is the set of positions of G. A play is winning if it is in W and a collection of strategies is winning if all its outcomes are winning.

▶ **Proposition 5** ([21, 4]).

1. *It is decidable, given a game with hierarchical information with ω-regular winning condition, whether it has a winning collection of strategies.*
2. *A game with hierarchical information with ω-regular winning condition has a winning collection of strategies if and only if it has a winning collection of finite-state strategies.*

3.2 The Model-checking Game

For the remainder of this paper, we fix a HyperQPTL sentence φ and a transition system $\mathfrak{T}$. We assume (w.l.o.g.)[4]

$$\varphi=\forall\pi_0\exists\pi_1\cdots\forall\pi_{k-2}\exists\pi_{k-1}.\ \psi$$

such that ψ is trace quantifier-free, define

$$\varphi_i=Q_{i+1}\pi_{i+1}Q_{i+2}\pi_{i+2}\cdots\forall\pi_{k-2}\exists\pi_{k-1}.\ \psi$$

for $i\in\{-1,0,\ldots,k-1\}$ and use the automata $\mathcal{A}_i^{\mathfrak{T}}$ constructed in Proposition 4 satisfying

$$L(\mathcal{A}_i^{\mathfrak{T}})=\{\mathrm{mrg}(\Pi(\pi_0),\ldots,\Pi(\pi_i))\mid\Pi(\pi_j)\in\mathrm{Tr}(\mathfrak{T})\text{ for all }0\leq j\leq i\text{ and }\mathrm{Tr}(\mathfrak{T}),\Pi,0\models\varphi_i\}.$$

We use the notation $(\mathcal{A}_i^{\mathfrak{T}})_q$ to denote the parity automaton obtained from $\mathcal{A}_i^{\mathfrak{T}}$ by making its state q the initial state. Finally, let $\mathcal{A}_{k-1}^{\mathfrak{T}}=(Q,\Sigma,q_I,\delta,\Omega)$. Note that $\mathcal{A}_{k-1}^{\mathfrak{T}}$ accepts the trace assignments coming from paths trough $\mathfrak{T}$ that satisfy ψ. We say that $\mathcal{A}_{k-1}^{\mathfrak{T}}$ checks ψ.

We define a multi-player game $\mathcal{G}(\mathfrak{T},\varphi)$ with hierarchical information induced by the transition system $\mathfrak{T}$ and the HyperQPTL sentence φ. This game is played between *Falsifier* (who takes on the role of Nature, cf. Section 3.1), who is in charge of providing traces (from paths trough the transition system) for the universally quantified variables, and a coalition of *Verifier-players* $\{1,3,\ldots,k-1\}$ (*Verifier i* for short), who are in charge of providing traces (also from paths trough the transition system) for the existentially quantified variables. The goal of Falsifier is to prove $\mathfrak{T}\not\models\varphi$ and the goal of the coalition of Verifier-players is to prove

[4] The following reasoning can easily be extended to general sentences with arbitrary quantifier prefixes, albeit at the cost of more complex notation. We substantiate this claim in Remark 20 on Page 16.

$\mathfrak{T} \models \varphi$. Therefore, the k traces built during a play are read synchronously by the parity automaton $\mathcal{A}_{k-1}^{\mathfrak{T}}$ accepting the trace assignments that satisfy ψ (recall that ψ is the trace quantifier-free part of φ).

The game has two phases, an initialization phase where initial vertices for all paths through the transition system are picked, and a second phase (of infinite duration) where the paths (which induce the trace assignment) are built. Formally, in the initialization phase, a position of the game is of the form $((v_0, \ldots, v_{i-1}, \underbrace{\bullet, \ldots, \bullet}_{k-i \text{ times}}), q_I, i)$ where $v_0, \ldots, v_{i-1} \in V_I$ are initial vertices in the transition system $\mathfrak{T}$, $\bullet$ is a fresh (placeholder) symbol, $q_I \in Q$ is the initial state of $\mathcal{A}_{k-1}^{\mathfrak{T}}$, and $i \in \{0, 1, \cdots, k-1\}$. In the second phase, a position of the game is of the form $((v_0, \ldots, v_{k-1}), q, i)$ where $v_0, v_1, \ldots, v_{k-1} \in V$, $q \in Q$, and $i \in \{0, 1, \ldots, k-1\}$. A vertex whose last component is an odd i is controlled by Verifier i and a vertex whose last component is even is controlled by Falsifier.

The edges (also called moves) of the game graph are defined as follows, where the first two items are the moves in the initialization phase:

- $\big(((v_0, \ldots, v_{i-1}, \bullet, \ldots, \bullet), q_I, i), ((v_0, \ldots, v_{i-1}, v_i, \bullet, \ldots, \bullet), q_I, i+1)\big)$ for all $v_i \in V_I$ and all $i \in \{0, 1, \ldots, k-2\}$: An initial vertex in $\mathfrak{T}$ for the i-th path is picked.
- $\big(((v_0, \ldots, v_{k-2}, \bullet), q_I, k-1), ((v_0, \ldots, v_{k-2}, v_{k-1}), q, 0)\big)$ for all $v_{k-1} \in V_I$ where $q = \delta(q_I, \mathrm{mrg}(\lambda(v_0), \ldots, \lambda(v_{k-1})))$: An initial vertex in $\mathfrak{T}$ for the last path is picked. With this move, the initialization phase is over and the state of $\mathcal{A}_{k-1}^{\mathfrak{T}}$ checking ψ is updated for the first time.
- $\big(((v'_0, \ldots, v'_{i-1}, v_i, v_{i+1}, \ldots, v_{k-1}), q, i), ((v'_0, \ldots, v'_{i-1}, v'_i, v_{i+1}, \ldots, v_{k-1}), q, i+1)\big)$ for all $(v_i, v'_i) \in E$ and all $i \in \{0, 1, \ldots, k-2\}$: The i-th path is updated by moving to a successor vertex in $\mathfrak{T}$.
- $\big(((v'_0, \ldots, v'_{k-2}, v_{k-1}), q, k-1), ((v'_0, \ldots, v'_{k-2}, v'_{k-1}), q', 0)\big)$ for all $(v_{k-1}, v'_{k-1}) \in E$ where $q' = \delta(q, \mathrm{mrg}(\lambda(v'_0), \lambda(v'_1), \ldots, \lambda(v'_{k-1})))$: The last path is updated by moving to a successor vertex in $\mathfrak{T}$. Simultaneously, the state of $\mathcal{A}_{k-1}^{\mathfrak{T}}$ checking ψ is updated.

The initial position is $((\bullet, \ldots, \bullet), q_I, 0)$. We use a parity winning condition. The color of all positions of the initialization phase is 0 (the color is of no consequence as these vertices are seen only once during the course of a play). The color of positions of the form $((v_0, \ldots, v_{k-1}), q, i)$ is the color that q has in $\mathcal{A}_{k-1}^{\mathfrak{T}}$, i.e., a play is winning for the Verifier-players if and only if the trace assignment picked by them and Falsifier satisfies ψ.

The game described must be a game of hierarchical information to capture the fact that the Skolem function for an existentially quantified π_i depends only on the universally quantified variables π_j with $j \in \{0, 2, \ldots, i-1\}$. To capture that, we define an equivalence relation $\equiv_i$ between positions of the game for $i \in \{1, 3, \ldots, k-1\}$ which ensures that for Verifier i, two positions are indistinguishable if they coincide on their first $i+1$ components and additionally belong to the same player. Formally, regarding positions from the initialization phase, let $((v_0, \ldots, v_{m-1}, \bullet, \ldots, \bullet), q_I, m)$ and $((v'_0, \ldots, v'_{n-1}, \bullet, \ldots, \bullet), q_I, n)$ be $\equiv_i$-equivalent if $v_j = v'_j$ for all $j \leq i$ and $m = n$. Regarding all other positions, let $((v_0, \ldots, v_{k-1}), p, m)$ and $((v'_0, \ldots, v'_{k-1}), q, n)$ be $\equiv_i$-equivalent if $v_j = v'_j$ for all $j \leq i$ and $m = n$. Now, we define the observation functions: For Verifier i, it maps positions to their $\equiv_i$-equivalence classes.

3.3 The Model-Checking Game with Prophecies

The game $\mathcal{G}(\mathfrak{T}, \varphi)$ as introduced in Section 3.2 does not capture $\mathfrak{T} \models \varphi$. While it is sound (see Lemma 8 for the empty set of prophecies), i.e., the coalition of Verifier-players having a winning collection of strategies for $\mathcal{G}(\mathfrak{T}, \varphi)$ implies that $\mathfrak{T} \models \varphi$, but the converse is not necessarily true. This is witnessed, e.g., by a transition system $\mathfrak{T}$ with $\mathrm{Tr}(\mathfrak{T}) = (2^{\{\mathtt{p}\}})^\omega$ and the sentence $\forall\pi.\ \exists\pi'.\ \mathtt{p}_{\pi'} \leftrightarrow \mathbf{F}\,\mathtt{p}_{\pi}$. As explained in the introduction, the Verifier-player does not have a strategy to select, step-by-step, a trace t' for π' while given, again step-by-step, a trace t for π, as the choice of the first letter of t' depends on all positions of t. Hence, the coalition does not win $\mathcal{G}(\mathfrak{T}, \varphi)$, even though $\mathfrak{T} \models \varphi$.

In the following, we show how *prophecies*, binding commitments about future moves by Falsifier, make the game-based approach to model-checking complete. In our example, Falsifier has to make, with his first move, a commitment about whether he will ever play a $\mathtt{p}$ or not. This allows the Verifier-player to pick the "right" first letter of t' and thereby win the game, if Falsifier honors the commitment. If not, the rules of the game make him loose. To add prophecies to $\mathcal{G}(\mathfrak{T}, \varphi)$, we do not change the rules of the game, but instead modify the transition system the game is played on (to allow Falsifier to select truth values for the *prophecy variables*[5], which is the mechanism he uses to make the commitments) and modify the formula (to ensure that Falsifier loses if he breaks a commitment). Hence, given $\mathfrak{T}$ and φ, we construct $\mathfrak{T}^{\mathcal{P}}$ and $\varphi^{\mathcal{P},\Xi}$ such that the following two properties are satisfied:

- Soundness: If the coalition of Verifier-players has a winning collection of strategies for $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$, then $\mathfrak{T} \models \varphi$.
- Completeness: If $\mathfrak{T} \models \varphi$, then the coalition of Verifier-players has a winning collection of strategies for $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$.

The challenge here is to construct the "right" prophecies that allow us to prove completeness, as soundness is independent of the prophecies chosen.

Recall that $\pi_0, \pi_2, \ldots, \pi_{k-2}$ resp. $\pi_1, \pi_3, \ldots, \pi_{k-1}$ are the universally resp. existentially quantified variables in our fixed formula φ. We frequently need to refer to their indices. Hence, we define $I_\forall = \{0, 2, \ldots, k-2\}$ and $I_\exists = \{1, 3, \ldots, k-1\}$.

We begin by defining the transition system $\mathfrak{T}^{\mathcal{P}}$ from the fixed transition system $\mathfrak{T}$, in which paths additionally determine truth values for prophecy variables.

▶ **Definition 6** (*System manipulation*). *Let $\mathcal{P} = (\mathrm{P}_i)_{i \in I_\forall}$ be a collection of sets of atomic propositions such that* AP, *the* P_i, *and* $\{\mathtt{m}_i \mid i \in I_\forall\}$ *are all pairwise disjoint, where the* $\mathtt{m}_i$ *are propositions used to mark copies of the transition system* $\mathfrak{T} = (V, E, V_I, \lambda)$.

For $i \in I_\forall$, *we define* $\mathfrak{T}^{\mathrm{P}_i} = (V^{\mathrm{P}_i}, E^{\mathrm{P}_i}, V_I^{\mathrm{P}_i}, \lambda^{\mathrm{P}_i})$ *over* $\mathrm{AP} \uplus \mathrm{P}_i \uplus \{\mathtt{m}_i\}$ *where* $V^{\mathrm{P}_i} = V \times 2^{\mathrm{P}_i} \times \{i\}$, $E^{\mathrm{P}_i} = \{((s, A, i), (s', A', i)) \mid (s, s') \in E$ *and* $A, A' \in 2^{\mathrm{P}_i}\}$, $V_I^{\mathrm{P}_i} = V_I \times 2^{\mathrm{P}_i} \times \{i\}$ *and* $\lambda^{\mathrm{P}_i}(s, A, i) = \lambda(s) \cup A \cup \{\mathtt{m}_i\}$.

Furthermore, we define $\mathfrak{T}^{\mathcal{P}} = (V^{\mathcal{P}}, E^{\mathcal{P}}, V_I^{\mathcal{P}}, \lambda^{\mathcal{P}})$ *as the disjoint union of the* $\mathfrak{T}^{\mathrm{P}_i}$, *where a vertex of* $\mathfrak{T}^{\mathcal{P}}$ *is in* $V_I^{\mathcal{P}}$ *if and only if it is in some* $V_I^{\mathrm{P}_i}$. *Consequently, we have* $\mathrm{Tr}(\mathfrak{T}^{\mathcal{P}}) = \bigcup_{i \in I_\forall} \{t^\frown t'^\frown \{\mathtt{m}_i\}^\omega \mid t \in \mathrm{Tr}(\mathfrak{T})$ *and* $t' \in (2^{\mathrm{P}_i})^\omega\}$.

An effect of this definition is that all paths trough the manipulated system select truth values for the prophecy variables with each move. Furthermore, each such path is marked by a proposition of the form $\mathtt{m}_i$ indicating for which i the prophecy variables are selected. This will later be used to ensure that Falsifier selects the correct prophecies (cf. Definition 7) for each path he picks for a universally quantified variable. For the Verifier-players, this additional information is simply ignored, as can be seen from the manipulated formula (cf. Definition 7) we introduce now.

We define the sentence $\varphi^{\mathcal{P},\Xi}$ which ensures that Falsifier loses, if he breaks his commitments made via prophecy variables: for every prophecy variable, we have an associated *prophecy*, a language $\mathfrak{P} \subseteq ((2^{\mathrm{AP}})^i)^\omega$ for some i. Note that Falsifier can only make commitments about

[5] Note that prophecy variables are Boolean variables that are set by Falsifier during each move of a play and should not be confused with trace variables or with propositional variables.

his own moves, as the Verifier-players could otherwise falsify the commitments (made by Falsifier) about their moves: Prophecies can only refer to traces for universally quantified variables. Also, we will have prophecies for each universally quantified variable.

▶ **Definition 7** (*Property manipulation*). *Let $\Xi = (\Xi_i)_{i \in I_\forall}$ be a family $\Xi_i = \{\xi_{i,1}, \dots, \xi_{i,n_i}\}$ of sets of QPTL sentences (over* AP*) such that each $\xi \in \Xi_i$ uses only trace variables π_j with even $j \leq i$. Let $\mathcal{P} = (\mathrm{P}_i)_{i \in I_\forall}$ satisfy the disjointness requirements of Definition 6 and $\mathrm{P}_i = \{\mathtt{p}_{i,1}, \dots, \mathtt{p}_{i,n_i}\}$, i.e., we have $|\Xi_i| = |\mathrm{P}_i|$. We define the HyperQPTL sentence $\varphi^{\mathcal{P},\Xi}$ as*

$$\forall \pi_0 \exists \pi_1 \cdots \forall \pi_{k-2} \exists \pi_{k-1}. \left[\bigwedge\nolimits_{i \in I_\forall} (\mathtt{m}_i)_{\pi_i} \wedge \mathbf{G} \left(\bigwedge\nolimits_{\ell=1}^{n_i} ((\mathtt{p}_{i,\ell})_{\pi_i} \leftrightarrow \xi_{i,\ell}) \right) \right] \rightarrow \psi.$$

We denote the trace quantifier-free part of $\varphi^{\mathcal{P},\Xi}$ by $\psi^{\mathcal{P},\Xi}$.

Note that $\varphi^{\mathcal{P},\Xi}$ has the same quantifier prefix as φ and that restricting each $\xi_{i,\ell}$ to trace variables π_j with even $j \leq i$ ensures that the prophecies only refer to trace variables under the control of Falsifier. Also note that the truth values of prophecy variables on trace variables under the control of the Verifier-players are not used in the formula. Finally, Falsifier has to pick the i-th path in $\mathfrak{T}^{\mathrm{P}_i}$ (and thus select valuations for the prophecy variables in P_i), otherwise he loses immediately.

Our construction is sound, independently of the choice of prophecies.

▶ **Lemma 8.** *Let Ξ and $\mathcal{P}$ satisfy the requirements of Definition 7. If the coalition of Verifier-players has a winning collection of strategies in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$, then $\mathfrak{T} \models \varphi$.*

4 Complete Prophecies for Safety Properties

We show that there are sets Ξ and $\mathcal{P}$ such that if $\mathfrak{T} \models \varphi$, then the coalition of Verifier-players wins $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$. We first consider the case where ψ is a safety property, i.e., the automaton $\mathcal{A}^{\mathfrak{T}}_{k-1}$, which checks ψ, is a deterministic safety automaton. A *safety automaton* is a parity automaton using only two colors, an even one and an odd one, and moreover, all its states are even-colored except for an odd-colored sink state. In other words, all runs avoiding the single unsafe state are accepting. We start with safety as this allows us to work with "simpler" prophecies while presenting all underlying concepts needed for arbitrary quantifier prefixes. The idea for safety is that a prophecy should indicate which successor vertices are safe for the Verifier-players to move to, i.e., from which successor vertices it is possible to successfully continue the play without immediately losing. In the general case, we have to additionally handle a more complex acceptance condition. This is shown in Section 5.

▶ **Definition 9** (Prophecy construction for safety). *For each $i \in I_\forall$, each vector $\bar{v} = (v_0, v_1, \dots, v_{i+1})$ of vertices of $\mathfrak{T}$, and each state q of $\mathcal{A}^{\mathfrak{T}}_{i+1}$ we define*

$$(\mathfrak{S}_i)^{\bar{v}}_q = \{\mathrm{mrg}(t_0, t_2, \dots, t_i) \mid t_0 \in \mathrm{Tr}(\mathfrak{T}_{v_0}), t_2 \in \mathrm{Tr}(\mathfrak{T}_{v_2}), \dots, t_i \in \mathrm{Tr}(\mathfrak{T}_{v_i}), \text{ and there are } t_1 \in \mathrm{Tr}(\mathfrak{T}_{v_1}), t_3 \in \mathrm{Tr}(\mathfrak{T}_{v_3}), \dots, t_{i+1} \in \mathrm{Tr}(\mathfrak{T}_{v_{i+1}}) \text{ s.t. } \mathrm{mrg}(t_0, \dots, t_{i+1}) \in L((\mathcal{A}^{\mathfrak{T}}_{i+1})_q)\}.$$

Note that the t_i for $i \in I_\exists$ may depend on all t_j with $j \in I_\forall$. Our game definition ensures that the choice for an existential variable π_i only depends on the choices for π_j with $j < i$.

Also note that each prophecy is an ω-regular language, as ω-regular languages are closed under projection, the analogue of existential quantification. So, due to Proposition 4, there are QPTL-sentences expressing the prophecies allowing us to verify them in the formula $\varphi^{\mathcal{P},\Xi}$.

Next, we define the sets $\Xi = (\Xi_i)_{i \in I_\forall}$ and $\mathcal{P} = (\mathrm{P}_i)_{i \in I_\forall}$ of QPTL formulas expressing the prophecies defined above and the corresponding prophecy variables.

▶ **Definition 10.** *Let Ξ_i be the set of QPTL formulas that contains sentences $(\xi_i)_q^{\bar{v}}$ for each state q of $\mathcal{A}_{i+1}^{\mathfrak{T}}$ and each vector $\bar{v}$ of vertices expressing the prophecy $(\mathfrak{S}_i)_q^{\bar{v}}$ using only trace variables π_j with even $j \leq i$. Let P_i be the set that contains the prophecy variable $(\mathtt{p}_i)_q^{\bar{v}}$ for each state q of $\mathcal{A}_{i+1}^{\mathfrak{T}}$ and each vector $\bar{v}$ of vertices. The prophecy variable $(\mathtt{p}_i)_q^{\bar{v}}$ corresponds to $(\xi_i)_q^{\bar{v}}$.*

Recall that ψ is the maximal trace quantifier-free subformula of φ. Our main technical lemma shows that the prophecies defined above are indeed complete.

▶ **Lemma 11.** *Let $\mathfrak{T} \models \varphi$ such that ψ is a safety property. Then the coalition of Verifier-players has a winning collection of strategies in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ with Ξ and $\mathcal{P}$ as in Definition 10.*

Before we turn to the proof of Lemma 11, we introduce some notation about the game $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ and define the strategies we will prove winning.

Firstly, recall that $\mathfrak{T}^{\mathcal{P}}$ is the union $\biguplus_{i \in I_\forall} \mathfrak{T}^{\mathrm{P}_i}$. Thus, a vertex v of $\mathfrak{T}^{\mathcal{P}}$ is of the form $(s, A, i) \in V \times 2^{\mathrm{P}_i} \times \{i\}$ for some P_i, where V is the set of vertices of $\mathfrak{T}$, and the label $\lambda^{\mathrm{P}_i}(v)$ of v is $\lambda(s) \cup A \cup \{\mathtt{m}_i\}$, i.e., the union of its original (meaning AP-based) label $\lambda(s)$ in $\mathfrak{T}$, its associated prophecy variables A, and its marker $\mathtt{m}_i$ indicating that v belongs to $\mathfrak{T}^{\mathrm{P}_i}$. We need to access these bits of information. Hence, we define $\lambda_{\mathrm{AP}}(v) = \lambda(s)$, $\mathrm{Prophecies}(v) = A$, and $\mathrm{mark}(v) = \mathtt{m}_i$. Given a path $\rho = v_0 v_1 v_2 \cdots$ through $\mathfrak{T}^{\mathcal{P}}$, let $\lambda_{\mathrm{AP}}(\rho) = \lambda_{\mathrm{AP}}(v_0)\lambda_{\mathrm{AP}}(v_1)\lambda_{\mathrm{AP}}(v_2)\cdots$. Moreover, for $v = (s, A, i)$ in $\mathfrak{T}^{\mathrm{P}_i}$, we are often interested in reasoning about s in $\mathfrak{T}$. To simplify our notation, we will also write v for the (unique) vertex s in $\mathfrak{T}$ induced by v.

Secondly, recall that $\psi^{\mathcal{P},\Xi}$ is the trace quantifier-free part of $\varphi^{\mathcal{P},\Xi}$. Let $\mathcal{B}$ denote the parity automaton $(Q_{\mathcal{B}}, \Sigma, q_I^{\mathcal{B}}, \delta_{\mathcal{B}}, \Omega_{\mathcal{B}})$ that accepts the trace assignments that satisfy $\psi^{\mathcal{P},\Xi}$.

Lastly, recall that a position of $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ is of the form $((v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}), q, i)$ where $v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}$ are vertices in the transition system $\mathfrak{T}^{\mathcal{P}}$, $q \in Q_{\mathcal{B}}$ is a state of the parity automaton $\mathcal{B}$ checking $\psi^{\mathcal{P},\Xi}$, and $i \in \{0, 1, \ldots, k-1\}$. Technically, in the initialization phase, a position is of the form $((v_0, \ldots, v_{i-1}, \underbrace{\bullet, \ldots, \bullet}_{k-i \text{ times}}), q_I, i)$ where $v_0, \ldots, v_{i-1}$ are vertices and $\bullet$ is a placeholder. For simplicity, we always refer to a position with $((v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}), q, i)$, even though $\bullet$ might be present. Also, for convenience, we define the successors $\mathrm{S}(\bullet)$ of the placeholder symbol $\bullet$ to be V_I, the set of initial vertices of $\mathfrak{T}$.

We say a play in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ is in *Round (r, i)* for $r \geq 0$ and $i \in \{0, 1, \ldots, k-1\}$ if the play is in a position of the form $((v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}), q, i)$ for the $(r+1)$-th time. Also, we say we are in *Round r* if the play is in Round (r, i) for some i.

Given a play α in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$, for $i \in \{0, 1, \ldots, k-1\}$, let ρ_i^α denote the path through $\mathfrak{T}^{\mathcal{P}}$ that is induced by α when considering the moves made in Round (r, i) for $r \geq 0$: Formally, if in Round (r, i) the move from position $p = ((v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}), q, i)$ to $p' = ((v_0', \ldots, v_{i-1}', v_i', \ldots, v_{k-1}), q', (i+1) \mod k)$ is made, then $\rho_i^\alpha(r) = v_i'$. Note that $q' = q$, unless $i = k-1$, then $q' = \delta_{\mathcal{B}}(q, \mathrm{mrg}(\lambda^{\mathcal{P}}(v_0'), \ldots, \lambda^{\mathcal{P}}(v_{k-1}')))$. Furthermore, note that the move (p, p') is uniquely identified by p and v_i'. So in the future, we simply write the move from v_i to v_i' when p is clear from the context. We drop the index α from ρ_i^α and write ρ_i when α is clear from the context.

We continue by defining a collection of strategies for the Verifier-players and then show that $\mathfrak{T} \models \varphi$ implies that this collection is winning in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$. Recall that $\varphi^{\mathcal{P},\Xi}$ is $\forall \pi_0 \exists \pi_1 \cdots \forall \pi_{k-2} \exists \pi_{k-1}.\ \psi^{\mathcal{P},\Xi}$. Assume the play is in Round (r, i) for $r \geq 0$ and $i \in I_\exists$. Thus, it is in a position of the form

$$((v_0', \ldots, v_{i-1}', v_i, \ldots, v_{k-1}), q, i)$$

and Verifier i has to move. We review some information Verifier i can use to base her choice on. In Round r, Verifier i has access to the first r letters of the traces $\lambda_{\mathrm{AP}}(\rho_0), \lambda_{\mathrm{AP}}(\rho_1), \ldots, \lambda_{\mathrm{AP}}(\rho_i)$ induced by the play.[6] Let q_i be the state that $\mathcal{A}_i^{\mathfrak{T}}$ has reached on $\mathrm{mrg}(\lambda_{\mathrm{AP}}(\rho_0)[0,r), \lambda_{\mathrm{AP}}(\rho_1)[0,r), \ldots, \lambda_{\mathrm{AP}}(\rho_i)[0,r))$ for $i \in I_\exists$.

▶ **Definition 12** (Strategy definition for safety)**.** *Let*

$$M_i(v_0', \ldots, v_{i-1}', v_i, q_i) = \{v_i' \in \mathrm{S}(v_i) \mid (\mathfrak{p}_{i-1})_{q_i}^{\bar{v}} \in \mathrm{Prophecies}(v_{i-1}')\}$$

*where $\bar{v} = (v_0', \ldots, v_i')$. If $M_i(v_0', \ldots, v_{i-1}', v_i, q_i)$ is nonempty, then Verifier i can move to any v_i' in the set (the "*NONEMPTY*-case"). If it is empty, then Verifier i can move to any $v_i' \in \mathrm{S}(v_i)$ (the "*EMPTY*-case").*

Proof sketch of Lemma 11. We show that the just constructed strategies form a winning collection for the Verifier-players. Here, we reason about a play under the assumption that all prophecies are set correctly (i.e., the premise of $\psi^{\mathcal{P},\Xi}$ is true) as otherwise, the play is trivially won by the Verifier-players. Under this assumption, and the basis that $\mathfrak{T} \models \varphi$, we prove that the Verifier-players have always used the NONEMPTY-case to pick their moves. This implies that the play is winning for them: We have to argue that the conclusion of $\psi^{\mathcal{P},\Xi}$ (which is ψ) is true. Recall that $\mathcal{A}_{k-1}^{\mathfrak{T}}$ checks ψ. Let $t_0, \ldots, t_{k-1}$ be the traces induced by the play. We show that $\mathcal{A}_{k-1}^{\mathfrak{T}}$ accepts $\mathrm{mrg}(t_0, \ldots, t_{k-1})$. In Round $(r, k-1)$, $\mathcal{A}_{k-1}^{\mathfrak{T}}$ has reached a state q_{k-1} on the prefix of length r of $\mathrm{mrg}(t_0, \ldots, t_{k-1})$, the prophecy $(\mathfrak{S}_{k-1})_{q_{k-1}}^{\bar{v}}$ for the correct $\bar{v}$ (cf. Definition 12) holds, which implies that $L((\mathcal{A}_{k-1}^{\mathfrak{T}})_{q_{k-1}}) \neq \emptyset$. Hence, q_{k-1} is a safe state, and not the single (unsafe) sink state. Thus, an induction shows that the run of $\mathcal{A}_{k-1}^{\mathfrak{T}}$ on $\mathrm{mrg}(t_0, \ldots, t_{k-1})$ is accepting. To establish that the Verifier-players have always used the NONEMPTY-case, we proceed by induction on the rounds. We show that, in Round r, Verifier i for each $i \in I_\exists$ has so far produced a trace prefix of length r such that it is possible to successfully continue the prefix in a way that allows the Verifier-players with higher indices to also continue successfully. Successful simply meaning the traces can be continued in a way such that $\mathcal{A}_{k-1}^{\mathfrak{T}}$ accepts. This implies that they can also use the NONEMPTY-case in the next round. ◀

Combining Lemma 8 and Lemma 11, we obtain our main result of this section for formulas φ where the maximal trace quantifier-free subformula ψ is a safety property.

▶ **Theorem 13.** *Let Ξ, $\mathcal{P}$ be as in Definition 10, and let ψ be a safety property. The coalition of Verifier-players has a winning collection of strategies for $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ if and only if $\mathfrak{T} \models \varphi$.*

5 Complete Prophecies for ω-Regular Properties

In this section, we show how to construct complete prophecies for trace quantifier-free ψ beyond the safety fragment. In the safety case, prophecies indicate moves which prevent the Verifier-players from losing. In the case of safety, not losing equals winning. However for general ω-regular objectives, this is not enough. It is also necessary to (again and again) make progress towards satisfying the acceptance condition.

In this section, we assume that the automata $\mathcal{A}_i^{\mathfrak{T}}$ are *Rabin automata*. The class of languages recognized by (deterministic respectively nondeterministic) Rabin automata is the class of ω-regular languages. A (deterministic) Rabin automaton is a (deterministic)

[6] For $j < i$, she actually has access to $r + 1$ letters, but our construction is independent of the last ones.

ω-automaton whose acceptance condition is given as a set $\{(B_1, F_1), \dots, (B_m, F_m)\}$ of pairs of sets B_i, F_i of states. A run is accepting if there exists an i such that the run visits states in B_i only finitely many times and states in F_i infinitely many times.

5.1 One-Pair Rabin Automata

First, we present our construction for one-pair Rabin automata, then show how to generalize to Rabin automata with an arbitrary number of pairs in Section 5.2. Here, we assume that for *each* $i \in I_\exists$, the automaton $\mathcal{A}_i^{\mathfrak{T}}$ is a one-pair Rabin automaton, not only $\mathcal{A}_k^{\mathfrak{T}}$. We always denote the single pair as (B, F), it will be clear from the context to which automaton it belongs. Intuitively, making progress towards winning in a one-pair Rabin automaton means avoiding states in B and visiting states in F.

Let κ be an infinite run of such an automaton. We define $\mathrm{last}_B(\kappa)$ as $\perp$ if κ never visits B, as i if $\kappa(i)$ is the last visit of κ in B, and as ∞ if κ visits B infinitely many times (meaning there is no last visit). Furthermore, we define $\mathrm{first}_F(\kappa)$ as i if $\kappa(i)$ is the first visit of κ in F and as ∞ if κ never visits F. Finally, we define the value $\mathrm{val}(\kappa)$ of a run κ as $\mathrm{first}_F(\kappa)$ if $\mathrm{last}_B(\kappa) = \perp$ and as $\max\{\mathrm{last}_B(\kappa), \mathrm{first}_F(\kappa)\}$ otherwise. Given two runs κ, κ', we say that κ "makes progress faster" than κ' if $\mathrm{val}(\kappa) < \mathrm{val}(\kappa')$.

Before we can define our new prophecies, we need additional notation that gives us the value of a best run (over several traces) if we fix all but the last trace. For each $i \in I_\exists$, each vertex v of $\mathfrak{T}$, each state q of $\mathcal{A}_i^{\mathfrak{T}}$, and traces $t_0, t_1, \dots, t_{i-1} \in \mathrm{Tr}(\mathfrak{T})$, we define

$$\mathrm{opt}_i(q, v, t_0, \dots, t_{i-1}) = \min_{t_i \in \mathrm{Tr}(\mathfrak{T}_{\mathrm{S}(v)})} \mathrm{val}(\kappa_i),$$

where κ_i is the run of $(\mathcal{A}_i^{\mathfrak{T}})_q$ on $\mathrm{mrg}(t_0, \dots, t_{i-1}, t_i)$.

We are now ready to introduce our prophecies.

▶ **Definition 14** (Prophecy construction). *For each $i \in I_\forall$, vectors $\bar{u} = (v_0, v_1, \dots, v_{i+1})$ and $\bar{v} = (v'_0, v'_1, \dots, v'_{i+1})$ of vertices such that $v'_j \in \mathrm{S}(v_j)$ for $j \leq i+1$, and each vector $\bar{q} = (q_1, q_3, \dots, q_{i+1})$ of states q_j of $\mathcal{A}_j^{\mathfrak{T}}$, we define*

$$\begin{aligned}(\mathfrak{P}_i)_{\bar{q}}^{\bar{u},\bar{v}} = \{&\mathrm{mrg}(t_0, t_2, \dots, t_i) \mid t_0 \in \mathrm{Tr}(\mathfrak{T}_{v'_0}), t_2 \in \mathrm{Tr}(\mathfrak{T}_{v'_2}), \dots, t_i \in \mathrm{Tr}(\mathfrak{T}_{v'_i}), \textit{ and} \\ &\textit{there exist } t_1 \in \mathrm{Tr}(\mathfrak{T}_{v'_1}), t_3 \in \mathrm{Tr}(\mathfrak{T}_{v'_3}), \dots, t_{i+1} \in \mathrm{Tr}(\mathfrak{T}_{v'_{i+1}}) \textit{ such that} \\ &\quad \textit{the run } \kappa_j \textit{ of } (\mathcal{A}_j^{\mathfrak{T}})_{q_j} \textit{ on } \mathrm{mrg}(t_0, \dots, t_j) \textit{ is accepting and satisfies} \\ &\quad\quad \mathrm{val}(\kappa_j) = \mathrm{opt}_j(q_j, v_j, t_0, \dots, t_{j-1}) \textit{ for all odd } j \leq i+1\}.\end{aligned}$$

Intuitively, the prophecy $(\mathfrak{P}_i)_{\bar{q}}^{\bar{u},\bar{v}}$ indicates that it is safe for Verifier $i+1$ to move from v_{i+1} to v'_{i+1} as before, and additionally, among all successors of v_{i+1}, picking v'_{i+1} allows Verifier $i+1$ to make the most progress. This prophecy is intended to be used, if for odd $j < i+1$, Verifier j already picked the move v_j to v'_j which was optimal. In particular, the prophecy repeats the prophecies made (in the same round) for Verifier j with odd $j < i+1$.

These prophecies are ω-regular.

▶ **Lemma 15.** *For each $i \in I_\forall$, each vector $\bar{q}$ of states and all vectors $\bar{u}, \bar{v}$ such that the ℓ-th vertex of $\bar{v}$ is a successor of the ℓ-th vertex of $\bar{u}$, the set $(\mathfrak{P}_i)_{\bar{q}}^{\bar{u},\bar{v}}$ is ω-regular.*

We set up the manipulated system and formula, then state the completeness result.

▶ **Definition 16.** *We define $\Xi = (\Xi_i)_{i \in I_\forall}$ and $\mathcal{P} = (\mathrm{P}_i)_{i \in I_\forall}$. Let Ξ_i be the set of QPTL formulas that contains sentences $(\xi_i)_{\bar{q}}^{\bar{u},\bar{v}}$ for each vector $\bar{q}$ of states and vectors $\bar{u}, \bar{v}$ such that the ℓ-th vertex of $\bar{v}$ is a successor of the ℓ-th vertex of $\bar{u}$. The sentence $(\xi_i)_{\bar{q}}^{\bar{u},\bar{v}}$ expresses the prophecy $(\mathfrak{P}_i)_{\bar{q}}^{\bar{u},\bar{v}}$ using only even trace variables π_j with $j \leq i$.*

Moreover, let P_i *be such that it contains the prophecy variable* $(\mathfrak{p}_i)^{\bar{u},\bar{v}}_{\bar{q}}$ *for each vector* $\bar{q}$ *of states and vectors* $\bar{u}, \bar{v}$ *such that the* ℓ*-th vertex of* $\bar{v}$ *is a successor of the* ℓ*-th vertex of* $\bar{u}$*. The prophecy variable* $(\mathfrak{p}_i)^{\bar{u},\bar{v}}_{\bar{q}}$ *corresponds to* $(\xi_i)^{\bar{u},\bar{v}}_{\bar{q}}$*.*

▶ **Lemma 17.** *Assume* $\mathfrak{T} \models \varphi$ *and each* $\mathcal{A}^{\mathfrak{T}}_i$ *is a one-pair Rabin automaton. Then the coalition of Verifier-players has a winning collection of strategies in* $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ *with* Ξ *and* $\mathcal{P}$ *as in Definition 16.*

Our proof of this result generalizes the one for the safety case (see Lemma 11), using the same notation introduced there: We define a collection of strategies for the Verifier-players and then show, that $\mathfrak{T} \models \varphi$ implies that this collection is winning in $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$.

Assume the play is in Round (r, i) for $r \geq 0$ and $i \in I_\exists$. Thus, it is in a position

$$((v'_0, \dots, v'_{i-1}, v_i, \dots, v_{k-1}), q, i)$$

and Verifier i has to move. We review some information Verifier i can use to base her choice on. In Round r, Verifier i has access to the first r letters of the traces $\lambda_{\mathrm{AP}}(\rho_0), \lambda_{\mathrm{AP}}(\rho_1), \dots, \lambda_{\mathrm{AP}}(\rho_i)$ induced by the play[6]. Let q_j be the state that $\mathcal{A}^{\mathfrak{T}}_j$ has reached on $\mathrm{mrg}(\lambda_{\mathrm{AP}}(\rho_0)[0,r), \lambda_{\mathrm{AP}}(\rho_1)[0,r), \dots, \lambda_{\mathrm{AP}}(\rho_j)[0,r))$ for odd $j \leq i$. Also, let $((v_0, \dots, v_{i-1}, v_i, \dots, v_{k-1}), q, 0)$ be the position reached in Round $(r, 0)$.

▶ **Definition 18** (Strategy construction). *Let*

$$M_i(\bar{u}, \bar{v}, \bar{q}) = \{v'_i \in \mathrm{S}(v_i) \mid (\mathfrak{p}_{i-1})^{\bar{u},\bar{w}}_{\bar{q}} \in \mathrm{Prophecies}(v'_{i-1})\}$$

where $\bar{u} = (v_0, \dots, v_i)$*,* $\bar{v} = (v'_0, \dots, v'_{i-1}, v_i)$*,* $\bar{w} = (v'_0, \dots, v'_{i-1}, v'_i)$ *and* $\bar{q} = (q_1, q_3, \dots, q_i)$*.*

If $M_i(\bar{u}, \bar{v}, \bar{q})$ *is nonempty, then Verifier* i *can move to any* v'_i *in the set (the "*NONEMPTY*-case"). If it is empty, then Verifier* i *can move to any* $v'_i \in \mathrm{S}(v_i)$ *(the "*EMPTY*-case").*

Now, Lemma 17 can be proven by generalizing the proof of Lemma 11. Then, we directly obtain our main result by combining Lemma 8 and Lemma 17.

▶ **Theorem 19.** *Let* Ξ *and* $\mathcal{P}$ *as in Definition 16, and assume that each* $\mathcal{A}^{\mathfrak{T}}_i$ *is a one-pair Rabin automaton. The coalition of Verifier-players has a winning collection of strategies for* $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$ *if and only if* $\mathfrak{T} \models \varphi$*.*

5.2 Beyond One-Pair Rabin Automata

We have proven Theorem 19 for the case that all automata referred to in the prophecies are one-pair Rabin automata. We sketch how to drop this constraint closely following Beutner and Finkbeiner [5].

A Rabin automaton $\mathcal{R}$ with pairs $(B_1, F_1), \dots, (B_m, F_m)$ can be seen as a union of m one-pair Rabin automata (B_i, F_i)-$\mathcal{R}$. A run that is accepting in (B_i, F_i)-$\mathcal{R}$ is also accepting in $\mathcal{R}$ and each accepting run of $\mathcal{R}$ is accepting in some (B_i, F_i)-$\mathcal{R}$. Before we present how to go from one-pair to general Rabin automata, we recall that the prophecies exclusively talk about traces provided by Falsifier. Hence, if Falsifier signals that his future traces belong to prophecy $\mathcal{P}$ which, for example, implies that Rabin automaton $\mathcal{R}$ accepts them, then Falsifier can also specify the pair(s) for which $\mathcal{R}$ will accept. No further context from the Verifier-players is needed.

The idea is to annotate our prophecies (which refer to $\mathcal{A}^{\mathfrak{T}}_1, \mathcal{A}^{\mathfrak{T}}_3, \dots, \mathcal{A}^{\mathfrak{T}}_{k-1}$) with the indices of the pairs used for acceptance. More concretely, for a prophecy $(\mathfrak{P}_i)^{\bar{u},\bar{v}}_{\bar{q}}$, we introduce prophecies of the form $(\mathfrak{P}_i)^{\bar{u},\bar{v}}_{\bar{q},\bar{a}}$, where $\bar{a}$ is a vector of indices of the same length as $\bar{q}$. Say the

j-th entry of $\bar{q}$ is q_ℓ, then the prophecy $(\mathfrak{P}_i)^{\bar{u},\bar{v}}_{\bar{q}}$ refers to $(\mathcal{A}^{\mathfrak{T}}_\ell)_{q_\ell}$ (among other automata), and say the j-th entry of $\bar{a}$ is a_j, then $(\mathfrak{P}_i)^{\bar{u},\bar{v}}_{\bar{q},\bar{a}}$ refers to the one-pair Rabin automaton (B_{a_j}, F_{a_j})-$(\mathcal{A}^{\mathfrak{T}}_\ell)_{q_\ell}$ (among other automata). Except for the refined prophecies, no other changes to the game are required.

In a play of the game $\mathcal{G}(\mathfrak{T}^{\mathcal{P}}, \varphi^{\mathcal{P},\Xi})$, Falsifier specifies with his prophecies for which pairs the Rabin automata $\mathcal{A}^{\mathfrak{T}}_1, \mathcal{A}^{\mathfrak{T}}_3, \ldots, \mathcal{A}^{\mathfrak{T}}_{k-1}$ can accept.

In Round $(0,0)$, it is specified for $\mathcal{A}^{\mathfrak{T}}_1$, in Round $(0,1)$, Verifier 1 chooses one option specified by Falsifier for $\mathcal{A}^{\mathfrak{T}}_1$. That choice is simply made by her move. In Round $(0,0)$, Falsifier moves from $\bullet$ to v_0, and in Round $(0,1)$, Verifier 1 moves from $\bullet$ to v_1. Say Prophecies(v_0) indicates that (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$ accepts when Verifier 1 moves to v_1, then she has chosen (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$. She then (in order be guaranteed to build a winning play) has to stick to it for the rest of the play, meaning that from now on all her moves must have a target vertex v such that the prophecy given by Falsifier in the move before indicates that (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$ accepts if she moves to v.

In Round $(0,2)$, it is specified also for $\mathcal{A}^{\mathfrak{T}}_3$, in Round $(0,3)$, Verifier 3 chooses one option for $\mathcal{A}^{\mathfrak{T}}_3$ and then has to stick to it for the rest of the play. Additionally, she must pick the option chosen by Verifier 1 for $\mathcal{A}^{\mathfrak{T}}_1$ to ensure consistency. In Round $(0,2)$, Falsifier moves from $\bullet$ to v_2. For example, say Verifier 3 has the option to move from $\bullet$ to v', v'', or v''' with Prophecies(v_2) indicating that (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$ and (B_j, F_j)-$\mathcal{A}^{\mathfrak{T}}_3$ accept if she moves to v', that (B_k, F_k)-$\mathcal{A}^{\mathfrak{T}}_1$ and (B_j, F_j)-$\mathcal{A}^{\mathfrak{T}}_3$ accept if she moves to v'', and that (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$ and (B_ℓ, F_ℓ)-$\mathcal{A}^{\mathfrak{T}}_3$ accept if she moves to v'''. Then she must pick v' or v''' because Verifier 1 has committed to (B_i, F_i)-$\mathcal{A}^{\mathfrak{T}}_1$ which Verifier 3 must honor. If Verifier 3 moves to v' she picks (B_j, F_j)-$\mathcal{A}^{\mathfrak{T}}_3$, if she moves to v''' she picks (B_ℓ, F_ℓ)-$\mathcal{A}^{\mathfrak{T}}_3$.

The same principle applies for the next rounds. After Round $(0, k-1)$, for each automaton a pair has been fixed to which the Verifier-players stick for the rest of the play.

We end by substantiating the claim in Footnote 4 on Page 8 that our construction can also be applied to formulas with arbitrary, i.e., not strictly alternating, trace quantifier prefixes.

▶ Remark 20. In case of of arbitrary trace quantifier prefixes, we have one player in the coalition for each maximal block of existentially quantified trace variables in the prefix, who selects traces for these variables. For example, for a formula of the form

$$\forall\pi_0 \exists\pi_1 \exists\pi_2 \forall\pi_3 \exists\pi_4 \exists\pi_5.\ \psi$$

with trace quantifier-free ψ, we have just two players (say, Player 12 and Player 45) in the coalition. Player 12 is in charge of the variables π_1 and π_2, and Player 45 is in charge of the variables π_4 and π_5. These two players are in a coalition against Falsifier, who is in charge of the variables π_0 and π_3. In each round, Falsifier picks a move for π_0, then Player 12 picks a move for π_1 and a move for π_2, then Falsifier picks a move for π_3, and then Player 45 picks a move for π_4 and a move for π_5.

Note that just adding universally quantified dummy trace variables between any pair of consecutive existentially quantified trace variables is logically equivalent, but increases the complexity of our approach (as that depends on the number of players).

6 Conclusion

We have presented the first sound and complete game-based algorithm for full HyperLTL (even HyperQPTL) model-checking via multi-player games with hierarchical information, thereby extending the prophecy-based framework of Coenen et al. and Beutner and Finkbeiner from the $\forall^*\exists^*$-fragment to arbitrary quantifier prefixes. Similarly, we have extended Winter and

Zimmermann's game-based characterization of the existence of computable Skolem functions witnessing $\mathfrak{T} \models \varphi$. One way of understanding our result is that we compute finite-state implementations of Skolem functions via transducers with ω-regular lookahead.

In further work, we aim to exhibit lower bounds on the number of prophecies needed for completeness. Recall that HyperLTL model-checking is Tower-complete. Even an elementary number of prophecies, while unlikely, would not contradict any complexity-lower bounds, as solving multi-player games of hierarchical information is already Tower-complete (in the number of players). However, as it is, the number of prophecies we use is only bounded by an n-fold exponential, where n is linear in the number of quantifier alternations (taking both trace and propositional quantification into account), as the size of the automata $\mathcal{A}_i^{\mathfrak{T}}$ grows like that.

One of the motivations for the introduction of the game-based approach using prophecies is the need for complementation of ω-automata in the classical automata-based HyperLTL model-checking algorithm. For $\forall^*\exists^*$-formulas, the game-based approach "only" requires deterministic ω-automata, which can be directly computed from the quantifier-free part, which is essentially an LTL formula. On the other hand, the automata-based approach needs automata complementation, already for $\forall^*\exists^*$-formulas. For arbitrary quantifier alternations, both approaches require complementation. In future work, we aim to study whether nondeterministic or history-deterministic automata are sufficient to construct complete prophecies.

Finally, we are currently extending the techniques developed here to even more expressive logics, e.g., HyperRecHML, recursive Hennessy-Milner logic with trace quantification [2].

References

1 Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284, 1991. doi:10.1016/0304-3975(91)90224-P.

2 Luca Aceto, Antonis Achilleos, Elli Anastasiadi, and Adrian Francalanza. Monitoring hyperproperties with circuits. In Mohammad Reza Mousavi and Anna Philippou, editors, *FORTE 2022*, volume 13273 of *LNCS*, pages 1–10. Springer, 2022. doi:10.1007/978-3-031-08679-3_1.

3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *LICS 2012*, pages 65–74. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.18.

4 Dietmar Berwanger, Anup Basil Mathew, and Marie van den Bogaard. Hierarchical information and the synthesis of distributed strategies. *Acta Informatica*, 55(8):669–701, 2018. doi:10.1007/S00236-017-0306-5.

5 Raven Beutner and Bernd Finkbeiner. Prophecy variables for hyperproperty verification. In *CSF 2022*, pages 471–485. IEEE, 2022. doi:10.1109/CSF54842.2022.9919658.

6 Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *POST 2014*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.

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

8 Norine Coenen, Bernd Finkbeiner, César Sánchez, and Leander Tentrup. Verifying hyperliveness. In Isil Dillig and Serdar Tasiran, editors, *CAV 2019, Part I*, volume 11561 of *LNCS*, pages 121–139. Springer, 2019. doi:10.1007/978-3-030-25540-4_7.

9 Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.worldcat.org/oclc/310535248.

10 E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986. doi:10.1145/4904.4999.

11 Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Syst. Theory*, 10:289–303, 1977. doi:10.1007/BF01683280.

12 Emmanuel Filiot and Sarah Winter. Synthesizing computable functions from rational specifications over infinite words. *Int. J. Found. Comput. Sci.*, 35(01n02):179–214, 2024. doi:10.1142/S012905412348009X.

13 Bernd Finkbeiner, Christopher Hahn, Jana Hofmann, and Leander Tentrup. Realizing omega-regular hyperproperties. In Shuvendu K. Lahiri and Chao Wang, editors, *CAV 2020, Part II*, volume 12225 of *LNCS*, pages 40–63. Springer, 2020. doi:10.1007/978-3-030-53291-8_4.

14 Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. Algorithms for Model Checking HyperLTL and HyperCTL*. In Daniel Kroening and Corina S. Pasareanu, editors, *CAV 2015, Part I*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015. doi:10.1007/978-3-319-21690-4_3.

15 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002. doi:10.1007/3-540-36387-4.

16 Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In *ICALP 1972*, pages 45–60, 1972.

17 Felix Klein and Martin Zimmermann. How much lookahead is needed to win infinite games? *Log. Methods Comput. Sci.*, 12(3), 2016. doi:10.2168/LMCS-12(3:4)2016.

18 Corto Mascle and Martin Zimmermann. The keys to decidable HyperLTL satisfiability: Small models or very simple formulas. In Maribel Fernández and Anca Muscholl, editors, *CSL 2020*, volume 152 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CSL.2020.29.

19 Daryl McCullough. Noninterference and the composability of security properties. In *SSP 1988*, pages 177–186. IEEE Computer Society, 1988. doi:10.1109/SECPRI.1988.8110.

20 Amir Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE, October 1977. doi:10.1109/SFCS.1977.32.

21 Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS 1990, Volume II*, pages 746–757. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89597.

22 Markus N. Rabe. *A temporal logic approach to information-flow control.* PhD thesis, Saarland University, 2016. URL: http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/.

23 A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with appplications to temporal logic. *Theor. Comput. Sci.*, 49:217–237, 1987. doi:10.1016/0304-3975(87)90008-9.

24 Sarah Winter and Martin Zimmermann. Finite-state strategies in delay games. *Inf. Comput.*, 272:104500, 2020. doi:10.1016/J.IC.2019.104500.

25 Sarah Winter and Martin Zimmermann. Prophecies all the way: Game-based model-checking for HyperQPTL beyond $\forall^*\exists^*$. *arXiv*, 2504.08575, 2025. doi:10.48550/arXiv.2504.08575.