The Complexity of Asynchronous HyperLTL

Joint work with Gaëtan Regaud (ENS Rennes, France)

Martin Zimmermann
Aalborg University

GandALF 2026, Aalborg, Denmark

The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.

The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL	Σ^2_1-complete.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ^2_1-compl.	Σ^0_1-compl.	TOWER-compl.

The First Generation of Hyperlogics

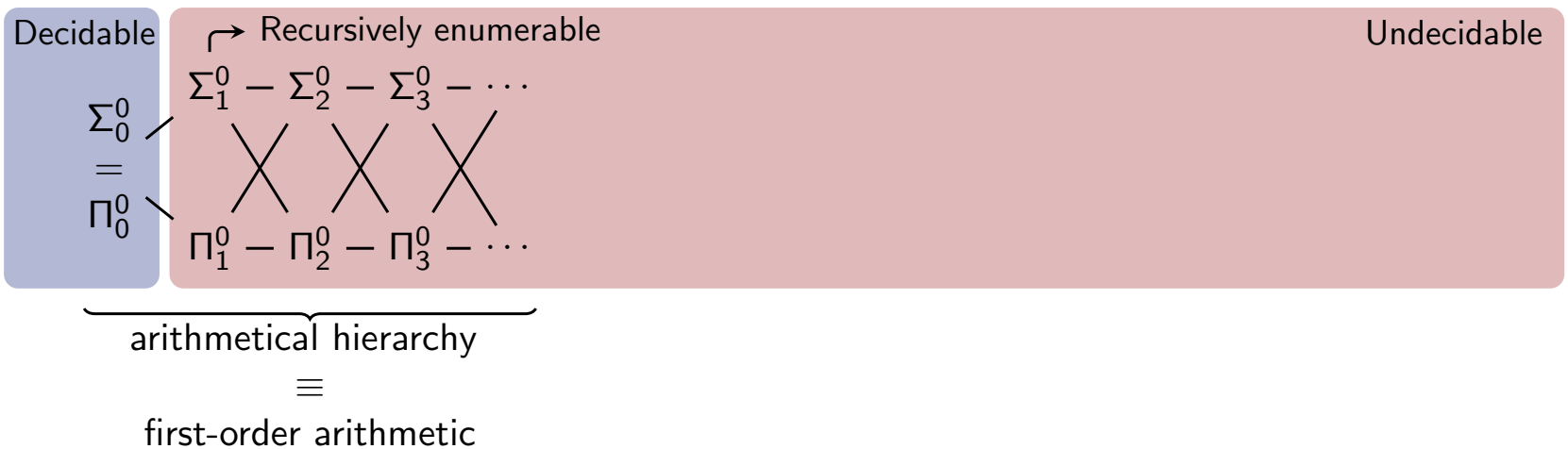
Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL	Σ^2_1-complete.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ^2_1-compl.	Σ^0_1-compl.	TOWER-compl.

The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL	Σ^2_1-complete.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ^2_1-compl.	Σ^0_1-compl.	TOWER-compl.

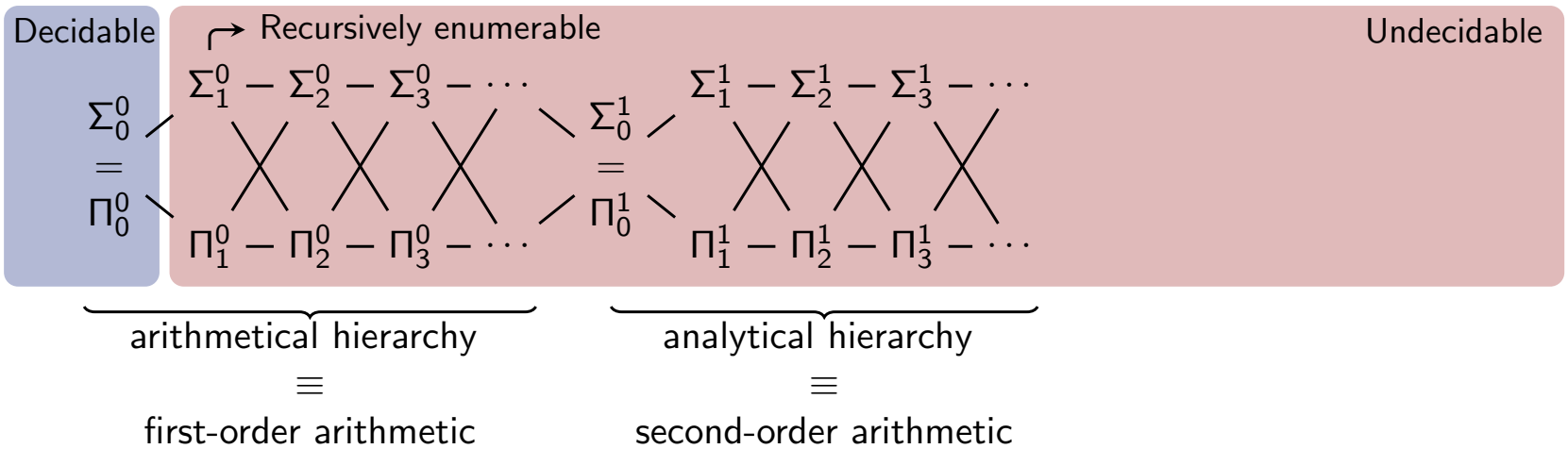
The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL	Σ^2_1-complete.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ^2_1-compl.	Σ^0_1-compl.	TOWER-compl.



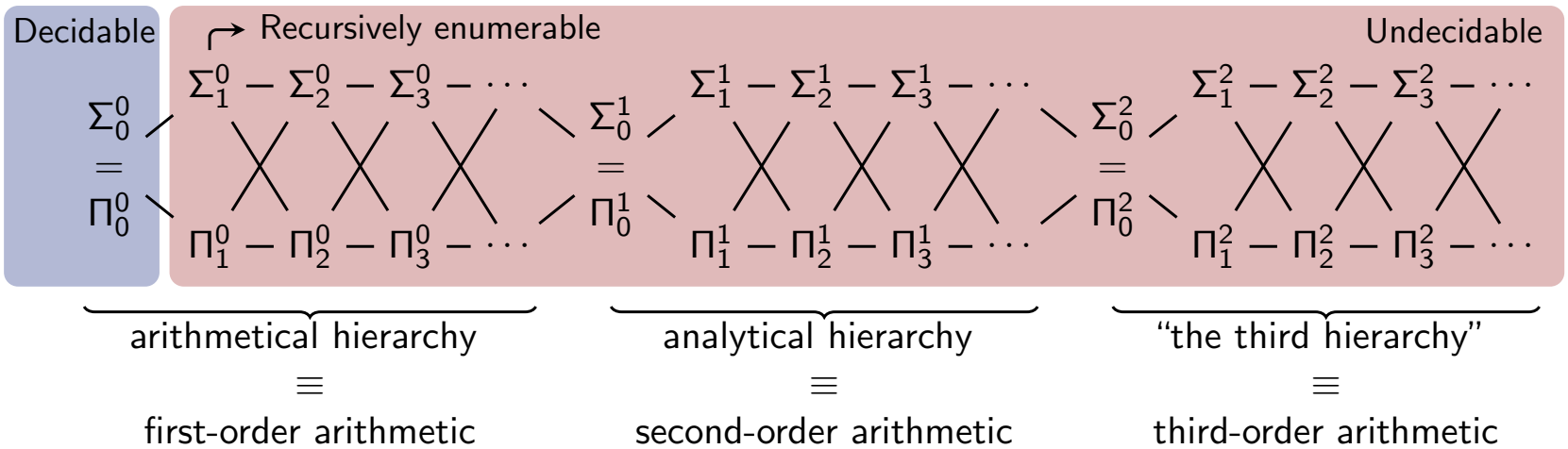
The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperPDL-Δ	Σ^1_1-compl.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL	Σ^2_1-complete.	Σ^0_1-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ^2_1-compl.	Σ^0_1-compl.	TOWER-compl.



The First Generation of Hyperlogics

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperPDL-Δ	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL	Σ_1^2-complete.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ_1^2-compl.	Σ_1^0-compl.	TOWER-compl.



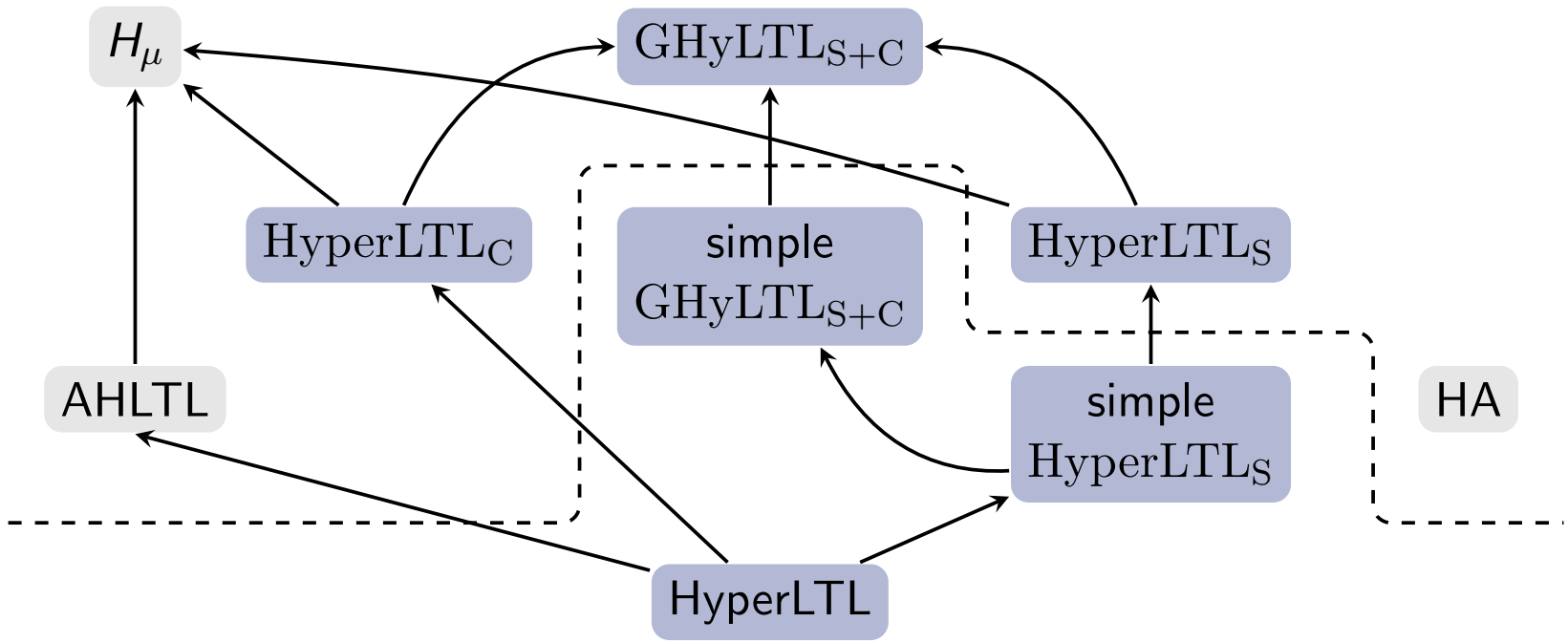
Synchronicity

- All these logics are synchronous, but
- not every system is synchronous.

Synchronicity

- All these logics are synchronous, but
- not every system is synchronous.

The second generation of **asynchronous** hyperlogics:



Results for Stuttering and Contexts

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperPDL-Δ	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL	Σ_1^2-complete.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ_1^2-compl.	Σ_1^0-compl.	TOWER-compl.
$\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
$\mathrm{HyperLTL}_{\mathrm{S}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
$\mathrm{HyperLTL}_{\mathrm{C}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
simple $\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
simple $\mathrm{HyperLTL}_{\mathrm{S}}$	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.

Trajectories

Trajectories describe how time passes on different traces.

Example

$$\pi: \quad \emptyset \quad \emptyset \quad \emptyset \quad \{a\} \quad \emptyset \quad \emptyset \quad \emptyset \quad \emptyset \quad \emptyset \quad \cdots$$

$$\pi': \quad \{b\} \quad \emptyset \quad \{b\} \quad \emptyset \quad \{b\} \quad \emptyset \quad \emptyset \quad \emptyset \quad \emptyset \quad \cdots$$

Trajectories

Trajectories describe how time passes on different traces.

Example

$$\begin{array}{lccccccccc} \pi\colon & \emptyset & \emptyset & \emptyset & \{a\} & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \cdots \\ \pi'\colon & \{b\} & \emptyset & \{b\} & \emptyset & \{b\} & \emptyset & \emptyset & \emptyset & \emptyset & \cdots \end{array}$$

- With trajectory $\{\pi'\}(\{\pi,\pi'\})^\omega$, $\mathbf{F}(a_\pi \wedge b_{\pi'})$ is satisfied.

Trajectories

Trajectories describe how time passes on different traces.

Example

$$\begin{array}{lccccccccc} \pi\colon & \emptyset & \emptyset & \emptyset & \{a\} & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \cdots \\ \pi'\colon & \{b\} & \emptyset & \{b\} & \emptyset & \{b\} & \emptyset & \emptyset & \emptyset & \emptyset & \cdots \end{array}$$

- With trajectory $\{\pi'\}(\{\pi,\pi'\})^\omega$, $\mathbf{F}(a_\pi \wedge b_{\pi'})$ is satisfied.
- With trajectory $\{\pi\}^3(\emptyset)^\omega$, $\mathbf{G}\,\mathbf{F}(a_\pi \wedge b_{\pi'})$ is satisfied.

Asynchronous HyperLTL (AHLTL)

AHLTL extends HyperLTL with trajectory quantification:

$$\mathcal{O}_0\pi_0\mathcal{O}_1\pi_1\cdots\mathcal{O}_{k-1}\pi_{k-1}\,\mathsf{Q}\,\psi$$

where

- $\mathcal{O}_0\pi_0\mathcal{O}_1\pi_1\cdots\mathcal{O}_{k-1}\pi_{k-1}$ is a block of trace quantifiers (i.e., $\mathcal{O}_j \in \{\exists, \forall\}$) as in HyperLTL,
- Q existentially ($\mathsf{Q} = \mathsf{E}$) or universally ($\mathsf{Q} = \mathsf{A}$) quantifies a trajectory t for the variables $\pi_0, \ldots, \pi_{k-1}$, and
- ψ is a quantifier-free HyperLTL formula over $\pi_0, \ldots, \pi_{k-1}$ that is evaluated with respect to the trajectory t.

Asynchronous HyperLTL (AHLTL)

AHLTL extends HyperLTL with trajectory quantification:

$$\mathcal{O}_0\pi_0\mathcal{O}_1\pi_1\cdots\mathcal{O}_{k-1}\pi_{k-1}\,\mathsf{Q}\,\psi$$

where

- $\mathcal{O}_0\pi_0\mathcal{O}_1\pi_1\cdots\mathcal{O}_{k-1}\pi_{k-1}$ is a block of trace quantifiers (i.e., $\mathcal{O}_j \in \{\exists, \forall\}$) as in HyperLTL,
- Q existentially ($\mathsf{Q} = \mathsf{E}$) or universally ($\mathsf{Q} = \mathsf{A}$) quantifies a trajectory t for the variables $\pi_0, \ldots, \pi_{k-1}$, and
- ψ is a quantifier-free HyperLTL formula over $\pi_0, \ldots, \pi_{k-1}$ that is evaluated with respect to the trajectory t.

Fragments:

- E-AHLTL: existential trajectory quantification
- A-AHLTL: universal trajectory quantification

State of the Art

Baumeister, Coenen, Bonakdarpour, Finkbeiner, and Sánchez ('21)

- Model-checking E-AHLTL is undecidable, but
- there are fragments with decidable model-checking that can express important asynchronous hyperproperties.

State of the Art

Baumeister, Coenen, Bonakdarpour, Finkbeiner, and Sánchez ('21)

- Model-checking E-AHLTL is undecidable, but
- there are fragments with decidable model-checking that can express important asynchronous hyperproperties.

Bozzelli, Peron, and Sánchez ('22)

- HyperLTL satisfiability can be reduced to E-AHLTL satisfiability, so it is Σ^1_1-hard.

Accessing Trajectories

The quantified trajectory t can only indirectly be accessed:
$p_\pi \leftrightarrow \mathbf{X}\, \neg p_\pi$ can only hold if time advances one step on π w.r.t. t.

Accessing Trajectories

The quantified trajectory t can only indirectly be accessed:
$p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi$ can only hold if time advances one step on π w.r.t. t.

Lemma

Let t be a trajectory.

1. *If $\mathbf{G}(p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi)$ is satisfied w.r.t. t, then time passes on π at every position.*

Accessing Trajectories

The quantified trajectory t can only indirectly be accessed:
$p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi$ can only hold if time advances one step on π w.r.t. t.

Lemma

Let t be a trajectory.

1. *If $\mathbf{G}(p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi)$ is satisfied w.r.t. t, then time passes on π at every position.*
2. *Let π be mapped to a trace where the truth value of p changes at every step. Then, $\mathbf{G}(p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi)$ is satisfied w.r.t. t if and only if time passes on π at every position.*

Accessing Trajectories

The quantified trajectory t can only indirectly be accessed:
$p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi$ can only hold if time advances one step on π w.r.t. t.

Lemma

Let t be a trajectory.

1. *If $\mathbf{G}(p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi)$ is satisfied w.r.t. t, then time passes on π at every position.*
2. *Let π be mapped to a trace where the truth value of p changes at every step. Then, $\mathbf{G}(p_\pi \leftrightarrow \mathbf{X}\,\neg p_\pi)$ is satisfied w.r.t. t if and only if time passes on π at every position.*

This allows us to reduce HyperLTL to AHLTL and to constrain trajectories.

Satisfiability

Recall:

Lemma (Bozzelli, Peron, and Sánchez ('22))

E-AHLTL satisfiability is as hard as HyperLTL satisfiability, i.e., Σ^1_1*-hard.*

A similar reduction works for A-AHLTL, too.

Satisfiability

Recall:

Lemma (Bozzelli, Peron, and Sánchez ('22))

E-AHLTL satisfiability is as hard as HyperLTL satisfiability, i.e., Σ^1_1-hard.

A similar reduction works for A-AHLTL, too.

Theorem

- *E-AHLTL satisfiability is Σ^1_1-complete.*
- *A-AHLTL satisfiability is Σ^1_1-hard and in Σ^1_2.*

Satisfiability

Recall:

Lemma (Bozzelli, Peron, and Sánchez ('22))

E-AHLTL satisfiability is as hard as HyperLTL satisfiability, i.e., Σ^1_1-hard.

A similar reduction works for A-AHLTL, too.

Theorem

- *E-AHLTL satisfiability is Σ^1_1-complete.*
- *A-AHLTL satisfiability is Σ^1_1-hard and in Σ^1_2.*

Proof sketch

- Every satisfiable AHLTL formula has a countable model.
- Express the existence of countable models in arithmetic.

“Small” Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

“Small” Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall \pi_0 \exists \pi'_0 \cdots \forall \pi_k \exists \pi'_k \, \mathsf{Q} \, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π'_j.

“Small” Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall \pi_0 \exists \pi_0' \cdots \forall \pi_k \exists \pi_k' \, \mathsf{Q} \, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π_j'.

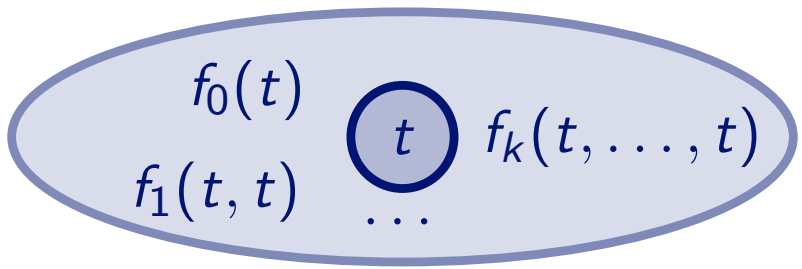
"Small" Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall\pi_0\exists\pi'_0 \cdots \forall\pi_k\exists\pi'_k\, \mathsf{Q}\, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π'_j.



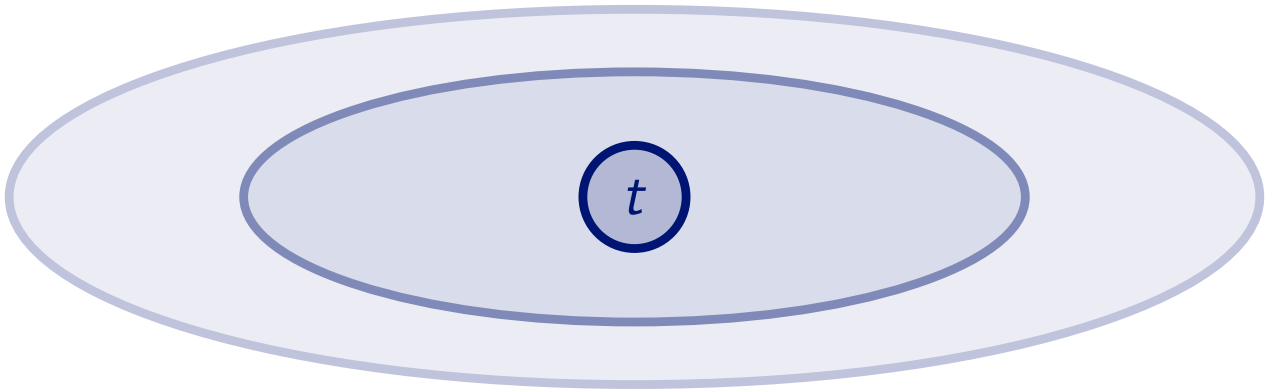
“Small” Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall \pi_0 \exists \pi_0' \cdots \forall \pi_k \exists \pi_k' \, \mathsf{Q} \, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π_j'.



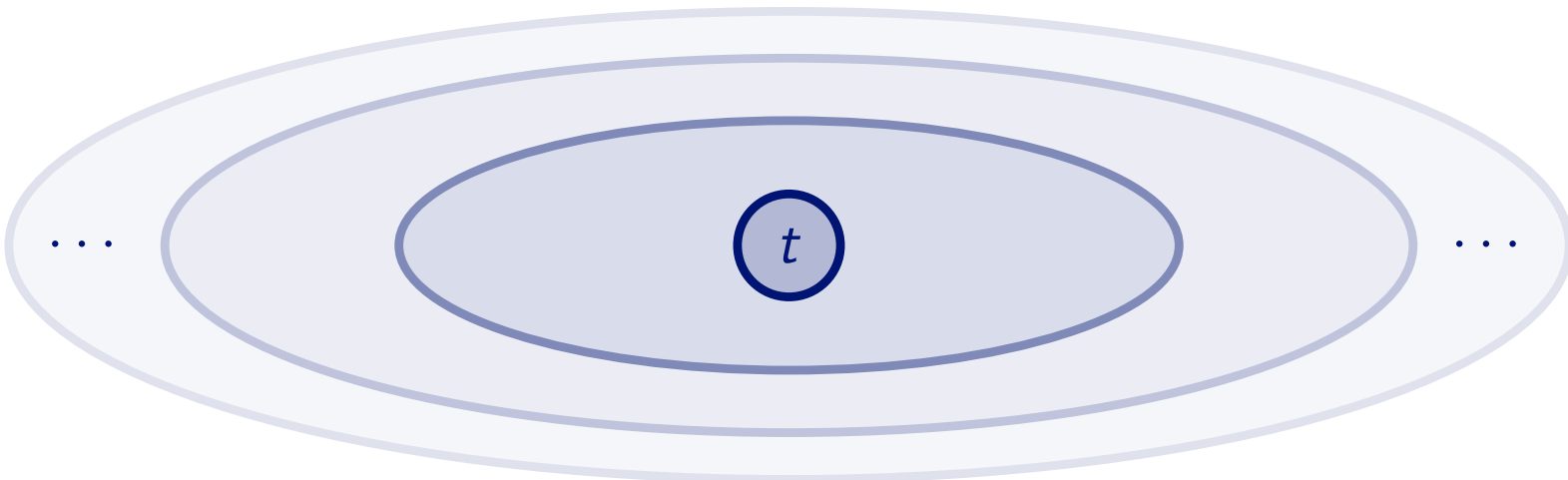
“Small” Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall \pi_0 \exists \pi_0' \cdots \forall \pi_k \exists \pi_k' \, \mathsf{Q} \, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π_j'.



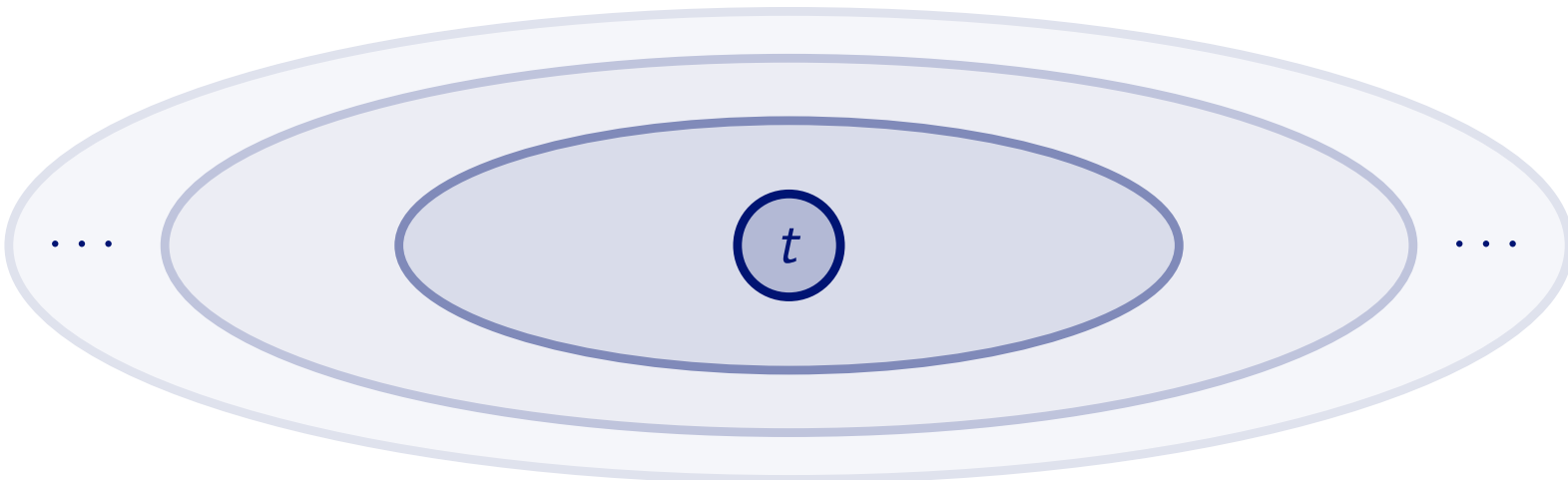
"Small" Models

Theorem

Every satisfiable AHLTL sentence has a countable model.

Proof

- W.l.o.g. $\varphi = \forall \pi_0 \exists \pi_0' \cdots \forall \pi_k \exists \pi_k' \, \mathsf{Q} \, \psi$ with quantifier-free ψ.
- Fix a Skolem function f_j for every existentially quantified π_j'.



The limit is a model of φ and countable.

Upper Bound Proof

- A countable model can be encoded by a function from $\mathbb{N}^3$ to $\{0,1\}$ and, therefore, also by a set of natural numbers (encoding the graph of the function).

Upper Bound Proof

- A countable model can be encoded by a function from $\mathbb{N}^3$ to $\{0,1\}$ and, therefore, also by a set of natural numbers (encoding the graph of the function).
- Skolem functions and trajectories can be encoded by sets of natural numbers.

Upper Bound Proof

- A countable model can be encoded by a function from $\mathbb{N}^3$ to $\{0,1\}$ and, therefore, also by a set of natural numbers (encoding the graph of the function).
- Skolem functions and trajectories can be encoded by sets of natural numbers.
- The (unique!) tableaux of a quantifier-free formula can be encoded by a set of natural numbers.

Upper Bound Proof

- A countable model can be encoded by a function from $\mathbb{N}^3$ to $\{0,1\}$ and, therefore, also by a set of natural numbers (encoding the graph of the function).
- Skolem functions and trajectories can be encoded by sets of natural numbers.
- The (unique!) tableaux of a quantifier-free formula can be encoded by a set of natural numbers.
- All these objects can be existentially quantified for E-AHLTL satisfiability: $\exists T\, \exists \mathcal{S}\, \exists t\, \exists \mathit{Tblx}\, \psi(T, \mathcal{S}, t, \mathit{Tblx})$ with first-order ψ.

Upper Bound Proof

- A countable model can be encoded by a function from $\mathbb{N}^3$ to $\{0,1\}$ and, therefore, also by a set of natural numbers (encoding the graph of the function).
- Skolem functions and trajectories can be encoded by sets of natural numbers.
- The (unique!) tableaux of a quantifier-free formula can be encoded by a set of natural numbers.
- All these objects can be existentially quantified for E-AHLTL satisfiability: $\exists T\, \exists \mathcal{S}\, \exists t\, \exists \mathit{Tblx}\, \psi(T, \mathcal{S}, t, \mathit{Tblx})$ with first-order ψ.
- For A-AHLTL satisfiability, the trajectory needs to be quantified universally: $\exists T\, \exists \mathcal{S}\, \forall t\, \forall \mathit{Tblx}\, \psi'(T, \mathcal{S}, t, \mathit{Tblx})$ with first-order ψ'.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Upper bound: Encode AHLTL model-checking in second-order arithmetic:

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Upper bound: Encode AHLTL model-checking in second-order arithmetic:

- Traces can be encoded as sets of natural numbers.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Upper bound: Encode AHLTL model-checking in second-order arithmetic:

- Traces can be encoded as sets of natural numbers.
- One can write a formula of second-order arithmetic that checks whether a trace comes from a given transition system.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Upper bound: Encode AHLTL model-checking in second-order arithmetic:

- Traces can be encoded as sets of natural numbers.
- One can write a formula of second-order arithmetic that checks whether a trace comes from a given transition system.
- Assignments and trajectories can be encoded as sets of natural numbers.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Upper bound: Encode AHLTL model-checking in second-order arithmetic:

- Traces can be encoded as sets of natural numbers.
- One can write a formula of second-order arithmetic that checks whether a trace comes from a given transition system.
- Assignments and trajectories can be encoded as sets of natural numbers.
- Hence, the whole semantics of AHLTL can be encoded in second-order arithmetic.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

- Traces can encode sets of natural numbers (and thus natural numbers).

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

- Traces can encode sets of natural numbers (and thus natural numbers).
- Use a fixed transition system that contains all those traces.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

- Traces can encode sets of natural numbers (and thus natural numbers).
- Use a fixed transition system that contains all those traces.
- Membership ($x \in X$) and order ($x < y$) straightforward.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

- Traces can encode sets of natural numbers (and thus natural numbers).
- Use a fixed transition system that contains all those traces.
- Membership ($x \in X$) and order ($x < y$) straightforward.
- Addition and multiplication: clever use of trajectories.

Model-Checking

Theorem

Model-checking AHLTL is equivalent to truth in second-order arithmetic. The lower bound holds for E-AHLTL and A-AHLTL.

Proof idea

Lower bound: Encode second-order arithmetic in A-AHLTL model-checking:

- Traces can encode sets of natural numbers (and thus natural numbers).
- Use a fixed transition system that contains all those traces.
- Membership ($x \in X$) and order ($x < y$) straightforward.
- Addition and multiplication: clever use of trajectories.

The result for E-AHLTL follows by duality.

Results

Logic	Satisfiability	Finite-state Sat.	Model-checking
HyperLTL	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperPDL-Δ	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL	Σ_1^2-complete.	Σ_1^0-compl.	TOWER-compl.
HyperQPTL$^+$	T3A-equiv.	T3A-equiv.	T3A-equiv.
Hyper^{2}LTL	T3A-equiv.	T3A-equiv.	T3A-equiv.
HyperCTL*	Σ_1^2-compl.	Σ_1^0-compl.	TOWER-compl.
$\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
$\mathrm{HyperLTL}_{\mathrm{S}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
$\mathrm{HyperLTL}_{\mathrm{C}}$	Σ_1^1-compl.	T2A-equiv.	T2A-equiv.
simple $\mathrm{GHyLTL}_{\mathrm{S+C}}$	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
simple $\mathrm{HyperLTL}_{\mathrm{S}}$	Σ_1^1-compl.	Σ_1^0-compl.	TOWER-compl.
AHLTL	Σ_1^1-hard/in Σ_2^1	T2A-equiv.	T2A-equiv.
A-AHLTL	Σ_1^1-hard/in Σ_2^1	T2A-equiv.	T2A-equiv.
E-AHLTL	Σ_1^1-complete.	T2A-equiv.	T2A-equiv.

Future Work

- Close the gap for A-AHLTL satisfiability (Σ^1_1/Σ^1_2).

Future Work

- Close the gap for A-AHLTL satisfiability (Σ^1_1/Σ^1_2).
- Consider AHLTL with multiple trajectory quantifiers.

Future Work

- Close the gap for A-AHLTL satisfiability (Σ^1_1/Σ^1_2).
- Consider AHLTL with multiple trajectory quantifiers.
- There are more logics: