Logical Methods in Computer Science
Volume 22, Issue 4, 2026, pp. 13:1–13:28
https://lmcs.episciences.org/

Submitted Nov. 21, 2025
Published Nov. 03, 2026

THE COMPLEXITY OF GENERALIZED HYPERLTL WITH STUTTERING AND CONTEXTS

GAËTAN REGAUD [a] AND MARTIN ZIMMERMANN [b]

[a] ENS Rennes, Rennes, France
e-mail address: gaetan.regaud@ens-rennes.fr

[b] Aalborg University, Aalborg, Denmark
e-mail address: mzi@cs.aau.dk

Abstract. We settle the complexity of satisfiability, finite-state satisfiability, and model-checking for generalized HyperLTL with stuttering and contexts, an expressive logic for the specification of asynchronous hyperproperties. Such properties cannot be specified in HyperLTL, as it is restricted to synchronous hyperproperties.

Nevertheless, we prove that satisfiability is Σ^1_1-complete and thus not harder than for HyperLTL. On the other hand, we prove that model-checking and finite-state satisfiability are equivalent to truth in second-order arithmetic, and thus much harder than the decidable HyperLTL model-checking problem and the Σ^1_0-complete HyperLTL finite-state satisfiability problem. The lower bounds for the model-checking and finite-state satisfiability problems hold even when only allowing stuttering or only allowing contexts.

This paper is part of the special issue *Selected Papers of the 16th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2025).*

1. Introduction

The introduction of hyperlogics has been an important milestone in the specification, analysis, and verification of hyperproperties [CS10], properties that relate several execution traces of a system. These have important applications in, e.g., information-flow security. Before their introduction, temporal logics (e.g., LTL, CTL, CTL*, QPTL and PDL) were only able to reason about a single trace at a time [Fin17]. However, this is not sufficient to reason about the complex flow of information. For example, noninterference [GM82] requires that all traces that coincide on their low-security inputs also coincide on their low-security

A preliminary version of this article has been published in the proceedings of GandALF 2025 [RZ25]. This version contains a new section on finite-state satisfiability and results for the simple fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and $\mathrm{HyperLTL}_{\mathrm{S}}$.

Gaëtan Regaud has been supported by the European Union.



Martin Zimmermann has been supported by DIREC – Digital Research Centre Denmark and by the project 'Hyperlogics: Expressiveness, Monitorability and Tools (H.-Lo)' of the Icelandic Research Fund, project no. 2612260-051.

DOI:10.46298/LMCS-22(4:13)2026
© G. Regaud and M. Zimmermann
Creative Commons

outputs, independently of their high-security inputs (which may differ, but may not leak via low-security outputs). This property intuitively "compares" the inputs and outputs on pairs of traces and is therefore not expressible in LTL, which can only reason about individual traces.

The first generation of hyperlogics has been introduced by equipping the temporal logics LTL, CTL*, QPTL, and PDL with quantification over traces, obtaining HyperLTL [CFK+14], HyperCTL* [CFK+14], HyperQPTL [FHHT20, Rab16] and HyperPDL-Δ [GMO20]. They are able to express noninterference (and many other hyperproperties), have intuitive syntax and semantics, and a decidable model-checking problem, making them attractive specification languages for hyperproperties. For example, noninterference is expressed in HyperLTL as

$$\forall x.\, \forall x'.\, \Big(\bigwedge\nolimits_{i \in I_\ell} \mathbf{G}\, (i_x \leftrightarrow i_{x'}) \Big) \rightarrow \Big(\bigwedge\nolimits_{o \in O_\ell} \mathbf{G}\, (o_x \leftrightarrow o_{x'}) \Big),$$

where I_ℓ is the set of low-security inputs and O_ℓ is the set of low-security outputs. The formula expresses that for all pairs of traces (assigned to the trace variables x and x'), if they agree at every position on the truth value of all propositions in I_ℓ, then they must also agree at every position on the truth value of all propositions in O_ℓ.

All the hyperlogics mentioned above are synchronous in the sense that time passes on all quantified traces at the same rate, e.g., the subformula $i_x \leftrightarrow i_{x'}$ above, which is in the scope of an always operator, is evaluated, for all $n \in \mathbb{N}$, at the positions n of x and x'. However, not every system is synchronous, e.g., multi-threaded systems in which processes are not scheduled in lockstep. The first generation of hyperlogics is not able to express asynchronous hyperproperties. Hence, in a second wave, several asynchronous hyperlogics have been introduced, which employ various mechanisms to enable the asynchronous evolution of time on different traces under consideration.

- Asynchronous HyperLTL (A-HLTL) [BCB+21] adds so-called trajectories to HyperLTL, which intuitively specify the rates at which different traces evolve.
- HyperLTL with stuttering (HyperLTL$_\mathrm{S}$) [BPS21] changes the semantics of the temporal operators of HyperLTL so that time does not evolve synchronously on all traces, but instead evolves based on LTL-definable stuttering.
- HyperLTL with contexts (HyperLTL$_\mathrm{C}$) [BPS21] adds a context-operator to HyperLTL, which allows to select a subset of traces on which time passes synchronously, while it is frozen on all others.
- Generalized HyperLTL with stuttering and contexts (GHyLTL$_{\mathrm{S+C}}$) [BBST24] adds both stuttering and contexts to HyperLTL and additionally allows trace quantification under the scope of temporal operators, which HyperLTL does not allow.
- H_μ [GMO21] adds trace quantification to the linear-time μ-calculus with asynchronous semantics for the modal operators.
- Hypernode automata (HA) [BHNdC23] combine automata and hyperlogic with stuttering.

The known relations between these logics are depicted in Figure 1.

However, all these logics have an undecidable model-checking problem, thereby losing one of the key features of the first generation logics. Thus, much research focus has been put on fragments of these logics, e.g., simple GHyLTL$_{\mathrm{S+C}}$ and simple HyperLTL$_\mathrm{S}$, which both have a decidable model-checking problem. The same is true for fragments of A-HLTL [BCB+21], H_μ [GMO21], and HA [BHNdC23]. Furthermore, for almost all of the logics, the satisfiability problem has never been studied. Thus, the landscape of complexity results for the second

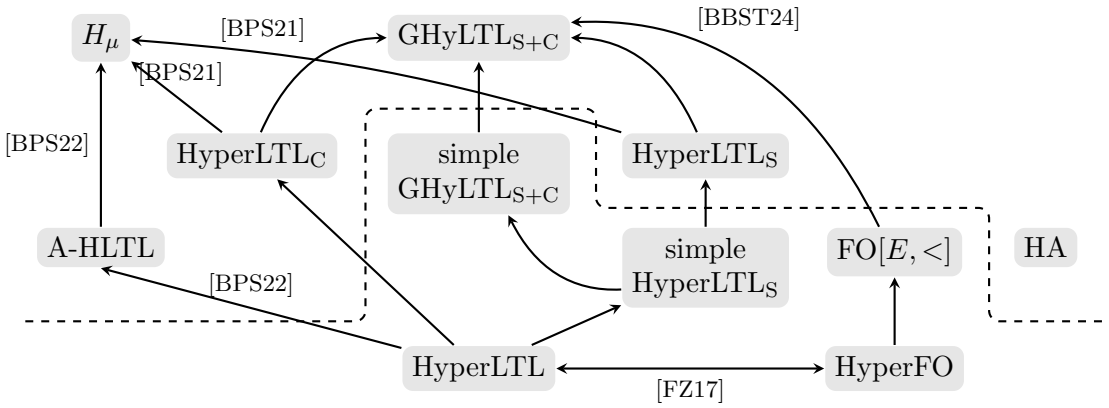


Figure 1: The landscape of logics for asynchronous hyperproperties. Here, FO$[E,<]$ is first-order logic with the equal-level predicate E and order $<$ evaluated over sets of infinite words, and HyperFO is a fragment that is equivalent to HyperLTL [FZ17]. Arrows denote known inclusions (where the inclusion of HyperLTL into A-HLTL requires an additional proposition) and the dashed line denotes the decidability border for model-checking. For non-inclusions, we refer the reader to work on the expressiveness of asynchronous hyperlogics [BPS22, BBST24].

generation is still incomplete, while the complexity of satisfiability, finite-state satisfiability, and model-checking for the first generation has been settled (see Table 1).

In these preceding works, and here, one uses the complexity of arithmetic, predicate logic over the signature $(+, \cdot, <, \in)$, as a yardstick. In first-order arithmetic, quantification ranges over natural numbers while second-order arithmetic adds quantification over sets of natural numbers and third-order arithmetic adds quantification over sets of sets of natural numbers. Figure 2 gives an overview of the arithmetic, analytic, and "third" hierarchy, each spanned by the classes of languages definable by restricting the number of alternations of the highest-order quantifiers, i.e., Σ^0_n contains languages definable by formulas of first-order arithmetic with $n-1$ quantifier alternations, starting with an existential one.

Table 1: List of complexity results for synchronous hyperlogics. "T2A-equiv." ("T3A-equiv.") stands for "equivalent to truth in second-order (third-order) arithmetic". The result for HyperPDL-Δ satisfiability can be shown using techniques developed by Fortin et al. for HyperLTL satisfiability [FKTZ25], the result for HyperPDL-Δ finite-state satisfiability follows from the lower bound for its fragment HyperLTL and HyperPDL-Δ model-checking being decidable.

Logic	Satisfiability	Finite-state Satisfiability	Model-checking
HyperLTL	Σ^1_1-compl. [FKTZ25]	Σ^0_1-compl. [FH16]	Tower-compl. [Rab16, MZ20]
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	Tower-compl. [GMO20]
HyperQPTL	Σ^2_1-complete. [RZ26b]	Σ^0_1-compl. [Rab16]	Tower-compl. [Rab16]
HyperQPTL$^+$	T3A-equiv. [RZ26b]	T3A-equiv. [RZ26b]	T3A-equiv. [RZ26b]
Hyper^{2}LTL	T3A-equiv. [FRZ26]	T3A-equiv. [FRZ26]	T3A-equiv. [FRZ26]
HyperCTL*	Σ^2_1-compl. [FKTZ25]	Σ^0_1-compl. [FH16]	Tower-compl. [Rab16, MZ20]

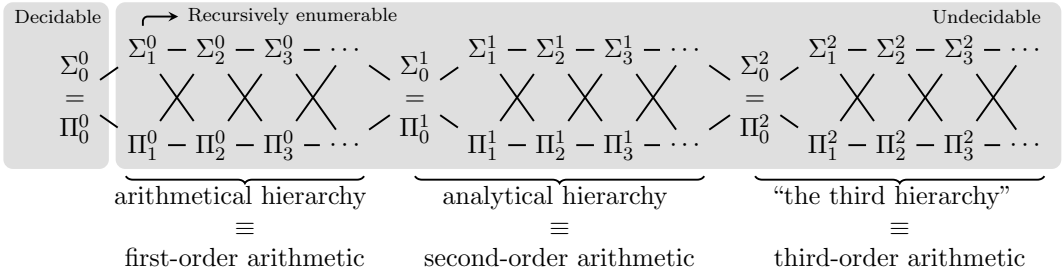


Figure 2: The arithmetical hierarchy, the analytical hierarchy, and beyond.

Our goal is to obtain a similarly clear picture for asynchronous logics, both for model-checking (for which, as mentioned above, only some lower bounds are known) and for satisfiability (for which almost nothing is known). In this work, we focus on $\mathrm{GHyLTL}_{\mathrm{S+C}}$, as it is one of the most expressive logics and subsumes many of the other logics.

Our Contributions. First, we study the satisfiability problem. It is known that HyperLTL satisfiability is Σ_1^1-complete. Here, we show that satisfiability for $\mathrm{GHyLTL}_{\mathrm{S+C}}$ is not harder, i.e., also Σ_1^1-complete. The lower bound is trivial, as HyperLTL is a fragment of $\mathrm{GHyLTL}_{\mathrm{S+C}}$. However, we show that adding stuttering, contexts, and quantification under the scope of temporal operators all do not increase the complexity of satisfiability. Intuitively, the underlying reason is that $\mathrm{GHyLTL}_{\mathrm{S+C}}$ is a first-order linear-time logic, i.e., it is evaluated over a set of traces and Skolem functions for the existentially quantified variables map tuples of traces to traces. We exploit this property to show that every satisfiable formula has a countable model. The existence of such a "small" model can be captured in Σ_1^1. This should be contrasted with HyperCTL*, which only adds quantification under the scope of temporal operators to HyperLTL, but with a branching-time semantics. In HyperCTL*, one can write formulas that have only uncountable models, which in turn allows one to encode existential third-order quantification [FKTZ25]. Consequently, HyperCTL* satisfiability is Σ_1^2-hard (and in fact Σ_1^2-complete) and thus much harder than that of $\mathrm{GHyLTL}_{\mathrm{S+C}}$. Let us also mention that these results settle the complexity of FO$[E, <]$ satisfiability: it is Σ_1^1-complete as well. Here, the lower bound is inherited from HyperLTL and the upper bound follows from the fact that FO$[E, <]$ can be translated into $\mathrm{GHyLTL}_{\mathrm{S+C}}$ [BBST24]. Similarly, satisfiability for all fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ that include HyperLTL is also Σ_1^1-complete (see Figure 1).

Then, we turn our attention to the model-checking problem, which we show to be equivalent to truth in second-order arithmetic and therefore much harder than satisfiability. Here, we show that, surprisingly, the lower bounds already hold for the fragments $\mathrm{HyperLTL}_{\mathrm{S}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$, i.e., adding one feature is sufficient, and adding the second one does not increase the complexity further. This result also has to be contrasted with HyperLTL model-checking: adding stuttering or contexts takes the model-checking problem from Tower-complete [FRS15] (and thus decidable) to truth in second-order arithmetic.

The intuitive reason for model-checking being much harder than satisfiability is that every satisfiable formula of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ has a countable model while in the model-checking problem, one has to deal with possibly uncountable models, as (finite) transition systems may have uncountably many traces. This allows us to encode second-order arithmetic in $\mathrm{HyperLTL}_{\mathrm{S}}$ and in $\mathrm{HyperLTL}_{\mathrm{C}}$. A similar situation occurs for a fragment of second-order

Table 2: List of our complexity results for $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and its fragments. "T2A-equiv." stands for "equivalent to truth in second-order arithmetic".

Logic	Satisfiability	Finite-state Satisfiability	Model-checking
$\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ^1_1-compl. (Thm. 4.1)	T2A-equiv. (Thm. 6.4)	T2A-equiv. (Thm. 5.4)
$\mathrm{HyperLTL}_{\mathrm{S}}$	Σ^1_1-compl. (Cor. 4.2)	T2A-equiv. (Thm. 6.4)	T2A-equiv. (Thm. 5.4)
$\mathrm{HyperLTL}_{\mathrm{C}}$	Σ^1_1-compl. (Cor. 4.2)	T2A-equiv. (Thm. 6.4)	T2A-equiv. (Thm. 5.4)
simple $\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ^1_1-compl. (Rem 4.3)	Σ^0_1-compl. (Rem. 6.5)	TOWER-compl. [BBST24, Rab16]
simple $\mathrm{HyperLTL}_{\mathrm{S}}$	Σ^1_1-compl. (Rem. 4.3)	Σ^0_1-compl. (Rem. 6.5)	TOWER-compl. [BPS21, Rab16]

HyperLTL [FRZ26], but satisfiability is typically harder than or as hard as model-checking (see Table 1).

Finally, we study the finite-state satisfiability problem. Here, we show that it is also equivalent to truth in second-order arithmetic and therefore much harder than (plain) satisfiability. Again, the lower bounds already hold for the fragments $\mathrm{HyperLTL}_{\mathrm{S}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$. The reason for finite-state satisfiability being much harder than satisfiability is again found in the fact that every satisfiable $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence has a countable model (i.e., a countable set of traces), but we can write a HyperLTL formula that has only finite-state models that have uncountably many traces. This in turn allows one to encode second-order quantification in $\mathrm{HyperLTL}_{\mathrm{S}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$.

Our results are collected in Table 2. Note in particular that the simple fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and $\mathrm{HyperLTL}_{\mathrm{S}}$ are only simple when it comes to model-checking: Their satisfiability problems are as hard as for full $\mathrm{GHyLTL}_{\mathrm{S+C}}$.

2. Preliminaries

The set of nonnegative integers is denoted by $\mathbb{N}$. An alphabet is a nonempty finite set Σ. The set of infinite words over Σ is Σ^ω. Given $w \in \Sigma^\omega$ and $i \in \mathbb{N}$, $w(i)$ denotes the i-th letter of w (starting with $i = 0$). Let AP be a fixed finite set of propositions. A trace σ is an element of $(2^{\mathrm{AP}})^\omega$, and a pointed trace is a pair (σ, i) consisting of a trace and a pointer $i \in \mathbb{N}$ pointing to a position of σ. It is initial if $i = 0$.

A transition system is a tuple $\mathcal{T} = (V, E, I, \ell)$ where V is a nonempty finite set of vertices, $E \subseteq V \times V$ is a set of directed edges, $I \subseteq V$ is a nonempty set of initial vertices, and $\ell \colon V \to 2^{\mathrm{AP}}$ is a labeling function that maps each vertex to a set of propositions. We require that each vertex has at least one outgoing edge. A run of a transition system $\mathcal{T}$ is an infinite word $v_0 v_1 \cdots \in V^\omega$ such that $v_0 \in I$ and $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. The set $\mathrm{Tr}(\mathcal{T}) = \{\ell(v_0)\ell(v_1)\cdots \mid v_0 v_1 \cdots \text{is a run of } \mathcal{T}\}$ is the set of traces induced by $\mathcal{T}$, which is always nonempty.

2.1. LTL with Past.

The logic PLTL [Pnu77] extends classical LTL [Pnu77] by adding temporal operators to describe past events. The syntax of PLTL is defined as

$$\theta ::= p \mid \neg\theta \mid \theta \vee \theta \mid \mathbf{X}\,\theta \mid \theta\,\mathbf{U}\,\theta \mid \mathbf{Y}\,\theta \mid \theta\,\mathbf{S}\,\theta,$$

where $p \in \mathrm{AP}$. Here, **Y** (yesterday) and **S** (since) are the past-variants of **X** (next) and **U** (until). We use the usual syntactic sugar, e.g., $\wedge$, $\rightarrow$, $\leftrightarrow$, **F** (eventually), **G** (always), **O** (once, the past-variant of eventually), and **H** (historically, the past-variant of always).

The semantics of PLTL is defined over pointed traces (σ, i) as

- $(\sigma, i) \models p$ if $p \in \sigma(i)$,
- $(\sigma, i) \models \neg\theta$ if $(\sigma, i) \not\models \theta$,
- $(\sigma, i) \models \theta_1 \vee \theta_2$ if $(\sigma, i) \models \theta_1$ or $(\sigma, i) \models \theta_2$,
- $(\sigma, i) \models \mathbf{X}\,\theta$ if $(\sigma, i+1) \models \theta$,
- $(\sigma, i) \models \theta_1\,\mathbf{U}\,\theta_2$ if there exists an $i' \geq i$ such that $(\sigma, i') \models \theta_2$ and $(\sigma, j) \models \theta_1$ for all $i \leq j < i'$,
- $(\sigma, i) \models \mathbf{Y}\,\theta$ if $i > 0$ and $(\sigma, i-1) \models \theta$, and
- $(\sigma, i) \models \theta_1\,\mathbf{S}\,\theta_2$ if there exists an $0 \leq i' \leq i$ such that $(\sigma, i') \models \theta_2$ and $(\sigma, j) \models \theta_1$ for all $i' < j \leq i$.

2.2. **Stuttering.** Let Γ be a finite set of PLTL formulas and σ a trace. We say that $i \in \mathbb{N}$ is a proper Γ-changepoint of σ if either $i = 0$ or $i > 0$ and there is a $\theta \in \Gamma$ such that $(\sigma, i) \models \theta$ if and only if $(\sigma, i-1) \not\models \theta$, i.e., the truth value of θ at positions i and $i-1$ differs. If σ has only finitely many proper Γ-changepoints (say i is the largest one), then $i+1, i+2, \ldots$ are Γ-changepoints of σ by convention. Thus, every trace has infinitely many Γ-changepoints.

The Γ-successor of a pointed trace (σ, i) is the pointed trace $\mathrm{succ}_\Gamma(\sigma, i) = (\sigma, i')$ where i' is the minimal Γ-changepoint of σ that is strictly greater than i. Dually, the Γ-predecessor of (σ, i) for $i > 0$ is the pointed trace $\mathrm{pred}_\Gamma(\sigma, i) = (\sigma, i')$ where i' is the maximal Γ-changepoint of σ that is strictly smaller than i; $\mathrm{pred}_\Gamma(\sigma, 0)$ is undefined.

Remark 2.1. Let σ be a trace over some set AP' of propositions and let Γ only contain PLTL formulas using propositions in AP'' such that $\mathrm{AP}' \cap \mathrm{AP}'' = \emptyset$ (note that this is in particular satisfied, if $\Gamma = \emptyset$). Then, 0 is the only proper Γ-changepoint of σ. Hence, by our convention, every position of σ is a Γ-changepoint which implies $\mathrm{succ}_\Gamma(\sigma, i) = (\sigma, i+1)$ for all i and $\mathrm{pred}_\Gamma(\sigma, i) = (\sigma, i-1)$ for all $i > 0$.

2.3. **Generalized HyperLTL with Stuttering and Contexts.** Recall that HyperLTL extends LTL with trace quantification in prenex normal form, i.e., first some traces are quantified and then an LTL formula is evaluated (synchronously) over these traces. $\mathrm{GHyLTL}_{\mathrm{S+C}}$ extends HyperLTL by two new constructs to express asynchronous hyperproperties:

- Contexts allow one to restrict the set of quantified traces over which time passes when evaluating a formula, e.g., $\langle C \rangle \psi$ for a nonempty finite set C of trace variables expresses that ψ holds when time passes synchronously on the traces bound to variables in C, but time does not pass on variables bound to variables that are not in C.
- Furthermore, temporal operators are labeled by sets Γ of PLTL formulas and time stutters w.r.t. Γ, e.g., $\mathbf{X}_\Gamma$ stutters to the Γ-successor on each trace in the current context.

Also, unlike HyperLTL, $\mathrm{GHyLTL}_{\mathrm{S+C}}$ allows trace quantification under the scope of temporal operators and adds past operators.

Fix a finite set VAR of trace variables. The syntax of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ is given by the grammar

$$\varphi ::= p_x \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle C \rangle \varphi \mid \mathbf{X}_\Gamma\,\varphi \mid \varphi\,\mathbf{U}_\Gamma\,\varphi \mid \mathbf{Y}_\Gamma\,\varphi \mid \varphi\,\mathbf{S}_\Gamma\,\varphi \mid \exists x.\varphi \mid \forall x.\varphi,$$

where $p \in \mathrm{AP}$, $x \in \mathsf{VAR}$, $C \in 2^{\mathsf{VAR}} \setminus \{\emptyset\}$, and Γ ranges over finite sets of PLTL formulas. A sentence is a formula without free trace variables, which are defined as expected. To declutter our notation, we write $\langle x_1, \ldots, x_n \rangle$ for contexts instead of $\langle \{x_1, \ldots, x_n\} \rangle$. We

measure the size of a formula in the number of syntactically distinct subformulas it has (including the PLTL formulas labeling temporal operators).

To define the semantics of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ we need to introduce some notation. A (pointed) trace assignment $\Pi\colon \mathsf{VAR} \to (2^{\mathrm{AP}})^\omega \times \mathbb{N}$ is a partial function that maps trace variables to pointed traces. The domain of a trace assignment Π, written as $\mathrm{Dom}(\Pi)$, is the set of variables for which Π is defined. For $x \in \mathsf{VAR}$, $\sigma \in (2^{\mathrm{AP}})^\omega$, and $i \in \mathbb{N}$, the assignment $\Pi[x \mapsto (\sigma, i)]$ maps x to (σ, i) and each other $x' \in \mathrm{Dom}(\Pi) \setminus \{x\}$ to $\Pi(x')$.

Fix a set Γ of PLTL formulas and a context $C \subseteq \mathsf{VAR}$. The (Γ, C)-successor and the (Γ, C)-predecessor of a trace assignment Π are the trace assignments $\mathrm{succ}_{(\Gamma,C)}(\Pi)$ and $\mathrm{pred}_{(\Gamma,C)}(\Pi(x))$ defined as

$$\mathrm{succ}_{(\Gamma,C)}(\Pi)(x) = \begin{cases} \mathrm{succ}_\Gamma(\Pi(x)) & \text{if } x \in C, \\ \Pi(x) & \text{otherwise,} \end{cases}$$

and

$$\mathrm{pred}_{(\Gamma,C)}(\Pi)(x) = \begin{cases} \mathrm{pred}_\Gamma(\Pi(x)) & \text{if } x \in C, \\ \Pi(x) & \text{otherwise.} \end{cases}$$

The (Γ, C)-predecessor of Π is only defined when $\mathrm{pred}_\Gamma(\Pi(x))$ is defined for all $x \in C$.[1]

Iterated (Γ, C)-successors and (Γ, C)-predecessors are defined as

- $\mathrm{succ}^0_{(\Gamma,C)}(\Pi) = \Pi$ and $\mathrm{succ}^{j+1}_{(\Gamma,C)}(\Pi) = \mathrm{succ}_{(\Gamma,C)}(\mathrm{succ}^j_{(\Gamma,C)}(\Pi))$, and
- $\mathrm{pred}^0_{(\Gamma,C)}(\Pi) = \Pi$ and $\mathrm{pred}^{j+1}_{(\Gamma,C)}(\Pi) = \mathrm{pred}_{(\Gamma,C)}(\mathrm{pred}^j_{(\Gamma,C)}(\Pi))$, which may again be undefined.

Now, the semantics of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ is defined with respect to a set $\mathcal{L}$ of traces, an assignment Π, and a context $C \subseteq \mathsf{VAR}$ as

- $(\mathcal{L}, \Pi, C) \models p_x$ if $\Pi(x) = (\sigma, i)$ and $p \in \sigma(i)$,
- $(\mathcal{L}, \Pi, C) \models \neg\varphi$ if $(\mathcal{L}, \Pi, C) \not\models \varphi$,
- $(\mathcal{L}, \Pi, C) \models \varphi_1 \vee \varphi_2$ if $(\mathcal{L}, \Pi, C) \models \varphi_1$ or $(\mathcal{L}, \Pi, C) \models \varphi_2$,
- $(\mathcal{L}, \Pi, C) \models \langle C' \rangle \varphi$ if $(\mathcal{L}, \Pi, C') \models \varphi$,
- $(\mathcal{L}, \Pi, C) \models \mathbf{X}_\Gamma\, \varphi$ if $(\mathcal{L}, \mathrm{succ}_{(\Gamma,C)}(\Pi), C) \models \varphi$,
- $(\mathcal{L}, \Pi, C) \models \varphi_1\, \mathbf{U}_\Gamma\, \varphi_2$ if there exists an $i \geq 0$ such that $(\mathcal{L}, \mathrm{succ}^i_{(\Gamma,C)}(\Pi), C) \models \varphi_2$ and $(\mathcal{L}, \mathrm{succ}^j_{(\Gamma,C)}(\Pi), C) \models \varphi_1$ for all $0 \leq j < i$,
- $(\mathcal{L}, \Pi, C) \models \mathbf{Y}_\Gamma\, \varphi$ if $\mathrm{pred}_{(\Gamma,C)}(\Pi)$ is defined and $(\mathcal{L}, \mathrm{pred}_{(\Gamma,C)}(\Pi), C) \models \varphi$,
- $(\mathcal{L}, \Pi, C) \models \varphi_1\, \mathbf{S}_\Gamma\, \varphi_2$ if there exists an $i \geq 0$ such that $\mathrm{pred}^i_{(\Gamma,C)}(\Pi)$ is defined, $(\mathcal{L}, \mathrm{pred}^i_{(\Gamma,C)}(\Pi), C) \models \varphi_2$, and $(\mathcal{L}, \mathrm{pred}^j_{(\Gamma,C)}(\Pi), C) \models \varphi_1$ for all $0 \leq j < i$,
- $(\mathcal{L}, \Pi, C) \models \exists x.\varphi$ if there exists a trace $\sigma \in \mathcal{L}$ such that $(\mathcal{L}, \Pi[x \mapsto (\sigma, 0)], C) \models \varphi$, and
- $(\mathcal{L}, \Pi, C) \models \forall x.\varphi$ if for all traces $\sigma \in \mathcal{L}$ we have $(\mathcal{L}, \Pi[x \mapsto (\sigma, 0)], C) \models \varphi$.

Note that quantification ranges over *initial* pointed traces, even under the scope of a temporal operator.

[1] Note that this definition differs from the one in the original paper introducing $\mathrm{GHyLTL}_{\mathrm{S+C}}$ [BBST24], which required that $\mathrm{pred}_\Gamma(\Pi(x))$ is defined for every $x \in \mathrm{Dom}(\Pi)$. However, this is too restrictive, as the predecessor operation is only applied to traces $\Pi(x)$ with $x \in C$. Furthermore, it leads to undesirable side effects, e.g., $\mathcal{L} \models \varphi$ may hold, but $\mathcal{L} \models \forall x.\varphi$ does not hold, where x is a variable not occurring in φ.

We say that a set $\mathcal{L}$ of traces satisfies a sentence φ, written $\mathcal{L} \models \varphi$, if $(\mathcal{L}, \emptyset, \mathsf{VAR}) \models \varphi$, where $\emptyset$ represents the variable assignment with empty domain. Furthermore, a transition system $\mathcal{T}$ satisfies φ, written $\mathcal{T} \models \varphi$, if $\mathrm{Tr}(\mathcal{T}) \models \varphi$.

Remark 2.2. Let Π be an assignment, C a context, and φ a quantifier-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formula. Then, we have $(\mathcal{L}, \Pi, C) \models \varphi$ if and only if $(\mathcal{L}', \Pi, C) \models \varphi$ for all sets $\mathcal{L}, \mathcal{L}'$ of traces, i.e., satisfaction of quantifier-free formulas is independent of the set of traces, only the assignment Π and the context C matter. Hence, we will often write $(\Pi, C) \models \varphi$ for quantifier-free φ.

The logics HyperLTL [CFK+14], $\mathrm{HyperLTL}_{\mathrm{S}}$ [BPS21] (HyperLTL with stuttering), and $\mathrm{HyperLTL}_{\mathrm{C}}$ [BPS21] (HyperLTL with contexts) are syntactic fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$. Let us say that a formula is past-free, if it does not use the temporal operators $\mathbf{Y}$ and $\mathbf{S}$, Then,

- HyperLTL is the fragment obtained by considering only past-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas in prenex normal form, by disallowing the context operator $\langle\cdot\rangle$, and by indexing all temporal operators by the empty set,
- $\mathrm{HyperLTL}_{\mathrm{S}}$ is the fragment obtained by considering only past-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas in prenex normal form, by disallowing the context operator $\langle\cdot\rangle$, and by indexing all temporal operators by sets of past-free PLTL formulas, and
- $\mathrm{HyperLTL}_{\mathrm{C}}$ is the fragment obtained by considering only past-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas in prenex normal form and by indexing all temporal operators by the empty set.

2.4. **Arithmetic and Complexity Classes for Undecidable Problems.** To capture the complexity of undecidable problems, we consider formulas of arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over the structure $(\mathbb{N}, +, \cdot, <, \in)$. A type 0 object is a natural number in $\mathbb{N}$, and a type 1 object is a subset of $\mathbb{N}$. In the following, we use lower-case roman letters (possibly with decorations) for first-order variables, and upper-case roman letters (possibly with decorations) for second-order variables. Every fixed natural number is definable in first-order arithmetic, so we freely use them as syntactic sugar. For more detailed definitions, we refer to [Rog87].

Our benchmark is second-order arithmetic, i.e., predicate logic with quantification over type 0 and type 1 objects. Arithmetic formulas with a single free first-order variable define sets of natural numbers. In particular, Σ^1_1 contains the sets of the form

$$\{x \in \mathbb{N} \mid \exists X_1 \subseteq \mathbb{N}.\ \cdots \exists X_k \subseteq \mathbb{N}.\ \psi(x, X_1, \ldots, X_k)\},$$

where ψ is a formula of arithmetic with arbitrary quantification over type 0 objects (but no second-order quantifiers). Furthermore, truth in second-order arithmetic is the following problem: Given a sentence φ of second-order arithmetic, do we have $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$?

3. "Small" Models for $\mathrm{GHyLTL}_{\mathrm{S+C}}$

In this section, we prove that every satisfiable $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence has a countable model, which is an important stepping stone for determining the complexity of the satisfiability problem in Section 4. To do so, we first prove that for every $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ there is a $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ_p in prenex normal form that is "almost" equivalent in the following sense: A set $\mathcal{L}$ of traces is a model of φ if and only if $\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}$ is a model of φ_p, where $\mathcal{L}_{\mathtt{pos}}$ is a countable set of traces that is independent of φ.

Before we formally state our result, let us illustrate the obstacle we have to overcome, which traces are in $\mathcal{L}_{\texttt{pos}}$, and how they help to overcome the obstacle. For the sake of simplicity, we use an always formula as example, even though it is syntactic sugar: The same obstacle occurs for the until operator, but there we would have to deal with the two subformulas of $\psi_1 \,\mathbf{U}_\Gamma\, \psi_2$ instead of the single one of $\mathbf{G}_\Gamma\, \psi$.

In a formula of the form $\exists x.\, \mathbf{G}_\Gamma\, \exists x'.\, \psi$, the always operator acts like a quantifier too, i.e., the formula expresses that there is a trace σ such that for *every* position i on σ, there is another trace σ' (that may depend on i) so that $([x \mapsto (\sigma, i), x' \mapsto (\sigma', 0)], C)$ satisfies ψ, where C is the current context. Obviously, moving the quantification of x' before the always operator does not yield an equivalent formula, as x' then no longer depends on i. Instead, we simulate the implicit quantification over positions i by explicit quantification over natural numbers encoded by traces of the form $\emptyset^i\{\#\}\emptyset^\omega$, where $\# \notin \mathrm{AP}$ is a fresh proposition.

Recall that Γ is a set of PLTL formulas over AP, i.e., Remark 2.1 applies. Thus, the i-th Γ-successor of $(\emptyset^i\{\#\}\emptyset^\omega, 0)$ is the unique pointed trace $(\emptyset^i\{\#\}\emptyset^\omega, j)$ satisfying the formula $\#$, which is the case for $j = i$. Thus, we can simulate the evaluation of the formula ψ at the i-th (Γ, C)-successor by the formula $\langle (C \cup \{x_i\}) \setminus \{x'\}\rangle\, \mathbf{F}_\Gamma(\#_{x_i} \wedge \langle C\rangle \psi)$, where C is still the current context, i.e., we add x_i to the current context to reach the i-th Γ-successor (over the extended context $C \cup \{x_i\}$) and then evaluate ψ over the context C that our original formula is evaluated over. But, to simulate the quantification of x' correctly, we have to take it out of the scope for the eventually operator in order to ensure that the evaluation of ψ takes place on the initial pointed trace, as we have moved the quantifier for x' before the eventually.

To implement the same approach for the past operators, we also need to be able to let time proceed backwards from the position of $\emptyset^i\{\#\}\emptyset^\omega$ marked by $\#$ back to the initial position. To identify that position by a formula, we rely on the fact that $\neg\, \mathbf{Y}\, \texttt{true}_x$ holds exactly at position 0 of the trace bound to x, where $\texttt{true}_x$ is a shorthand for $p_x \vee \neg p_x$ for some $p \in \mathrm{AP}$. Then, the i-th Γ-predecessor of the unique position marked by $\#$ is the unique position where $\neg\, \mathbf{Y}\, \texttt{true}_x$ holds. So, we define $\mathcal{L}_{\texttt{pos}} = \{\emptyset^i\{\#\}\emptyset^\omega \mid i \in \mathbb{N}\}$.

The proof of the next lemma shows that one can, in a similar way, move quantifiers over *all* operators.

Lemma 3.1. *Let* AP *be a finite set of propositions and* $\# \notin \mathrm{AP}$*. For every GHyLTL$_{S+C}$ sentence φ over* AP*, there exists a polynomial-time computable GHyLTL$_{S+C}$ sentence φ_p in prenex normal form over* $\mathrm{AP} \cup \{\#\}$ *such that for all nonempty $\mathcal{L} \subseteq (2^{\mathrm{AP}})^\omega$: $\mathcal{L} \models \varphi$ if and only if $\mathcal{L} \cup \mathcal{L}_{pos} \models \varphi_p$.*

Proof. First, let us show how to express the set $\mathcal{L}_{\texttt{pos}}$ of traces encoding positions on traces (i.e., natural numbers). Let $\alpha_{\texttt{pos}}$ be the conjunction of the following formulas (where x and x' are fresh variables not appearing in φ):

- $\forall x.(\mathbf{F}_\emptyset\, \#_x) \rightarrow [((\neg \#_x)\, \mathbf{U}_\emptyset(\#_x \wedge \mathbf{X}_\emptyset\, \mathbf{G}_\emptyset\, \neg \#_x)) \wedge \mathbf{G}_\emptyset \bigwedge_{p \in \mathrm{AP}} \neg p_x]$: if a trace contains a $\#$, then it contains exactly one $\#$ and no other proposition holds anywhere in the trace.
- $\exists x.\#_x$: the trace $\{\#\}\emptyset^\omega$ is in the model (assuming the previous formula holds).
- $\forall x.\exists x'.(\mathbf{F}_\emptyset\, \#_x) \rightarrow (\mathbf{F}_\emptyset(\#_x \wedge \mathbf{X}_\emptyset\, \#_{x'}))$: if $\emptyset^i\{\#\}\emptyset^\omega$ is in the model, then also $\emptyset^{i+1}\{\#\}\emptyset^\omega$ (again assuming the first formula holds).

Hence, every model of the conjunction $\alpha_{\texttt{pos}}$ must contain the traces in $\mathcal{L}_{\texttt{pos}}$, but no other traces containing $\#$.

Next, let us introduce the shorthands $\exists^{\texttt{pos}} x.\psi$ and $\forall^{\texttt{pos}} x.\psi$ for $\exists x.(\mathbf{F}_\emptyset\, \#_x) \wedge \psi$ and $\forall x.(\mathbf{F}_\emptyset\, \#_x) \rightarrow \psi$, i.e., when ranging over $\mathcal{L} \cup \mathcal{L}_{\texttt{pos}}$ as above, then the quantifiers $\exists^{\texttt{pos}}$ and $\forall^{\texttt{pos}}$

only range over traces in $\mathcal{L}_{\mathtt{pos}}$ encoding positions. Conversely, to ensure that the quantifiers coming from φ only range over traces not in $\mathcal{L}_{\mathtt{pos}}$, we use the shorthands $\exists^{\mathtt{ori}} x.\psi$ and $\forall^{\mathtt{ori}} x.\psi$ for $\exists x.(\mathbf{G}_\emptyset \neg \#_x) \wedge \psi$ and $\forall x.(\mathbf{G}_\emptyset \neg \#_x) \rightarrow \psi$. Now, let φ' denote the $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence obtained from φ by inductively replacing each subformula $\exists x.\psi$ by $\exists^{\mathtt{ori}} x.\psi$ and each subformula $\forall x.\psi$ by $\forall^{\mathtt{ori}} x.\psi$. Then, we have $\mathcal{L} \models \varphi$ if and only if $\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}} \models \varphi'$.

In the following, we present equivalences for all constructs in the syntax of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ that allow one to move quantifiers to the front of a formula. These equivalences hold for all sets $\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}$ with nonempty $\mathcal{L}$, for all assignments Π, and for all contexts C. But note that we will use the shorthands $\exists^{\mathtt{pos}}$, $\forall^{\mathtt{pos}}$, $\exists^{\mathtt{ori}}$, and $\forall^{\mathtt{ori}}$ in the equivalences, i.e., when applying these equivalences to φ' from above, the resulting sentence is not yet in prenex normal form with respect to the quantifiers $\exists$ and $\forall$. However, the resulting sentence does not have any quantifier $\exists$ and $\forall$ under the scope of a temporal operator or a context. Hence, it can be brought into prenex normal form by moving the quantifiers $\exists$ and $\forall$ over the Boolean operators.

In the following, Q stands for a quantifier of the form $\exists$ or $\forall$.

- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \neg Q x.\psi$ if and only if $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \overline{Q}^{\mathtt{ori}} x.\neg\psi$, where $\overline{\exists}^{\mathtt{ori}} = \forall^{\mathtt{ori}}$ and $\overline{\forall}^{\mathtt{ori}} = \exists^{\mathtt{ori}}$.
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models (Q x.\psi_1) \vee \psi_2$ if and only if $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models Q^{\mathtt{ori}} x.(\psi_1 \vee \psi_2)$ where we assume w.l.o.g. that x is not a free variable in ψ_2 (which can always be achieved by renaming x in ψ_2).
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \langle C \rangle' Q x.\psi$ if and only if $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models Q^{\mathtt{ori}} x.\langle C \rangle' \psi$ for every $C' \subseteq \mathsf{VAR}$.
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \mathbf{X}_\Gamma\, Q x.\psi$ if and only if $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models Q^{\mathtt{ori}} x.\langle C \setminus \{x\} \rangle\, \mathbf{X}_\Gamma \langle C \rangle \psi$. Here, we use contexts in order to capture the fact that quantification of x ranges over initial pointed traces: When moving the quantification of x in front of the next operator we use the context $C \setminus \{x\}$ to ensure that time does not pass on the trace bound to x when evaluating the next operator. After the next operator, we restore the context C again.

In the following, we will use similar constructions for the until operator to correctly move the quantification in front of the temporal operator.

- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \psi_1\, \mathbf{U}_\Gamma (Q x.\psi_2)$ if and only if

$$\begin{aligned}(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \exists^{\mathtt{pos}} x_i.Q^{\mathtt{ori}} x.\forall^{\mathtt{pos}} x_j.\\ &\big[\langle (C \cup \{x_i\}) \setminus \{x\} \rangle\, \mathbf{F}_\Gamma(\#_{x_i} \wedge \langle C \rangle \psi_2)\big] \wedge \\ &\big[(\langle x_i, x_j \rangle\, \mathbf{F}_\emptyset(\#_{x_j} \wedge \mathbf{X}_\emptyset\, \mathbf{F}_\emptyset\, \#_{x_i})) \rightarrow \\ &\qquad \langle (C \cup \{x_j\}) \setminus \{x\} \rangle\, \mathbf{F}_\Gamma(\#_{x_j} \wedge \langle C \rangle \psi_1)\big],\end{aligned}$$

  where we assume w.l.o.g. that x is not a free variable in ψ_1 and where x_i and x_j are fresh variables.
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models (Q x.\psi_1)\, \mathbf{U}_\Gamma\, \psi_2$ if and only if

$$\begin{aligned}(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \exists^{\mathtt{pos}} x_i.\forall^{\mathtt{pos}} x_j.Q^{\mathtt{ori}} x.\\ &\big[\langle (C \cup \{x_i\}) \setminus \{x\} \rangle\, \mathbf{F}_\Gamma(\#_{x_i} \wedge \langle C \rangle \psi_2)\big] \wedge \\ &\big[(\langle x_i, x_j \rangle\, \mathbf{F}_\emptyset(\#_{x_j} \wedge \mathbf{X}_\emptyset\, \mathbf{F}_\emptyset\, \#_{x_i})) \rightarrow \\ &\qquad \langle (C \cup \{x_j\}) \setminus \{x\} \rangle\, \mathbf{F}_\Gamma(\#_{x_j} \wedge \langle C \rangle \psi_1)\big],\end{aligned}$$

where we now assume w.l.o.g. that x is not a free variable in ψ_2 and where x_i and x_j are fresh variables.

In both cases where we move a quantifier out of the until, we make the quantification over the positions implicit in the semantics of the until explicit and use $\mathbf{F}_\Gamma$ to reach the position where $\#_{x_i}$ ($\#_{x_j}$, respectively) holds, which must then reach the i-th (j-th) Γ-successor on the traces in the context C. A usual, we must take x out of these contexts but add x_i (x_j). Finally, the formula $\langle x_i, x_j\rangle \mathbf{F}_\emptyset(\#_{x_j} \wedge \mathbf{X}_\emptyset \mathbf{F}_\emptyset \#_{x_i})$ holds if and only if the position assigned to x_i is strictly greater than that assigned to x_j. Note that, in the first case, we quantify x_i before x and in the second case, we quantify both x_i and x_j before x to capture the correct dependencies of the variables.

The past modalities are translated as their corresponding future variants, we just have to let time pass "backwards" on the traces in $\mathcal{L}_{\mathtt{pos}}$.

- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \mathbf{Y}_\Gamma\, Qx.\psi$ if and only if $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models Q^{\mathtt{ori}}x.\langle C \setminus \{x\}\rangle\, \mathbf{Y}_\Gamma \langle C\rangle \psi$.
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models \psi_1\, \mathbf{S}_\Gamma (Qx.\psi_2)$ if and only if

$$
\begin{aligned}
(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models\ & \exists^{\mathtt{pos}} x_i.Q^{\mathtt{ori}}x.\forall^{\mathtt{pos}} x_j. \\
& \langle x_i\rangle \mathbf{F}_\emptyset \Big[\#_{x_i} \wedge \langle (C \cup \{x_i\}) \setminus \{x\}\rangle \mathbf{O}_\Gamma(\neg\, \mathbf{Y}\, \mathtt{true}_{x_i} \wedge \langle C\rangle \psi_2)\Big] \wedge \\
& \Big[(\langle x_i, x_j\rangle \mathbf{F}_\emptyset(\#_{x_j} \wedge \mathbf{X}_\emptyset \mathbf{F}_\emptyset \#_{x_i})) \rightarrow \\
& \qquad \langle x_i\rangle \mathbf{F}_\emptyset \left[\#_{x_i} \wedge \langle (C \cup \{x_j\}) \setminus \{x\}\rangle \mathbf{O}_\Gamma(\neg\, \mathbf{Y}\, \mathtt{true}_{x_j} \wedge \langle C\rangle \psi_1)\right]\Big],
\end{aligned}
$$

  where we assume w.l.o.g. that x is not a free variable in ψ_1 and where x_i and x_j are fresh variables.
- $(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models (Qx.\psi_1)\, \mathbf{S}_\Gamma\, \psi_2$ if and only if

$$
\begin{aligned}
(\mathcal{L} \cup \mathcal{L}_{\mathtt{pos}}, \Pi, C) \models\ & \exists^{\mathtt{pos}} x_i.\forall^{\mathtt{pos}} x_j.Q^{\mathtt{ori}}x. \\
& \langle x_i\rangle \mathbf{F}_\emptyset \Big[\#_{x_i} \wedge \langle (C \cup \{x_i\}) \setminus \{x\}\rangle \mathbf{O}_\Gamma(\neg\, \mathbf{Y}\, \mathtt{true}_{x_i} \wedge \langle C\rangle \psi_2)\Big] \wedge \\
& \Big[(\langle x_i, x_j\rangle \mathbf{F}_\emptyset(\#_{x_j} \wedge \mathbf{X}_\emptyset \mathbf{F}_\emptyset \#_{x_i})) \rightarrow \\
& \qquad \langle x_i\rangle \mathbf{F}_\emptyset \left[\#_{x_i} \wedge \langle (C \cup \{x_j\}) \setminus \{x\}\rangle \mathbf{O}_\Gamma(\neg\, \mathbf{Y}\, \mathtt{true}_{x_j} \wedge \langle C\rangle \psi_1)\right]\Big],
\end{aligned}
$$

  where we now assume w.l.o.g. that x is not a free variable in ψ_2 and where x_i and x_j are fresh variables.

Note that the current context C that the formula is evaluated on is used to construct the equivalent formulas. However, this is purely syntactic information, i.e., given a sentence φ and a subformula ψ one can determine the context that ψ is evaluated on: It is the last context that appears on the path from the root to the subformula ψ in the syntax tree of φ' (and is VAR if there is no context operator on the path).

Thus, we can use these equivalences as rewriting rules to turn the $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ' obtained from φ above in polynomial-time (recall that we measure the size of a formula in the number of distinct subformulas) into a sentence φ'', which has no quantifier under the scope of a temporal operator. Thus, bringing $\varphi'' \wedge \alpha_{\mathtt{pos}}$ into prenex normal form (using the standard rules for Boolean operators) yields a sentence φ_p with the desired properties. □

The previous lemma shows that for every $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ, there exists a $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ_p in prenex normal form that is almost equivalent, i.e., equivalent modulo adding the countable set $\mathcal{L}_{\mathtt{pos}}$ of traces to the model. Hence, showing that every satisfiable sentence in prenex normal form has a countable model implies that every satisfiable sentence has a countable model. Thus, we focus on prenex normal form sentences to study the cardinality of models of $\mathrm{GHyLTL}_{\mathrm{S+C}}$.

The proof of the next lemma shows that every model of a sentence φ contains a countable subset that is closed under the application of Skolem functions, generalizing a similar construction for HyperLTL [FZ17]. Such a set is also a model of φ.

Lemma 3.2. *Every satisfiable GHyLTL$_{S+C}$ sentence φ in prenex normal form has a countable model.*

Proof. Let $\varphi = Q_1 x_1. \cdots Q_k x_k. \psi$ be a $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formula in prenex normal form with quantifier-free ψ and $\{Q_1, \ldots, Q_k\} \subseteq \{\exists, \forall\}$. Let $\mathcal{L}$ be a model of φ. If $\mathcal{L}$ is already countable, then there is nothing to show. So, let us assume that $\mathcal{L}$ is uncountable.

For each existentially quantified x in φ, let U_x be the set of variables quantified universally before x. For every such x, say with $U_x = \{y_1, \ldots, y_{|U_x|}\}$, there exists a Skolem function $f_x \colon \mathcal{L}^{|U_x|} \to \mathcal{L}$ such that for each assignment Π mapping each existentially quantified variable x to $f_x(\Pi(y_1), \ldots, \Pi(y_{|U_x|}))$, we have $(\Pi, \mathsf{VAR}) \models \psi$.

We fix such Skolem functions f_x for the rest of the proof. For $\mathcal{L}' \subseteq \mathcal{L}$ and an existentially quantified variable x, we define

$$f_x(\mathcal{L}') = \{f_x(\sigma_1, \ldots, \sigma_{|U_x|}) \mid \sigma_1, \ldots, \sigma_{|U_x|} \in \mathcal{L}'\}.$$

Note that if $\mathcal{L}'$ is finite, then $f_x(\mathcal{L}')$ is also finite. Now, define $\mathcal{L}_0 = \{\sigma\}$ for some $\sigma \in \mathcal{L}$ and $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \bigcup_x f_x(\mathcal{L}_i)$ for all $i \in \mathbb{N}$, where x ranges over all existentially quantified variables. We show that $\mathcal{L}_\omega = \bigcup_{i \in \mathbb{N}} \mathcal{L}_i$ is a countable model of φ. In fact, $\mathcal{L}_\omega$ is trivially countable, as it is a countable union of finite sets.

By definition of Skolem functions, every assignment Π that maps universally quantified variables to traces in $\mathcal{L}_\omega$ and uses the Skolem functions for existentially quantified variables satisfies $(\mathcal{L}, \Pi, \mathsf{VAR}) \models \psi$. Thus, we also have $(\mathcal{L}_\omega, \Pi, \mathsf{VAR}) \models \psi$, as there are no quantifiers in ψ. Now, using an induction over the quantifier prefix of φ (from the inside out), one can prove that $(\mathcal{L}_\omega, \Pi, \mathsf{VAR}) \models Q_j x_j \cdots Q_k x_k \psi$, i.e., $\mathcal{L}_\omega$ is a model of φ. Here, we use the fact that $\mathcal{L}_\omega$ is closed under applications of the Skolem functions, i.e., whenever $\sigma_1, \ldots, \sigma_{|Ux|}$ are in $\mathcal{L}_\omega$, then $f_x(\sigma_1, \ldots, \sigma_{|Ux|})$ is also in $\mathcal{L}_\omega$. □

By combining Lemma 3.1 and Lemma 3.2 we obtain our main result of this section.

Theorem 3.3. *Every satisfiable GHyLTL$_{S+C}$ formula has a countable model.*

4. GHyLTL$_{\mathrm{S+C}}$ Satisfiability

In this section, we study the satisfiability problem for $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and its fragments: Given a sentence φ, is there a nonempty set $\mathcal{L}$ of traces such that $\mathcal{L} \models \varphi$? Due to Theorem 3.3, we can restrict ourselves to countable models. The proof of the following theorem shows that the existence of such a model can be captured in arithmetic, generalizing constructions developed for HyperLTL [FKTZ25] to handle stuttering and contexts.

Theorem 4.1. *The GHyLTL$_{S+C}$ satisfiability problem is Σ^1_1-complete with respect to polynomial-time reductions.*

Proof. As explained above, the Σ^1_1-lower bound already holds for the fragment HyperLTL of $\mathrm{GHyLTL}_{\mathrm{S+C}}$. Hence, let us consider the upper bound, which we prove by extending the proof technique showing that HyperLTL satisfiability is in Σ^1_1: We express the existence of a countable model, of Skolem functions for the existential variables, and witnesses of their correctness using type 1 objects. To this end, we need to capture the semantics of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ in arithmetic. To do so, we will make heavy use of the following fact: a function of the form $\mathbb{N}^k \to \mathbb{N}^\ell$ for $k, \ell \in \mathbb{N}$ can be encoded by a subset of $\mathbb{N}$ via its graph. Furthermore, this encoding is implementable in first-order arithmetic. Hence, we will in the following freely use such functions as type 1 objects.

Our goal is to write a Σ^1_1-sentence ar with a single free (first-order) variable x such that $(\mathbb{N}, +, \cdot, <, \in) \models ar(n)$ if and only if n encodes a satisfiable $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ. Here, and in the following, we fix some encoding of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas by natural numbers, which can be chosen such that it is implementable in first-order arithmetic. Due to Lemma 3.1, we can assume w.l.o.g. that φ has the form $= Q_1 x_1. \cdots Q_n x_n. \psi$ with quantifier-free ψ, as every $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence is equi-satisfiable to one in prenex normal form. Furthermore, due to Lemma 3.2, we know that φ is satisfiable if and only if it has a countable model.

First, let us remark that we can encode a nonempty countable set $\mathcal{L}$ of traces as a function $L \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ mapping trace names and positions to subsets of AP encoded by natural numbers. We assume w.l.o.g. that the variables $x_1, \ldots, x_n$ are equal to $1, \ldots, n$, which can always be achieved by renaming variables. Then, an assignment Π with domain $\{x_1, \ldots, x_n\}$ and codomain $\mathcal{L}$ can be encoded by the list $(\ell_1, i_1, \ldots, \ell_n, i_n) \in \mathbb{N}^{2n}$, where, for $\Pi(x_j) = (\sigma_j, i_j)$, ℓ_j is the name of the trace σ_j w.r.t. the encoding L. As there is a bijection between $\mathbb{N}^*$ and $\mathbb{N}$ that is implementable in first-order arithmetic, such an assignment can also be encoded by a single natural number. Hence, a countable model can be encoded by a type 1 object and a variable assignment by a type 0 object. Similarly, a context over $\{x_1, \ldots, x_n\}$ can be encoded by a type 0 object, as it is a list of natural numbers.

We start by capturing the semantics of the PLTL formulas we use to define the stuttering. Let θ be such a PLTL formula, let Θ be the set of subformulas of θ, and let σ be a trace. Then, we define the θ-expansion of σ as the function $e_\sigma \colon \Theta \times \mathbb{N} \to \{0, 1\}$ defined as

$$e_\sigma(\theta', i) = \begin{cases} 1 & \text{if } (\sigma, i) \models \theta', \\ 0 & \text{if } (\sigma, i) \not\models \theta'. \end{cases}$$

The θ-expansion of σ is uniquely identified by the following consistency requirements, i.e., it is the only function from $\Theta \times \mathbb{N}$ to $\{0, 1\}$ satisfying these requirements:

- $e_\sigma(p, i) = 1$ if and only if $p \in \sigma(i)$,
- $e_\sigma(\neg\theta', i) = 1$ if and only if $e_\sigma(\theta, i) = 0$,
- $e_\sigma(\theta_1 \vee \theta_2, i) = 1$ if and only if $e_\sigma(\theta_1, i) = 1$ or $e_\sigma(\theta_2, i) = 1$,
- $e_\sigma(\mathbf{X}\,\theta', i) = 1$ if and only if $e_\sigma(\theta', i + 1) = 1$,
- $e_\sigma(\theta_1\,\mathbf{U}\,\theta_2, i) = 1$ if and only if there exists an $i' \geq i$ such that $e_\sigma(\theta_2, i') = 1$ and $e_\sigma(\theta_1, j) = 1$ for all $i \leq j < i'$,
- $e_\sigma(\mathbf{Y}\,\theta', i) = 1$ if and only if $i > 0$ and $e_\sigma(\theta', i - 1) = 1$, and
- $e_\sigma(\theta_1\,\mathbf{S}\,\theta_2, i) = 1$ if and only if there exists an $0 \leq i' \leq i$ such that $e_\sigma(\theta_2, i') = 1$ and $e_\sigma(\theta_1, j) = 1$ for all $i' < j \leq i$.

Note that all requirements have a first-order flavor. Slightly more formally: One can write a formula $\alpha_e(L, x, t, E)$ of first-order arithmetic with free variables L (encoding a countable set

of traces as explained above), x (representing a trace name), t (encoding a PLTL formula θ with set Θ of subformulas), and E (encoding a function from $\Theta \times \mathbb{N}$ to $\{0,1\}$) that is satisfied in $(\mathbb{N}, +, \cdot, <, \in)$ if and only if the function encoded by E is the θ-expansion of the trace named x in the set of traces encoded by L.

Using the formula α_e, we can write a formula $\alpha_{cp}(L, x, g, i, E)$ of first-order arithmetic with free variables L (encoding again a countable set of traces), x (representing again a trace name), g (encoding a finite set Γ of PLTL formulas), i (representing a position), and E (which encodes the θ-expansion of the trace named x in $\mathcal{L}$ for every $\theta \in \Gamma$) that is satisfied in $(\mathbb{N}, +, \cdot, <, \in)$ if and only if i is a Γ-changepoint (proper or not) of the trace named x in the set of traces encoded by L.

Furthermore, using the formula α_{cp}, we can write formulas $\alpha_s(L, g, c, a, a', i, E)$ and $\alpha_p(L, g, c, a, a', i, E)$ of first-order arithmetic with free variables L (encoding a set $\mathcal{L}$ of traces as above), g (encoding a set Γ of PLTL formulas as above), c (encoding a context C), a and a' (encoding assignments), and E (encoding the θ-expansion of the traces in the domain of the assignment encoded by a in $\mathcal{L}$ for every $\theta \in \Gamma$) such that $\alpha_s(L, g, c, a, a', i)$ (respectively $\alpha_p(L, g, c, a, a', i)$) is satisfied in $(\mathbb{N}, +, \cdot, <, \in)$ if and only if a' encodes the i-th (Γ, C)-successor (predecessor) of the assignment encoded by a. In particular, the i-th (Γ, C)-predecessor of the assignment encoded by a is undefined if and only if the formula $\neg\exists a'.\alpha_p(L, g, c, a, a', i)$ holds in $(\mathbb{N}, +, \cdot, <, \in)$.

Now, we extend the definition of expansions to quantifier-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$. As the semantics update the assignment on which we evaluate the formula (for temporal operators) and the context (for the context operator), assignments and contexts need to be inputs to the expansion. Recall that Remark 2.2 states that satisfaction of quantifier-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas only depends on an assignment and a context, but not on a set of traces. Formally, the ψ-expansion e of a quantifier-free $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formula ψ is the function mapping an assignment Π, a context C (both over the set $\{x_1, \ldots, x_n\}$ of variables occurring in ψ), and a subformula ψ' of ψ to

$$e(\Pi, C, \psi') = \begin{cases} 1 & \text{if } (\Pi, C) \models \psi', \\ 0 & \text{if } (\Pi, C) \not\models \psi'. \end{cases}$$

The expansion is uniquely identified by the following consistency requirements, i.e., it is the only such function satisfying these requirements:

- $e(\Pi, C, p_x) = 1$ if and only if $\Pi(x) = (\sigma, i)$ and $p \in \sigma(i)$,
- $e(\Pi, C, \neg\psi) = 1$ if and only if $e(\Pi, C, \psi) = 0$,
- $e(\Pi, C, \psi_1 \vee \psi_2) = 1$ if and only if $e(\Pi, C, \psi_1) = 1$ or $e(\Pi, C, \psi_2) = 1$,
- $e(\Pi, C, \langle C' \rangle \psi) = 1$ if and only if $e(\Pi, C', \psi) = 1$,
- $e(\Pi, C, \mathbf{X}_\Gamma\, \psi) = 1$ if and only if $e(\mathrm{succ}_{(\Gamma,C)}(\Pi)), C, \psi) = 1$,
- $e(\Pi, C, \psi_1\, \mathbf{U}_\Gamma\, \psi_2) = 1$ if and only if there exists an $i \geq 0$ such that $e(\mathrm{succ}^i_{(\Gamma,C)}(\Pi), C, \psi_2) = 1$ and $e(\mathrm{succ}^j_{(\Gamma,C)}(\Pi), C, \psi_1) = 1$ for all $0 \leq j < i$,
- $e(\Pi, C, \mathbf{Y}_\Gamma\, \psi) = 1$ if and only if $\mathrm{pred}_{(\Gamma,C)}(\Pi)$ is defined and $e(\mathrm{pred}_{(\Gamma,C)}(\Pi), C, \psi) = 1$, and
- $e(\Pi, C, \psi_1\, \mathbf{S}_\Gamma\, \psi_2) = 1$ if and only if there exists an $i \geq 0$ such that $\mathrm{pred}^i_{(\Gamma,C)}(\Pi)$ is defined and $e(\mathrm{pred}^i_{(\Gamma,C)}(\Pi), C, \psi_2) = 1$ and $e(\mathrm{pred}^j_{(\Gamma,C)}(\Pi), C, \psi_1) = 1$ for all $0 \leq j < i$.

Using the previously defined α-formulas, we can again write a formula $\alpha'_e(L, p, E)$ with free variables L (encoding a set of traces), p (encoding a quantifier-free formula ψ), and E (encoding a function e mapping assignments over the set encoded by L, contexts, and

subformulas of ψ to $\{0,1\}$) such that $\alpha'_e(L,p,E)$ is satisfied in $(\mathbb{N},+,\cdot,<,\in)$ if and only of e is the ψ-expansion.

Now, we are in a position to construct the Σ^1_1-sentence $ar(x)$ with a free variable x such that $(\mathbb{N},+,\cdot,<,\in) \models ar(n)$ if and only if n encodes a $\mathrm{GHyLTL}_{\mathrm{S+C}}$ sentence φ that is satisfiable. Intuitively, we express the existence of the following type 1 objects:

- A countable set of traces over the propositions in φ encoded by a function $L\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ as described above.
- A function $S\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ interpreted as Skolem functions, i.e., S maps a variable of φ that is existentially quantified and the encoding of a trace assignment for the variables preceding it (encoded by a natural number as described above) to a trace name.
- A function $E\colon \mathbb{N}\times\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ interpreted as the ψ-expansion, i.e., it maps encodings of assignments, contexts over the variables occurring in ψ, and quantifier-free subformulas of φ (including the PLTL formulas labeling the temporal operators of φ) to $\{0,1\}$.

Then, we use only first-order quantification to express the following properties of these objects:

- The function encoded by E satisfies the consistency requirements for the ψ-expansion (using α'_e).
- For every encoding a of a variable assignment that is consistent with the Skolem function S, we have $E(a,c,p)=1$, where c encodes the context containing all variables occurring in ψ and where p encodes the maximal quantifier-free subformula of φ.

Altogether, the resulting formula $ar(x)$ has the form

$$\exists L.\ \exists S.\ \exists E.\ \exists p.\ \exists c.\ \alpha_{qf}(x,p) \wedge \alpha_{fc}(x,c) \wedge \alpha'_e(L,p,E) \wedge \forall a.\ \alpha_{asgn}(x,a) \to E(a,c,p)=1.$$

Here, we use the following auxiliary formulas:

- $\alpha_{qf}(x,p)$ holds if and only if p encodes the maximal quantifier-free subformula of the sentence encoded by x.
- $\alpha_{fc}(x,c)$ holds if and only if c encodes the context containing all variables appearing in the sentence encoded by x.
- $\alpha_{asgn}(x,a)$ holds if and only if a encodes an assignment for the sentence encoded by x.

We conclude by noting that $ar(x)$ can be constructed in polynomial time. □

As HyperLTL is a fragment of $\mathrm{HyperLTL}_{\mathrm{C}}$ and of $\mathrm{HyperLTL}_{\mathrm{S}}$, which in turn are fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$, we have also settled the complexity of their satisfiability problem as well.

Corollary 4.2. *The $\mathrm{HyperLTL}_C$ and $\mathrm{HyperLTL}_S$ satisfiability problems are both Σ^1_1-complete with respect to polynomial-time reductions.*

Thus, maybe slightly surprisingly, all four satisfiability problems have the same complexity, even though $\mathrm{GHyLTL}_{\mathrm{S+C}}$ adds stuttering, contexts, and quantification under the scope of temporal operators to HyperLTL. This result should also be compared with the HyperCTL* satisfiability problem, which is Σ^2_1-complete [FKTZ25], i.e., much harder. HyperCTL* is obtained by extending HyperLTL with just the ability to quantify under the scope of temporal operators. However, it has a branching-time semantics and trace quantification ranges over trace suffixes starting at the current position of the most recently quantified trace. This allows one to write a formula that has only uncountable models, the crucial step towards obtaining the Σ^2_1-lower bound. In comparison, $\mathrm{GHyLTL}_{\mathrm{S+C}}$ has a linear-time

semantics and trace quantification ranges over initial traces, which is not sufficient to enforce uncountable models.

As model-checking is undecidable for HyperLTL_S and $\text{GHyLTL}_{\text{S+C}}$ [BPS21, BBST24], fragments with decidable model-checking have been investigated:

- Simple HyperLTL_S requires that the maximal quantifier-free formula is a Boolean combination of HyperLTL_S formulas where all temporal operators are labeled by the same set Γ of PLTL formulas and of HyperLTL_S formulas with a single free variable [BPS21]. This fragment includes HyperLTL.
- Simple $\text{GHyLTL}_{\text{S+C}}$ extends simple HyperLTL_S with carefully restricted usage of contexts and past operators. We refer to [BBST24] for a formal definition.

Remark 4.3. As the simple fragments of $\text{GHyLTL}_{\text{S+C}}$ and HyperLTL_S both contain HyperLTL, their satisfiability problems are also Σ^1_1-hard, and thus, with the upper bound proven here, in fact Σ^1_1-complete with respect to polynomial-time reductions.

5. Model-Checking $\text{GHyLTL}_{\text{S+C}}$

In this section, we settle the complexity of the model-checking problems for $\text{GHyLTL}_{\text{S+C}}$ and its fragments: Given a sentence φ and a transition system $\mathcal{T}$, do we have $\mathcal{T} \models \varphi$? Recall that for HyperLTL, model-checking is decidable. We show here that $\text{GHyLTL}_{\text{S+C}}$ model-checking is equivalent to truth in second-order arithmetic, with the lower bounds already holding for HyperLTL_C and HyperLTL_S, i.e., adding only contexts and adding only stuttering makes HyperLTL model-checking much harder.

The proof is split into three lemmata. We begin with the upper bound for full $\text{GHyLTL}_{\text{S+C}}$. Here, in comparison to the upper bound for satisfiability, we have to work with possibly uncountable models, as the transition system may have uncountably many traces. The proof of the upper bound encodes the semantics of $\text{GHyLTL}_{\text{S+C}}$ over (possibly) uncountable sets of traces in arithmetic.

Lemma 5.1. *GHyLTL_{S+C} model-checking is polynomial-time reducible to truth in second-order arithmetic.*

Proof. We present a polynomial-time translation from pairs $(\mathcal{T}, \varphi)$ of finite transition systems and $\text{GHyLTL}_{\text{S+C}}$ sentences φ to sentences φ' of second-order arithmetic such that $\mathcal{T} \models \varphi$ if and only if $(\mathbb{N}, +, \cdot, <, \in) \models \varphi'$. Intuitively, we will capture the semantics of $\text{GHyLTL}_{\text{S+C}}$ in arithmetic as we have done in the proof of Theorem 4.1. However, dealing with possibly uncountable sets of traces requires a different encoding of traces.

Let us fix a $\text{GHyLTL}_{\text{S+C}}$ sentence φ and a transition system $\mathcal{T}$ and let AP be the propositions occurring in φ or $\mathcal{T}$. To encode traces over AP, we fix a bijection $h_{prop} \colon \text{AP} \to \{0, 1, \ldots, |\text{AP}|-1\}$ and use Cantor's pairing function $pair \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined as $pair(i, j) = \frac{1}{2}(i + j)(i + j + 1) + j$, which is a bijection that is implementable in first-order arithmetic. Thus, we can encode a trace $\sigma \in (2^{\text{AP}})^\omega$ by the set

$$S_\sigma = \{pair(j, h_{prop}(p)) \mid j \in \mathbb{N} \text{ and } p \in \sigma(j)\} \subseteq \mathbb{N}.$$

Note that $\sigma \neq \sigma'$ implies $S_\sigma \neq S_{\sigma'}$. Furthermore, there is a formula $\alpha_\mathcal{T}(X)$ of second-order arithmetic with a single free second-order variable X such that $(\mathbb{N}, +, \cdot, <, \in) \models \alpha_\mathcal{T}(X)$ if and only if X encodes a trace of $\mathcal{T}$ [FRZ26, Proof of Theorem 4.4].

Next, we encode trace assignments. To this end, let $\mathsf{VAR}' \subseteq \mathsf{VAR}$ be the set of variables appearing in φ. We fix some bijection $h_{var} \colon \mathsf{VAR}' \to \{0, 1, \dots, |\mathsf{VAR}'| - 1\}$. In the following, we restrict ourselves to assignments whose domains are subsets of VAR'. Let Π be such an assignment, i.e., $\Pi(x_j)$ is either undefined or of the form (σ, i), i.e., a pair containing a trace and a position. We encode Π by the set

$$S_\Pi = \bigcup_{x \in \mathrm{Dom}(\Pi)} \{pair(h_{var}(x), n) \mid \Pi(x) = (\sigma, i) \text{ and } n \in S_\sigma\} \cup \{pair(|\mathsf{VAR}'| + h_{var}(x), i) \mid \Pi(x) = (\sigma, i)\} \subseteq \mathbb{N}.$$

Note that $\Pi \neq \Pi'$ implies $S_\Pi \neq S_{\Pi'}$. Using this encoding, one can write the following formulas of second-order arithmetic:

- $\alpha_{lo}(A, p, i)$ with free variables A (encoding an assignment Π), and p and i such that $(\mathbb{N}, +, \cdot, <, \in) \models \alpha_{lo}(A, p, i)$ if and only if $h^{-1}_{prop}(p) \in \sigma(j)$, where $\Pi(h^{-1}_{var}(i)) = (\sigma, j)$, i.e., α_{lo} looks up whether the proposition encoded by p holds at the pointed trace assigned to the variable i by Π.
- $\alpha_{up}(A, A', X, i)$ with free variables A and A' (encoding two assignments Π and Π'), X (encoding a trace σ), and i such that $(\mathbb{N}, +, \cdot, <, \in) \models \alpha_{up}(A, A', X, i)$ if and only if $\Pi' = \Pi[x \mapsto (\sigma, 0)]$, where x is the variable with $h_{var}(x) = i$, i.e., α_{up} updates assignments.
- $\alpha^s_{(\Gamma,C)}(A, A', i)$, for each finite set Γ of PLTL formulas and each context $C \subseteq \mathsf{VAR}'$, with free variables A and A' (encoding two assignments Π and Π') and i such that $(\mathbb{N}, +, \cdot, <, \in) \models \alpha^s_{(\Gamma,C)}(A, A', i)$ if and only if Π' is the i-th (Γ, C)-successor of Π.
- $\alpha^p_{(\Gamma,C)}(A, A', i)$, for each finite set Γ of PLTL formulas and each context $C \subseteq \mathsf{VAR}'$, with free variables A and A' (encoding two assignments Π and Π') and i such that $(\mathbb{N}, +, \cdot, <, \in) \models \alpha^p_{(\Gamma,C)}(A, A', i)$ if and only if Π' is the i-th (Γ, C)-predecessor of Π (which in particular implies that Π' is defined).

The latter two formulas rely on the concept of expansions, as introduced in the proof of Theorem 4.1, to capture Γ-changepoints.

Now, we present the inductive translation of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ into second-order arithmetic. To not clutter our notation even further, we do not add $\mathcal{T}$ as an input to our translation function, but use it in its definition. For every context $D \subseteq \mathsf{VAR}'$, we define a translation function ar_D mapping $\mathrm{GHyLTL}_{\mathrm{S+C}}$ formulas ψ to formulas $ar_D(\psi)$ of second-order arithmetic such that each $ar_D(\psi)$ has a unique free second-order variable encoding an assignment:

- $ar_D(p_x) = \alpha_{lo}(A, h_{prop}(p), h_{var}(x))$, i.e., A is the (only) free variable of $ar_D(p_x)$, as $h_{prop}(p)$ and $h_{var}(x)$ are constants, and thus definable in first-order arithmetic.
- $ar_D(\neg\psi) = \neg ar_D(\psi)$, i.e., the free variable of $ar_D(\neg\psi)$ is the free variable of $ar_D(\psi)$.
- $ar_D(\psi_1 \vee \psi_2) = ar_D(\psi_1) \vee ar_D(\psi_2)$ where we assume w.lo.g. that the free variables of $ar_D(\psi_1)$ and $ar_D(\psi_2)$ are the same, which is then also the free variable of $ar_D(\psi_1 \vee \psi_2)$.
- $ar_D(\langle C \rangle \psi) = ar_C(\psi)$, i.e., the free variable of $ar_D(\langle C \rangle \psi)$ is the free variable of $ar_C(\psi)$.
- $ar_D(\mathbf{X}_\Gamma \psi) = \exists A'(\alpha^s_{(\Gamma,D)}(A, A', 1) \wedge ar_D(\psi))$ where A' is the free variable of $ar_D(\psi)$ and A is the free variable of $ar_D(\mathbf{X}_\Gamma \psi)$.
- $ar_D(\psi_1 \,\mathbf{U}_\Gamma\, \psi_2) = \exists i\, \exists A_2(\alpha^s_{(\Gamma,D)}(A, A_2, i) \wedge ar_D(\psi_2) \wedge \forall j(j < i \to \exists A_1(\alpha^s_{(\Gamma,D)}(A, A_1, j) \wedge ar_D(\psi_1))))$ where A_1 is the free variable of $ar_D(\psi_1)$, A_2 is the free variable of $ar_D(\psi_2)$, and A is the free variable of $ar_D(\psi_1 \,\mathbf{U}_\Gamma\, \psi_2)$.
- $ar_D(\mathbf{Y}_\Gamma \psi) = \exists A'(\alpha^p_{(\Gamma,D)}(A, A', 1) \wedge ar_D(\psi))$ where A' is the free variable of $ar_D(\psi)$ and A is the free variable of $ar_D(\mathbf{Y}_\Gamma \psi)$.

- $ar_D(\psi_1 \mathbf{S}_\Gamma \psi_2) = \exists i \exists A_2(\alpha^p_{(\Gamma,D)}(A, A_2, i) \wedge ar_D(\psi_2) \wedge \forall j(j < i \rightarrow \exists A_1(\alpha^p_{(\Gamma,D)}(A, A_1, j) \wedge ar_D(\psi_1))))$ where A_1 is the free variable of $ar_D(\psi_1)$, A_2 is the free variable of $ar_D(\psi_2)$, and A is the free variable of $ar_D(\psi_1 \mathbf{S}_\Gamma \psi_2)$.
- $ar_D(\exists x.\psi) = \exists X(\alpha_{\mathcal{T}}(X) \wedge \exists A'(\alpha_{up}(A, A', X, h_{var}(x)) \wedge ar_D(\psi)))$ where A' is the free variable of $ar_D(\psi)$ and A is the free variable of $ar_D(\exists x.\psi)$.
- $ar_D(\forall x.\psi) = \forall X(\alpha_{\mathcal{T}}(X) \rightarrow \exists A'(\alpha_{up}(A, A', X, h_{var}(x)) \wedge ar_D(\psi)))$ where A' is the free variable of $ar_D(\psi)$ and A is the free variable of $ar_D(\forall x.\psi)$.

Given a $\text{GHyLTL}_{\text{S+C}}$ sentence φ with variables VAR', we define $ar(\varphi) = ar_{\mathsf{VAR}'}(\varphi)(\emptyset)$, i.e., we interpret the free variable of $ar_{\mathsf{VAR}'}(\varphi)$ with the empty set, which encodes the assignment with empty domain. Then, an induction shows that we have $\mathcal{T} \models \varphi$ if and only if $(\mathbb{N}, +, \cdot, <, \in) \models ar(\varphi)$, where $\mathcal{T}$ is the fixed transition system that is used in translation of the quantifiers. □

Next, we prove the matching lower bounds for the two fragments $\text{HyperLTL}_{\text{S}}$ and $\text{HyperLTL}_{\text{C}}$ of $\text{GHyLTL}_{\text{S+C}}$. This shows that already stuttering alone and contexts alone reach the full complexity of $\text{GHyLTL}_{\text{S+C}}$ model-checking.

We proceed as follows: traces over a single proposition # encode sets of natural numbers, and thus also natural numbers (via singleton sets). Hence, trace quantification in a transition system that has all traces over $\{\#\}$ can mimic first- and second-order quantification in arithmetic. The main missing piece is thus the implementation of addition and multiplication using only stuttering and using only contexts. In the following, we present such implementations, thereby showing that one can embed second-order arithmetic in both $\text{HyperLTL}_{\text{S}}$ and $\text{HyperLTL}_{\text{C}}$. To implement multiplication, we need to work with traces of the form $\sigma = \{\$\}^{m_0}\emptyset^{m_1}\{\$\}^{m_2}\emptyset^{m_3}\cdots$ for some auxiliary proposition \$. We call a maximal infix of the form $\{\$\}^{m_j}$ or $\emptyset^{m_j}$ a block of σ. If we have $m_0 = m_1 = m_2 = \cdots$, then we say that σ is periodic and call m_0 the period of σ.

Lemma 5.2. *Truth in second-order arithmetic is polynomial-time reducible to HyperLTL$_S$ model checking.*

Proof. We present a polynomial-time translation mapping sentences φ of second-order arithmetic to pairs $(\mathcal{T}, hyp(\varphi))$ of transition systems $\mathcal{T}$ and $\text{HyperLTL}_{\text{S}}$ sentences $hyp(\varphi)$ such that $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models hyp(\varphi)$. Intuitively, we capture the semantics of arithmetic in $\text{HyperLTL}_{\text{S}}$.

We begin by formalizing our encoding of natural numbers and sets of natural numbers using traces. Intuitively, a trace σ over a set AP of propositions containing the proposition # encodes the set $\{n \in \mathbb{N} \mid \# \in \sigma(n)\} \subseteq \mathbb{N}$. In particular, a trace σ encodes a singleton set if it satisfies the formula $(\neg\#)\,\mathbf{U}(\# \wedge \mathbf{X}\,\mathbf{G}\,\neg\#)$. In the following, we use the encoding of singleton sets to encode natural numbers as well. Obviously, every set and every natural number is encoded by a trace in that manner. Thus, we can mimic first- and second-order quantification by quantification over traces.

However, to implement addition and multiplication using only stuttering, we need to adapt this simple encoding. We need a unique proposition for each first-order variable in φ. So, let us fix a sentence φ of second-order arithmetic and let V_1 be the set of first-order variables appearing in φ. We use the set $\text{AP} = \{\#\} \cup \{\#(y) \mid y \in V_1\} \cup \{\$, \$'\}$ of propositions, where \$ and \$′ are auxiliary propositions used to implement multiplication.

We define the function *hyp* mapping second-order formulas to $\text{HyperLTL}_{\text{S}}$ formulas:

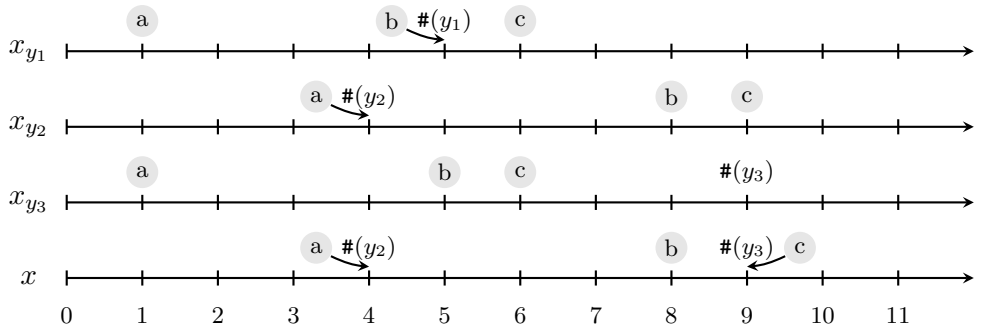


Figure 3: The formula α_{add} implements addition, illustrated here for $n_1 = 5$ and $n_2 = 4$.

- $hyp(\exists Y.\psi) = \exists x_Y. \left(\mathbf{G}_\emptyset \bigwedge_{p \neq \#} \neg p_{x_Y} \right) \wedge hyp(\psi)$.
- $hyp(\forall Y.\psi) = \forall x_Y. \left(\mathbf{G}_\emptyset \bigwedge_{p \neq \#} \neg p_{x_Y} \right) \rightarrow hyp(\psi)$.
- $hyp(\exists y.\psi) = \exists x_y. \big(\mathbf{G}_\emptyset \bigwedge_{p \neq \#(y)} \neg p_{x_y} \big) \wedge$
  $\big((\neg(\#(y))_{x_y}) \, \mathbf{U}_\emptyset ((\#(y))_{x_y} \wedge \mathbf{X}_\emptyset \, \mathbf{G}_\emptyset \, \neg(\#(y))_{x_y}) \big) \wedge hyp(\psi)$.
- $hyp(\forall y.\psi) = \forall x_y. \Big[\big(\mathbf{G}_\emptyset \bigwedge_{p \neq \#(y)} \neg p_{x_y} \big) \wedge \big((\neg(\#(y))_{x_y}) \, \mathbf{U}_\emptyset ((\#(y))_{x_y} \wedge \mathbf{X}_\emptyset \, \mathbf{G}_\emptyset \, \neg(\#(y))_{x_y}) \big) \Big] \rightarrow$
  $hyp(\psi)$.
- $hyp(\neg\psi) = \neg hyp(\psi)$.
- $hyp(\psi_1 \vee \psi_2) = hyp(\psi_1) \vee hyp(\psi_2)$.
- $hyp(y \in Y) = \mathbf{F}_\emptyset ((\#(y))_{x_y} \wedge \#_{x_Y})$.
- $hyp(y_1 < y_2) = \mathbf{F}_\emptyset ((\#(y_1))_{x_{y_1}} \wedge \mathbf{X}_\emptyset \, \mathbf{F}_\emptyset (\#(y_2))_{x_{y_2}})$.

At this point, it remains to implement addition and multiplication in HyperLTL$_\text{S}$. Let Π be an assignment that maps each x_{y_j} for $j \in \{1, 2, 3\}$ to a pointed trace $(\sigma_j, 0)$ for some σ_j of the form

$$\emptyset^{n_j} \{\#(y_j)\} \emptyset^\omega \tag{$\dagger$}$$

which is the form of traces that trace variables x_{y_j} encoding first-order variables y_j of φ range over. Our goal is to write formulas $hyp(y_1 + y_2 = y_3)$ and $hyp(y_1 \cdot y_2 = y_3)$ with free variables $x_{y_1}, x_{y_2}, x_{y_3}$ that are satisfied by Π if and only if $n_1 + n_2 = n_3$ and $n_1 \cdot n_2 = n_3$, respectively.

We begin with addition. We assume that n_1 and n_2 are both non-zero and handle the special cases $n_1 = 0$ and $n_2 = 0$ later separately. Consider the formula

$$\alpha_{\text{add}} = \exists x. \left[\psi \wedge \mathbf{X}_{\#(y_2)} \, \mathbf{F}_\emptyset ((\#(y_1))_{x_{y_1}} \wedge \mathbf{X}_\emptyset (\#(y_3))_x) \right],$$

where x is a fresh variable and where

$$\psi = \mathbf{G}_\emptyset [(\#(y_2))_{x_{y_2}} \leftrightarrow (\#(y_2))_x] \wedge \mathbf{G}_\emptyset [(\#(y_3))_{x_{y_3}} \leftrightarrow (\#(y_3))_x]$$

holds if the truth values of $\#y_2$ coincide on $\Pi(y_2)$ and $\Pi(x)$ and those of $\#y_3$ coincide on $\Pi(y_3)$ and $\Pi(x)$, respectively (see Figure 3).

Now, consider an assignment Π satisfying ψ and ($\dagger$). The outer next operator in α_{add} (labeled by $\#(y_2)$) updates the pointers in Π to the ones marked by "a". For the traces

assigned to x_{y_1} and x_{y_3} this is due to the fact that both traces do not contain $\#(y_2)$, which implies that every position is a $\#(y_2)$-changepoint in these traces. On the other hand, both the traces assigned to x_{y_2} and x contain a $\#(y_2)$. Hence, the pointers are updated to the first position where $\#(y_2)$ holds (here, we use $n_2 > 0$).

Next, the eventually operator (labeled by $\emptyset$) updates the pointers in Π to the ones marked by "b", as $\#(y_1)$ has to hold on $\Pi(x_{y_1})$. The distance between the positions marked "a" and "b" here is exactly $n_1 - 1$ (here, we use $n_1 > 0$) and is applied to all pointers, as each position on each trace is a $\emptyset$-changepoint.

Due to the same argument, the inner next operator (labeled by $\emptyset$) updates the pointers in Π to the ones marked by "c". In particular, on the trace $\Pi(x)$, this is position $n_1 + n_2$. As we require $\#(y_3)$ to hold there (and thus also at the same position on $\Pi(x_{y_3})$, due to ψ), we have indeed expressed $n_1 + n_2 = n_3$.

Accounting for the special cases $n_1 = 0$ (first line) and $n_2 = 0$ (second line), we define

$$\begin{aligned} hyp(y_1 + y_2 = y_3) = {} & \left[(\#(y_1))_{x_{y_1}} \wedge \mathbf{F}_\emptyset((\#(y_2))_{x_{y_2}} \wedge (\#(y_3))_{x_{y_3}})\right] \vee \\ & \left[(\#(y_2))_{x_{y_2}} \wedge \mathbf{F}_\emptyset((\#(y_1))_{x_{y_1}} \wedge (\#(y_3))_{x_{y_3}})\right] \vee \\ & \left[\neg(\#(y_1))_{x_{y_1}} \wedge \neg(\#(y_2))_{x_{y_2}} \wedge \alpha_{\text{add}}\right]. \end{aligned}$$

So, it remains to implement multiplication. Consider an assignment Π satisfying (†). We again assume n_1 and n_2 to be non-zero and handle the special cases $n_1 = 0$ and $n_2 = 0$ later. The formula

$$\begin{aligned} \alpha_1 = \$_x \wedge \mathbf{G}_\emptyset\, \mathbf{F}_\emptyset\, \$_x \wedge \mathbf{G}_\emptyset\, \mathbf{F}_\emptyset\, \neg\$_x \wedge \mathbf{G}_\emptyset \bigwedge_{p \neq \$, \#(y_3)} \neg p_x \wedge \\ \$'_{x'} \wedge \mathbf{G}_\emptyset\, \mathbf{F}_\emptyset\, \$'_{x'} \wedge \mathbf{G}_\emptyset\, \mathbf{F}_\emptyset\, \neg\$'_{x'} \wedge \mathbf{G}_\emptyset \bigwedge_{p \neq \$'} \neg p_{x'} \end{aligned}$$

with two (fresh) free variables x and x' expresses that if $\Pi(x) = (\sigma, i)$ satisfies $i = 0$, then σ is of the form $\sigma_{y_3} \cup \{\$\}^{m_0}\emptyset^{m_1}\{\$\}^{m_2}\emptyset^{m_3}\cdots$, for $\sigma_{y_3} \in (2^{\{\#(y_3)\}})^\omega$ and non-zero m_j, and if $\Pi(x') = (\sigma', i')$ satisfies $i' = 0$, then σ' is of the form $\{\$'\}^{m'_0}\emptyset^{m'_1}\{\$'\}^{m'_2}\emptyset^{m'_3}\cdots$, for non-zero m'_j. Here, $\cup$ denotes the pointwise union of two traces. The part x_{y_3} will only become relevant later, so we ignore it for the time being.

Then, the formula $\alpha_2 = \mathbf{G}_\emptyset(\$_x \leftrightarrow \$'_{x'})$ is satisfied by Π if and only if $m_j = m'_j$ for all j. If $\alpha_1 \wedge \alpha_2$ is satisfied, then the $\{\$, \$'\}$-changepoints on σ *and* σ' are $0, m_0, m_0 + m_1, m_0 + m_1 + m_2, \ldots$. Now, consider

$$\begin{aligned} \alpha_3 = \mathbf{G}_{\{\$,\$'\}} \Big[& \left[\$_x \rightarrow \mathbf{X}_{\$'}\left((\$_x \wedge \neg\$'_{x'})\, \mathbf{U}_\emptyset(\neg\$_x \wedge \neg\$'_{x'} \wedge \mathbf{X}_\emptyset\, \$'_{x'})\right)\right] \wedge \\ & \left[\neg\$_x \rightarrow \mathbf{X}_{\$'}\left((\neg\$_x \wedge \$'_{x'})\, \mathbf{U}_\emptyset(\$_x \wedge \$'_{x'} \wedge \mathbf{X}_\emptyset\, \neg\$'_{x'})\right)\right] \Big] \end{aligned}$$

and a trace assignment Π satisfying $\alpha_1 \wedge \alpha_2 \wedge \alpha_3$ and such that the pointers of $\Pi(x)$ and $\Pi(x')$ are both 0. Then, the always operator of α_3 updates both pointers of $\Pi(x)$ and $\Pi(x')$ to some $\{\$, \$'\}$-changepoint as argued above (see the positions marked "a" in Figure 4). This is the beginning of an infix of the form $\{\$\}^{m_j}$ in x and of the form $\{\$'\}^{m_j}$ in x' or the beginning of an infix of the form $\emptyset^{m_j}$ in x and in x'.

Let us assume we are in the former case, the latter is dual. Then, the premise of the upper implication of α_3 holds, i.e., $\mathbf{X}_{\$'}((\$_x \wedge \neg\$'_{x'})\, \mathbf{U}_\emptyset(\neg\$_x \wedge \neg\$'_{x'} \wedge \mathbf{X}_\emptyset\, \$'_{x'}))$ must be satisfied as well. The next operator (labeled by $\{\$'\}$ only) increments the pointer of $\Pi(x)$ by one (as σ does not contain any $\$'$) and updates the one of $\Pi(x')$ to the position after the infix $\{\$'\}^{m_j}$

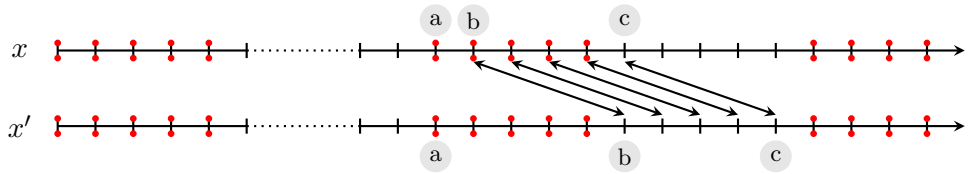


Figure 4: The formula α_3 ensures that the traces assigned to x and x' are periodic. Here, "$\vdots$" denotes a position where \$ (\$′) holds and "$\shortmid$" a position where \$ (\$′) does not hold.

in σ' (see the pointers marked with "b"). Now, the until formula only holds if we have $m_j = m_{j+1}$, as it compares the positions marked by the diagonal lines until it "reaches" the positions marked with "c". As the reasoning holds for every j we conclude that we have $m_0 = m_1 = m_2 = \cdots$, i.e., we have constructed a periodic trace with period m_0 that will allow us to implement multiplication by m_0. From here on, we ignore $\Pi(x')$, as it is only needed to construct $\Pi(x)$.

Next, we relate the trace $\Pi(x)$ to the traces x_{y_j} encoding the numbers we want to multiply using

$$\begin{aligned}\alpha_{\text{mult}} = \exists x.\exists x'.\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge (\$_x \,\mathbf{U}_\emptyset(\neg\$_x \wedge (\#(y_1))_{x_{y_1}}))\wedge \\ \mathbf{G}_\emptyset((\#(y_3))_{x_{y_3}} \leftrightarrow (\#(y_3))_x) \wedge \mathbf{F}_\$[(\#(y_2))_{x_{y_2}} \wedge (\#(y_3))_x],\end{aligned}$$

which additionally expresses that m_0 is equal to n_1, that $\#(y_3)$ holds on $\Pi(x)$ at position n_3 as well (and nowhere else), and finally that n_3 is equal to $n_1 \cdot n_2$: This follows from the fact that we stutter n_2 times on x_{y_2} to reach the position where $\#(y_2)$ holds (as every position is a \$-changepoint on σ_2). Thus, we reach position $m_0 \cdot n_2 = n_1 \cdot n_2$ on $\Pi(x)$, at which $\#(y_3)$ must hold. As $\#(y_3)$ holds at the same position on x_{y_3}, we have indeed captured $n_1 \cdot n_2 = n_3$.

So, taking the special cases $n_1 = 0$ and $n_2 = 0$ (in the first two disjuncts) into account, we define

$$\begin{aligned}hyp(y_1 \cdot y_2 = y_3) = [(\#(y_1))_{x_{y_1}} \wedge (\#(y_3))_{x_{y_3}}]\vee \\ [(\#(y_2))_{x_{y_2}} \wedge (\#(y_3))_{x_{y_3}}]\vee \\ [\neg(\#(y_1))_{x_{y_1}} \wedge \neg(\#(y_2))_{x_{y_2}} \wedge \alpha_{\text{mult}}].\end{aligned}$$

Now, let $\mathcal{T}$ be a transition system[2] with

$$\text{Tr}(\mathcal{T}) = (2^{\{\#\}})^\omega \cup \bigcup_{y,y' \in V_1} (2^{\{\#(y),\#(y'),\$\}})^\omega \cup (2^{\{\$'\}})^\omega,$$

where V_1 still denotes the set of first-order variables in the sentence φ of second-order arithmetic. Here, $(2^{\{\#\}})^\omega$ contains the traces to mimic set quantification, $\bigcup_{y,y' \in V_1}(2^{\{\#(y),\#(y'),\$\}})^\omega$ contains the traces to mimic first-order quantification and for the variable x used in the definition of multiplication, and $(2^{\{\$'\}})^\omega$ contains the traces for x', also used in the definition of multiplication. Such a $\mathcal{T}$ can, given φ, be constructed in polynomial time. It has several connected components, all fully connected (including self-loops at all vertices), and all vertices are initial.

[2]Note that this fixes an error in the conference version [RZ25] where we used a transition system with too few traces.

- The first component has two vertices, one labeled with the empty set, the other with $\{\#\}$. The traces of this component are the traces in $(2^{\{\#\}})^\omega$.
- For every $y, y' \in V_1$, there is a component with eight vertices, one for each subset S of $2^{\{\#(y),\#(y'),\$\}}$ that is labeled by S. The traces of this component are the traces in $(2^{\{\#(y),\#(y'),\$\}})^\omega$.
- The last component has two vertices, one labeled with the empty set, the other with $\{\$'\}$. The traces of this component are the traces in $(2^{\{\$'\}})^\omega$.

An induction shows that we have $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models \mathit{hyp}(\varphi)$. Note that while $\mathit{hyp}(\varphi)$ is not necessarily in prenex normal form, it can be brought into that as no quantifier is under the scope of a temporal operator, i.e., into a HyperLTL$_\mathrm{S}$ formula. □

Next, we consider the lower-bound for the second fragment, i.e., HyperLTL with contexts.

Lemma 5.3. *Truth in second-order arithmetic is polynomial-time reducible to HyperLTL$_C$ model checking.*

Proof. We present a polynomial-time translation mapping sentences φ of second-order arithmetic to pairs $(\mathcal{T}, \mathit{hyp}(\varphi))$ of transition systems $\mathcal{T}$ and HyperLTL$_\mathrm{C}$ sentences $\mathit{hyp}(\varphi)$ such that $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models \mathit{hyp}(\varphi)$. Intuitively, we will capture the semantics of arithmetic in HyperLTL$_\mathrm{C}$. As the temporal operators in HyperLTL$_\mathrm{C}$ are all labeled by the empty set, we simplify our notation by dropping them in this proof, i.e., we just write $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{U}$.

As in the proof of Lemma 5.2, we encode natural numbers and sets of natural numbers by traces. Here, it suffices to consider a single proposition $\#$ to encode these and an additional proposition $\$$ that we use to implement multiplication, i.e., our HyperLTL$_\mathrm{C}$ formulas are built over $\mathrm{AP} = \{\#, \$\}$.

We again define a function hyp mapping second-order formulas to HyperLTL$_\mathrm{C}$ formulas:

- $\mathit{hyp}(\exists Y.\psi) = \exists x_Y.\, (\mathbf{G}\, \neg \$_{x_Y}) \wedge \mathit{hyp}(\psi)$.
- $\mathit{hyp}(\forall Y.\psi) = \forall x_Y.\, (\mathbf{G}\, \neg \$_{x_Y}) \rightarrow \mathit{hyp}(\psi)$.
- $\mathit{hyp}(\exists y.\psi) = \exists x_y.\, \big(\mathbf{G}\, \neg \$_{x_y}\big) \wedge \big((\neg \#_{x_y})\, \mathbf{U} (\#_{x_y} \wedge \mathbf{X}\, \mathbf{G}\, \neg \#_{x_y})\big) \wedge \mathit{hyp}(\psi)$.
- $\mathit{hyp}(\forall y.\psi) = \forall x_y.\, \big[\big(\mathbf{G}\, \neg \$_{x_y}\big) \wedge \big((\neg \#_{x_y})\, \mathbf{U} (\#_{x_y} \wedge \mathbf{X}\, \mathbf{G}\, \neg \#_{x_y})\big)\big] \rightarrow \mathit{hyp}(\psi)$.
- $\mathit{hyp}(\neg\psi) = \neg \mathit{hyp}(\psi)$.
- $\mathit{hyp}(\psi_1 \vee \psi_2) = \mathit{hyp}(\psi_1) \vee \mathit{hyp}(\psi_2)$.
- $\mathit{hyp}(y \in Y) = \mathbf{F}(\#_{x_y} \wedge \#_{x_Y})$.
- $\mathit{hyp}(y_1 < y_2) = \mathbf{F}(\#_{x_{y_1}} \wedge \mathbf{X}\, \mathbf{F}\, \#_{x_{y_2}})$.

At this point, it remains to consider addition and multiplication. Let Π be an assignment that maps each x_{y_j} for $j \in \{1, 2, 3\}$ to a pointed trace $(\sigma_j, 0)$ for some σ_j of the form $\emptyset^{n_j}\{\#\}\emptyset^\omega$, which is the form of traces that the x_{y_j} encoding first-order variables y_j of φ range over. Our goal is to write formulas $\mathit{hyp}(y_1{+}y_2 = y_3)$ and $\mathit{hyp}(y_1{\cdot}y_2 = y_3)$ with free variables $x_{y_1}, x_{y_2}, x_{y_3}$ that are satisfied by (Π, VAR) if and only if $n_1 + n_2 = n_3$ and $n_1 \cdot n_2 = n_3$, respectively.

The case of addition is readily implementable in HyperLTL$_\mathrm{C}$ by defining

$$\mathit{hyp}(y_1 + y_2 = y_3) = \langle x_{y_1}, x_{y_3}\rangle\, \mathbf{F}\, \big[\#_{x_{y_1}} \wedge \langle x_{y_2}, x_{y_3}\rangle\, \mathbf{F}(\#_{x_{y_2}} \wedge \#_{x_{y_3}})\big]\,.$$

The first eventually updates the pointers of $\Pi(x_{y_1})$ and $\Pi(x_{y_3})$ by adding n_1 and the second eventually updates the pointers of $\Pi(x_{y_2})$ and $\Pi(x_{y_3})$ by adding n_2. At that position, $\#$ must hold on x_{y_3}, which implies that we have $n_1 + n_2 = n_3$.

At this point, it remains to implement multiplication, which is more involved than addition. In fact, we need to consider four different cases. If $n_1 = 0$ or $n_2 = 0$, then

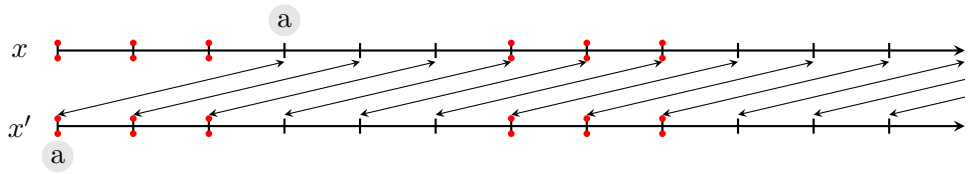


Figure 5: The formula α_3 ensures that the traces assigned to x (and x') are periodic. Here, "!" ("ı") denotes a position where \$ holds (does not hold).

we must have $n_3 = 0$ as well. This is captured by the formula $\psi_1 = (\#_{x_{y_1}} \vee \#_{x_{y_2}}) \wedge \#_{x_{y_3}}$. Further, if $n_1 = n_2 = 1$, then we must have $n_3 = 1$ as well. This is captured by the formula $\psi_2 = \mathbf{X}(\#_{x_{y_1}} \wedge \#_{x_{y_2}} \wedge \#_{x_{y_3}})$.

Next, let us consider the case $0 < n_1 \leq n_2$ with $n_2 \geq 2$. Let $z \in \mathbb{N} \setminus \{0\}$ be minimal with

$$z \cdot (n_2 - 1) = z' \cdot n_2 - n_1 \tag{5.1}$$

for some $z' \in \mathbb{N} \setminus \{0\}$. It is easy to check that $z = n_1$ is a solution of Equation (5.1) for $z' = n_1$. Now, consider some $0 < z < n_1$ to prove that n_1 is the minimal solution: Rearranging Equation 5.1 yields $-z + n_1 = (z' - z) \cdot n_2$, i.e., $-z + n_1$ must be a multiple of n_2 (possibly 0). But $0 < z < n_1$ implies $0 < n_1 - z < n_1 \leq n_2$, i.e., $-z + n_1$ is not a multiple of n_2. Hence, $z = n_1$ is indeed the smallest solution of Equation (5.1). So, for the minimal such z we have $z \cdot (n_2 - 1) + n_1 = n_1 \cdot n_2$, i.e., we have expressed multiplication of n_1 and n_2.

Let us show how to implement this argument in HyperLTL$_\mathrm{C}$. We begin by constructing two periodic traces with periods n_2 and $n_2 - 1$ using contexts. Consider the formula

$$\alpha_1 = (\$_x \wedge \mathbf{G}\,\mathbf{F}\,\$_x \wedge \mathbf{G}\,\mathbf{F}\,\neg\$_x \wedge \mathbf{G}\,\neg\#_x) \wedge (\$_{x'} \wedge \mathbf{G}\,\mathbf{F}\,\$_{x'} \wedge \mathbf{G}\,\mathbf{F}\,\neg\$_{x'} \wedge \mathbf{G}\,\neg\#_{x'})$$

with free variables x and x'. If Π is an assignment satisfying α_1 such that the pointers of $\Pi(x)$ and $\Pi(x')$ are both 0, then $\Pi(x) = (\{\$\}^{m_0}\emptyset^{m_1}\{\$\}^{m_2}\emptyset^{m_3}\cdots, 0)$ and $\Pi(x') = (\{\$\}^{m'_0}\emptyset^{m'_1}\{\$\}^{m'_2}\emptyset^{m'_3}\cdots, 0)$ for some $m_j, m'_j \geq 1$. Further, the formula $\alpha_2 = \mathbf{G}(\$_x \leftrightarrow \$_{x'})$ then expresses that $m_j = m'_j$ for all j, and the formula

$$\alpha_3 = \langle x \rangle \left(\$_x \,\mathbf{U}\, \left(\neg\$_x \wedge \langle x, x' \rangle\, \mathbf{G}(\$_x \leftrightarrow \neg\$_{x'})\right)\right)$$

expresses that $m'_j = m_{j+1}$ for all j: The until operator updates the pointers of $\Pi(x)$ and $\Pi(x')$ to the positions marked "a" in Figure 5 and the always operator then compares all following positions as depicted by the diagonal arrows. Thus, we have $m_j = m_0$ for all j if $\Pi(x)$ and $\Pi(x')$ satisfy $\alpha_{\text{per}} = \alpha_1 \wedge \alpha_2 \wedge \alpha_3$.

To conclude, consider the formula

$$\begin{aligned}\psi_3 = \Big[\mathbf{X}\,\mathbf{F}(\#_{x_{y_1}} \wedge \mathbf{F}\,\#_{x_{y_2}}) \wedge \mathbf{X}\,\mathbf{X}\,\mathbf{F}\,\#_{x_{y_2}}\Big] \wedge \\ \Big[\exists x_0.\exists x'_0.\exists x_1.\exists x'_1.\alpha_{\text{per}}[x/x_0, x'/x'_0] \wedge \alpha_{\text{per}}[x/x_1, x'/x'_1] \wedge \\ \left(\$_{x_0}\,\mathbf{U}(\neg\$_{x_0} \wedge \#_{x_{y_2}})\right) \wedge \left(\$_{x_1}\,\mathbf{U}(\neg\$_{x_1} \wedge \mathbf{X}\,\#_{x_{y_2}})\right) \wedge \\ \langle x_{y_1}, x_{y_3}, x_0\rangle \left(\mathbf{F}\left(\#_{x_{y_1}} \wedge \langle x_{y_3}, x_0, x_1\rangle(\neg\alpha_{\text{algn}})\,\mathbf{U}(\alpha_{\text{algn}} \wedge \mathbf{X}\,\#_{x_{y_3}})\right)\right)\Big]\end{aligned}$$

where $\alpha_{\text{algn}} = (\$_{x_0} \leftrightarrow \neg\,\mathbf{X}\,\$_{x_0}) \wedge (\$_{x_1} \leftrightarrow \neg\,\mathbf{X}\,\$_{x_1})$ holds at $\Pi(x_0)$ and $\Pi(x_1)$ if the pointers both point to the end of a block on the respective trace. Furthermore, $\alpha_{\text{per}}[x/x_j, x'/x'_j]$ denotes the formula obtained from replacing each occurrence of x by x_j and every occurrence

of x' by x'_j. Thus, we quantify two traces $\Pi(x_0) = (\sigma_0, 0)$ and $\Pi(x_1) = (\sigma_1, 0)$ (we disregard $\Pi(x'_0)$ and $\Pi(x'_1)$, as we just need them to construct the former two traces) that are periodic.

The third line of ψ_3 is satisfied if σ_0 has period n_2 and σ_1 has period $n_2 - 1$. Now, consider the last line of ψ_3 and see Figure 6:

- The first eventually-operator updates the pointers of $\Pi(x_{y_1})$, $\Pi(x_{y_3})$, and $\Pi(x_0)$ to n_1, as this is the unique position on $\Pi(x_{y_1})$ that satisfies #. Crucially, the pointer of $\Pi(x_1)$ is not updated, as it is not in the current context. These positions are marked by "a".
- Then, the until-operator compares positions i on x_1 and $i + n_1$ on x_0 (depicted by the diagonal lines) and thus subsequently updates the pointer of $\Pi(x_1)$ to $z \cdot (n_2 - 1) - 1$ for the smallest $z \in \mathbb{N} \setminus \{0\}$ such that $z \cdot (n_2 - 1) = z' \cdot n_2 - n_1$ (recall that the pointer of $\Pi(x_0)$ with period n_2 is already n_1 positions ahead) for some $z' \in \mathbb{N} \setminus \{0\}$, as α_{algn} only holds at the ends of the blocks of x_0 and x_1. Accordingly, the until-operator updates the pointer of $\Pi(x_0)$ to $z \cdot (n_2 - 1) - 1 + n_1$ and the pointer of $\Pi(x_{y_3})$ to $z \cdot (n_2 - 1) - 1 + n_1$. These positions are marked by "b".
- Then, the next-operator subsequently updates the pointer of $\Pi(x_{y_3})$ to $z \cdot (n_2 - 1) + n_1$, which is marked by "c" in Figure 6.

As argued above, z must be equal to n_1, i.e., the pointer of $\Pi(x_{y_3})$ is then equal to $n_1 \cdot (n_2 - 1) + n_1 = n_1 \cdot n_2$. At that position, ψ_3 requires that # holds on $\Pi(x_{y_3})$. Hence, we have indeed implemented multiplication of n_1 and n_2, provided we have $0 < n_1 \leq n_2$ and $n_2 \geq 2$.

For the final case, i.e., $0 < n_2 < n_1$, we use a similar construction, but swap the roles of y_1 and y_2 in ψ_3, to obtain a formula ψ_4. Then, let $hyp(y_1 \cdot y_2 = y_3) = \psi_1 \vee \psi_2 \vee \psi_3 \vee \psi_4$.

Now, let $\mathcal{T}$ be a fixed transition system with $\mathrm{Tr}(\mathcal{T}) = (2^{\{\#\}})^\omega \cup (2^{\{\$\}})^\omega$, Here, $(2^{\{\#\}})^\omega$ contains the traces to mimic set quantification and $(2^{\{\$\}})^\omega$ contains the traces for x_j and x'_j used to implement multiplication. An induction shows that we have $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models hyp(\varphi)$. Again, $hyp(\varphi)$ can be turned into a HyperLTL$_\mathrm{C}$ formula (i.e., in prenex normal form), as no quantifier is under the scope of a temporal operator. □

Combining the results of this section, we obtain our main result settling the complexity of model-checking for GHyLTL$_{\mathrm{S+C}}$ and its fragments HyperLTL$_\mathrm{S}$ and HyperLTL$_\mathrm{C}$.

Theorem 5.4. *The model-checking problems for the logics GHyLTL$_{S+C}$, HyperLTL$_S$, and HyperLTL$_C$ are all polynomial-time equivalent to truth in second-order arithmetic.*

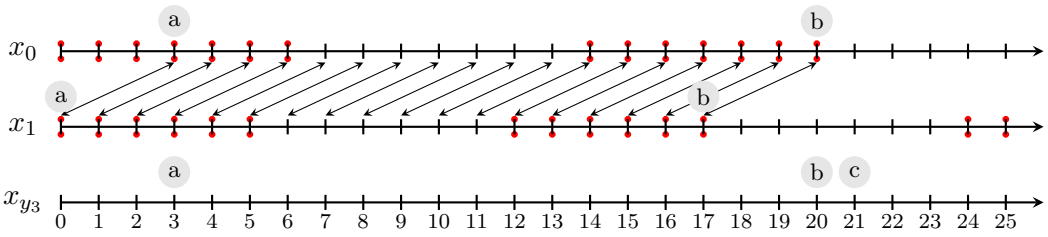


Figure 6: The formula ψ_3 implementing multiplication, for $n_1 = 3$ and $n_2 = 7$, i.e., x_0 has period 7 and x_1 has period 6. Here, "!" ("|") denotes a position where \$ holds (does not hold).

6. GHyLTL$_{S+C}$ Finite-state Satisfiability

In this section, we settle the complexity of the finite-state satisfiability problems for GHyLTL$_{S+C}$ and its fragments: Given a sentence φ, is there a transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$ (recall that our transition systems are all, by definition, finite)? We will show that the problem is, for GHyLTL$_{S+C}$, HyperLTL$_S$, and HyperLTL$_C$, equivalent to truth in second-order arithmetic, i.e., as hard as the model-checking problems.

We begin by proving a lower bound for HyperLTL$_S$.

Lemma 6.1. *Truth in second-order arithmetic is polynomial-time reducible to HyperLTL$_S$ finite-state satisfiability.*

Proof. Recall that in the proof of Lemma 5.2, we have presented a polynomial-time computable function *hyp* mapping sentences φ of second-order arithmetic to HyperLTL$_S$ sentences $hyp(\varphi)$ such that $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models hyp(\varphi)$, where $\mathcal{T}$ is a transition system with

$$\mathrm{Tr}(\mathcal{T}) = (2^{\{\#\}})^\omega \cup \bigcup_{y,y' \in V_1} (2^{\{\#(y),\#(y'),\$\}})^\omega \cup (2^{\{\$'\}})^\omega,$$

where V_1 denotes the set of first-order variables in φ. Call that set of traces $\mathcal{L}_{V_1}$.

To simplify the following construction, let us introduce some notation. Let $\mathrm{AP}_1 = \{\#\}$, $\mathrm{AP}_{(y,y')} = \{\#(y), \#(y'), \$\}$ for $(y, y') \in V_1 \times V_1$, and $\mathrm{AP}_2 = \{\$'\}$, let $I = \{1, 2\} \cup V_1 \times V_1$, and let $\mathrm{AP} = \bigcup_i \mathrm{AP}_i$, where i ranges over I. Note that AP is the set of propositions used in $hyp(\varphi)$.

In the following, we construct a formula α_{V_1} such that a finite transition system satisfies α_{V_1} if and only if its set of traces (projected to AP) is equal to $\mathcal{L}_{V_1}$. Then, $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\alpha_{V_1} \wedge hyp(\varphi)$ is finite-state satisfiable, which yields the desired lower bound.

We define α_{V_1} to be the conjunction of the following sentences which uses a fresh proposition †, the indices in I as fresh propositions, and the propositions in AP:

- $\forall x.\ \bigvee_{i \in I} (i_x \wedge \bigwedge_{i' \in I \setminus \{i\}} \neg i'_x)$, which expresses that each trace in a model is, at the first position, uniquely marked by an index $i \in I$.
- $\forall x.\ \mathbf{G}(\dagger_x \rightarrow \mathbf{X}\,\mathbf{G}\,\neg\dagger_x)$, which expresses that each trace in a model has at most one position where the proposition † holds.
- $\forall x.\ \bigwedge_{i \in I} i_x \rightarrow \bigwedge_{p \in \mathrm{AP} \setminus \mathrm{AP}_i} \mathbf{G}\,\neg p_x$, which expresses that each trace in a model may not contain propositions from set $\mathrm{AP} \setminus \mathrm{AP}_i$, where i is the unique mark of the trace.
- $\bigwedge_{i \in I} \exists x.\ i_x \wedge \dagger_x$, which expresses that each model contains, for each i, a trace marked by i where † holds at the first position.
- $\bigwedge_{i \in I} \bigwedge_{A \subseteq \mathrm{AP}_i} \forall x.\ \exists x'.\ (i_x \wedge \mathbf{F}\,\dagger_x) \rightarrow [i_{x'} \wedge (\bigwedge_{p \in \mathrm{AP}_i} p_x \leftrightarrow p_{x'})\,\mathbf{U}(\dagger_x \wedge (\bigwedge_{p \in A} p_{x'} \wedge \bigwedge_{p \notin A} \neg p_{x'}) \wedge \mathbf{X}\,\dagger_{x'})]$, which expresses that for every $i \in I$, every subset of A of AP_i, and every trace t in a model marked by i and containing a †, there is another trace t' in the model, also marked by i and containing a †, such that the AP_i-projection w of t up to the † and the AP_i-projection w' of t' up to the † satisfy $w' = wA$.

One can easily construct a finite-state transition system that satisfies α_{V_1}.

Fix some $i \in I$. An induction shows that the set of prefixes of the AP_i-projection of each model of α_{V_1} is equal to $(2^{\mathrm{AP}_i})^*$. Furthermore, let $\mathfrak{T}$ be a finite transition system that is a model of α_{V_1}. Then, the AP_i-projection of $\mathrm{Tr}(\mathfrak{T})$ must be equal to $(2^{\mathrm{AP}_i})^\omega$, which follows from the fact that the set of traces of a finite transition system is closed (see, e.g., [BK08] for the necessary definitions).

As this reasoning holds for every $i \in I$, and as α_{V_1} ensures that each trace in a model only contains propositions from a unique AP_i, we conclude that the AP-projection of $\mathrm{Tr}(\mathfrak{T})$ must be equal to $\mathcal{L}_{V_1}$. □

Next, we use the same approach to prove a similar lower bound for $\mathrm{HyperLTL_C}$.

Lemma 6.2. *Truth in second-order arithmetic is polynomial-time reducible to HyperLTL$_C$ finite-state satisfiability.*

Proof. Recall that in the proof of Lemma 5.3, we have presented a polynomial-time computable function *hyp* mapping sentences φ of second-order arithmetic to $\mathrm{HyperLTL_C}$ sentences $\mathit{hyp}(\varphi)$ such that $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\mathcal{T} \models \mathit{hyp}(\varphi)$, where $\mathcal{T}$ is a fixed transition system with $\mathrm{Tr}(\mathcal{T}) = (2^{\{\#\}})^\omega \cup (2^{\{\$\}})^\omega$. Call that set of traces $\mathcal{L}$.

As in the proof of Lemma 6.1, we can construct a formula α that is satisfied by a finite transition system if and only if the $\{\#, \$\}$-projection of its set of traces is equal to $\mathcal{L}$. Then, $(\mathbb{N}, +, \cdot, <, \in) \models \varphi$ if and only if $\alpha \wedge \mathit{hyp}(\varphi)$ is finite-state satisfiable, which yields the desired lower bound. □

Finally, we prove a matching upper bound. Recall that we have proved that $\mathrm{GHyLTL_{S+C}}$ model-checking (of finite-state transition systems) is reducible to truth in second-order arithmetic (see Lemma 5.1). To prove this, we captured the semantics of $\mathrm{GHyLTL_{S+C}}$ in second-order arithmetic, which in particular involved, for a finite transition system $\mathcal{T}$, a formula $\alpha_{\mathcal{T}}(X)$ with a free second-order variable X that is satisfied by $(\mathbb{N}, +, \cdot, <, \in)$ if and only if the interpretation of X encodes a trace of $\mathcal{T}$. To reduce finite-state satisfiability of a $\mathrm{GHyLTL_{S+C}}$ sentence φ to truth in second-order arithmetic, we existentially quantify the transition system $\mathcal{T}$ in second-order arithmetic and then intuitively model-check whether $\mathcal{T}$ satisfies φ in second-order arithmetic as before.

Lemma 6.3. *GHyLTL$_{S+C}$ finite-state satisfiability is polynomial-time reducible to truth in second-order arithmetic.*

Proof. Recall that in the proof of Lemma 5.1, we have encoded traces σ over a fixed set AP by the set

$$S_\sigma = \{\mathit{pair}(j, h_{\mathit{prop}}(p)) \mid j \in \mathbb{N} \text{ and } p \in \sigma(j)\} \subseteq \mathbb{N},$$

where $h_{\mathit{prop}} \colon \mathrm{AP} \to \{0, 1, \ldots, |\mathrm{AP}| - 1\}$ and $\mathit{pair} \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ are a bijections.

Using this encoding, there are formulas (see [FRZ26, Proof of Theorem 4.4])

- $\alpha_{\mathit{isTS}}(n, E, I, \ell)$ with free first-order variable n and free second-order variables E, I, and ℓ that is satisfied by $(\mathbb{N}, +, \cdot, <, \in)$ if and only if the interpretations of the free variables encode a finite transition system, and
- $\alpha_{\mathit{traceOf}}(X, n, E, I, \ell)$ that is satisfied by $(\mathbb{N}, +, \cdot, <, \in)$ if and only the interpretation of X encodes a trace (using the above encoding) of the transition system encoded by the interpretations of n, E, I, and ℓ.[3]

Then, the finite-state satisfiability of a $\mathrm{GHyLTL_{S+C}}$ sentence φ is equivalent to

$$(\mathbb{N}, +, \cdot, <, \in) \models \exists n.\ \exists E.\ \exists I.\ \exists \ell.\ \alpha_{\mathit{isTS}}(n, E, I, \ell) \wedge \mathit{ar}'_{\mathsf{VAR}'}(\varphi)(\emptyset),$$

where VAR' is the set of variables occurring in φ and where $\mathit{ar}'_{\mathsf{VAR}'}(\varphi)(\emptyset)$ is the formula obtained from the formula $\mathit{ar}_{\mathsf{VAR}'}(\varphi)(\emptyset)$ constructed in the proof of Lemma 5.1 by replacing each occurrence of $\alpha_{\mathcal{T}}(X)$ for a second-order variable X by $\alpha_{\mathit{traceOf}}(X, n, E, I, \ell)$. □

[3] Using the notation introduced in [FRZ26], $\alpha_{\mathit{traceOf}}(X, n, E, I, \ell)$ is the formula $\exists P.\ \varphi_{\mathit{isPath}}(P, n, E, I) \wedge \varphi_{\mathit{traceOf}}(X, P, \ell)$.

Combining the results of this section, we obtain our main result settling the complexity of finite-state satisfiability for $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and its fragments $\mathrm{HyperLTL}_{\mathrm{S}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$.

Theorem 6.4. *The finite-state satisfiability problems for the logics GHyLTL_{S+C}, $\mathrm{HyperLTL}_{S}$, and $\mathrm{HyperLTL}_{C}$ are all polynomial-time equivalent to truth in second-order arithmetic.*

Remark 6.5. Again, the simple fragments of $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$ contain HyperLTL. Hence, their finite-state satisfiability problems are Σ_1^0-hard with respect to polynomial-time reductions, as for HyperLTL [FH16]. Finally, as model-checking is decidable for both simple fragments [BPS21, BBST24], one can exhaustively check all finite transition systems until a satisfying one is found. Hence, their finite-state satisfiability problems are in fact Σ_1^0-complete with respect to polynomial-time reductions.

7. Conclusion

In this work, we have settled the complexity of $\mathrm{GHyLTL}_{\mathrm{S+C}}$, an expressive logic for the specification of asynchronous hyperproperties. Although it is obtained by adding stuttering, contexts, and trace quantification under the scope of temporal operators to HyperLTL, we have proven that its satisfiability problem is as hard as that of its (much weaker) fragment HyperLTL. On the other hand, model-checking $\mathrm{GHyLTL}_{\mathrm{S+C}}$ and finite-state satisfiability for $\mathrm{GHyLTL}_{\mathrm{S+C}}$ are much harder than for HyperLTL, i.e., equivalent to truth in second-order arithmetic vs. decidable and Σ_1^0-complete. Here, the lower bounds hold for simpler fragments, i.e., $\mathrm{HyperLTL}_{\mathrm{S}}$ and $\mathrm{HyperLTL}_{\mathrm{C}}$, and very simple transition systems, that in the case of $\mathrm{HyperLTL}_{\mathrm{C}}$ do not even depend on the formula.

Our work extends a line of work that has settled the complexity of synchronous hyperlogics like HyperLTL [FKTZ25], HyperQPTL [RZ26b], and second-order HyperLTL [FRZ26], as well as asynchronous HyperLTL [RZ26a]. In future work, we aim to resolve the exact complexity of other logics for asynchronous hyperproperties proposed in the literature.

References

[BBST24] Alberto Bombardelli, Laura Bozzelli, César Sánchez, and Stefano Tonetta. Unifying asynchronous logics for hyperproperties. In Siddharth Barman and Slawomir Lasota, editors, *FSTTCS 2024*, volume 323 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.FSTTCS.2024.14.

[BCB+21] Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. A temporal logic for asynchronous hyperproperties. In Alexandra Silva and K. Rustan M. Leino, editors, *CAV 2021, Part I*, volume 12759 of *LNCS*, pages 694–717. Springer, 2021. doi:10.1007/978-3-030-81685-8_33.

[BHNdC23] Ezio Bartocci, Thomas A. Henzinger, Dejan Nickovic, and Ana Oliveira da Costa. Hypernode automata. In Guillermo A. Pérez and Jean-François Raskin, editors, *CONCUR 2023*, volume 279 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CONCUR.2023.21.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BPS21] Laura Bozzelli, Adriano Peron, and César Sánchez. Asynchronous extensions of HyperLTL. In *LICS 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470583.

[BPS22] Laura Bozzelli, Adriano Peron, and César Sánchez. Expressiveness and decidability of temporal logics for asynchronous hyperproperties. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *CONCUR 2022*, volume 243 of *LIPIcs*, pages 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CONCUR.2022.27.

[CFK+14] 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.

[CS10] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.

[FH16] Bernd Finkbeiner and Christopher Hahn. Deciding hyperproperties. In Josée Desharnais and Radha Jagadeesan, editors, *CONCUR 2016*, volume 59 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.13.

[FHHT20] 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.

[Fin17] Bernd Finkbeiner. Temporal hyperproperties. *Bull. EATCS*, 123, 2017.

[FKTZ25] Marie Fortin, Louwe B. Kuijer, Patrick Totzke, and Martin Zimmermann. HyperLTL satisfiability is highly undecidable, HyperCTL* is even harder. *Log. Methods Comput. Sci.*, 21(1):3, 2025. doi:10.46298/LMCS-21(1:3)2025.

[FRS15] 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.

[FRZ26] Hadar Frenkel, Gaëtan Regaud, and Martin Zimmermann. The complexity of second-order HyperLTL. *Log. Methods Comput. Sci.*, 22(1), 2026. doi:10.46298/LMCS-22(1:26)2026.

[FZ17] Bernd Finkbeiner and Martin Zimmermann. The First-Order Logic of Hyperproperties. In *STACS 2017*, volume 66 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.30.

[GM82] Joseph A. Goguen and José Meseguer. Security policies and security models. In *S&P 1982*, pages 11–20. IEEE Computer Society, 1982. doi:10.1109/SP.1982.10014.

[GMO20] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. Propositional dynamic logic for hyperproperties. In Igor Konnov and Laura Kovács, editors, *CONCUR 2020*, volume 171 of *LIPIcs*, pages 50:1–50:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.CONCUR.2020.50.

[GMO21] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. Automata and fixpoints for asynchronous hyperproperties. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021. doi:10.1145/3434319.

[MZ20] 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.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE, Oct 1977. doi:10.1109/SFCS.1977.32.

[Rab16] Markus N. Rabe. *A temporal logic approach to information-flow control*. PhD thesis, Saarland University, 2016.

[Rog87] Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.

[RZ25] Gaëtan Regaud and Martin Zimmermann. The complexity of generalized HyperLTL with stuttering and contexts. In Giorgio Bacci and Adrian Francalanza, editors, *GandALF 2025*, volume 428 of *EPTCS*, pages 161–176. Open Publishing Association, 2025. doi:10.4204/EPTCS.428.12.

[RZ26a] Gaëtan Regaud and Martin Zimmermann. The complexity of asynchronous HyperLTL. *arXiv*, 2606.06091, 2026. Accepted for publication at GandALF 2026. arXiv:2606.06091, doi:10.48550/ARXIV.2606.06091.

[RZ26b] Gaëtan Regaud and Martin Zimmermann. The complexity of HyperQPTL. *Inf. Process. Lett.*, 193:106626, 2026. doi:10.1016/J.IPL.2026.106626.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany