The Complexity of Second-order HyperLTL

Joint work with Hadar Frenkel
(Bar-Ilan University, Ramat Gan, Israel)

Martin Zimmermann

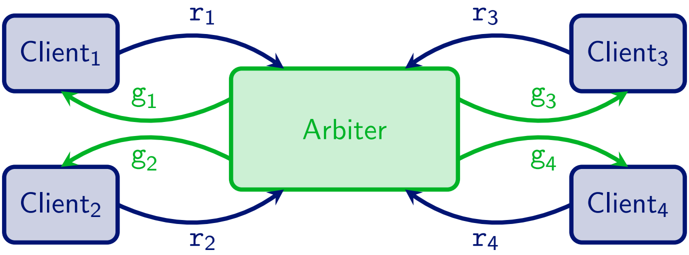
Aalborg University

February 2025

CSL 2025

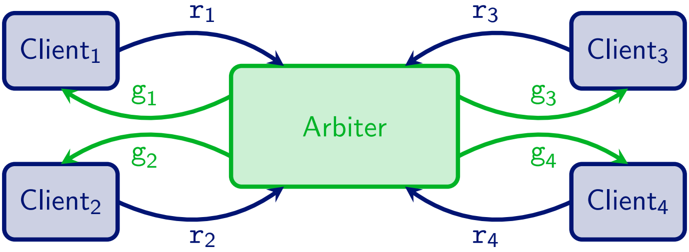
Reactive Systems

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Reactive Systems

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system

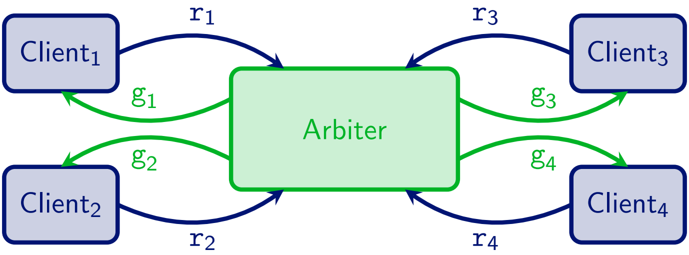


Env:

Sys:

Reactive Systems

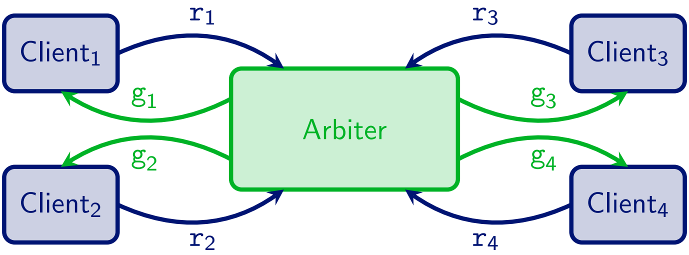
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env: $\{\mathtt{r}_1, \mathtt{r}_2\}$
Sys:

Reactive Systems

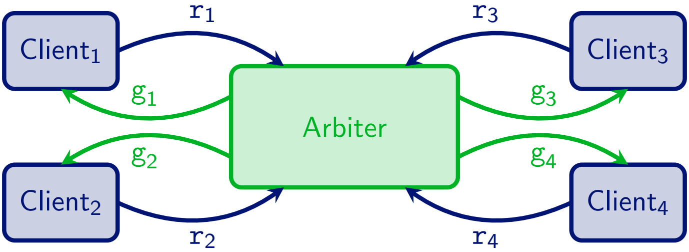
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env: $\{\mathtt{r}_1, \mathtt{r}_2\}$
Sys: $\{\mathtt{g}_1\}$

Reactive Systems

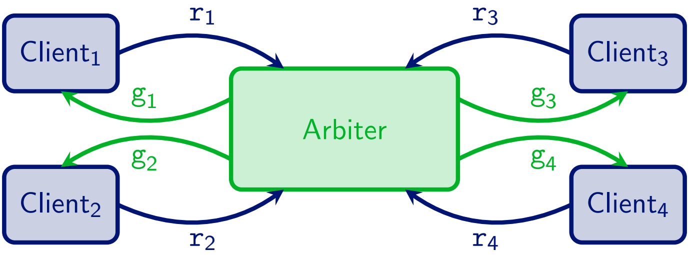
- Setting: an arbiter with n clients
- requests $\texttt{r}_i$ from client i controlled by the environment
- grants $\texttt{g}_i$ for client i controlled by the system



Env: $\{\texttt{r}_1, \texttt{r}_2\}$ $\{\texttt{r}_1\}$
Sys: $\{\texttt{g}_1\}$

Reactive Systems

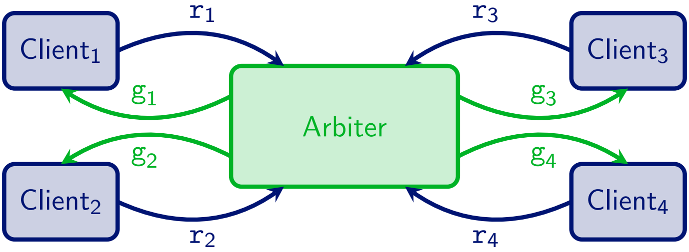
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$

Reactive Systems

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system

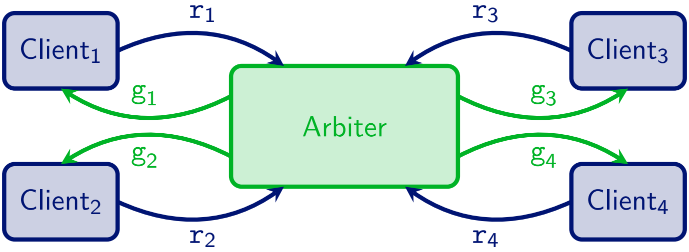


Env: $\{\mathtt{r}_1, \mathtt{r}_2\}$ $\{\mathtt{r}_1\}$ $\{\mathtt{r}_1, \mathtt{r}_4\}$

Sys: $\{\mathtt{g}_1\}$ $\{\mathtt{g}_2\}$

Reactive Systems

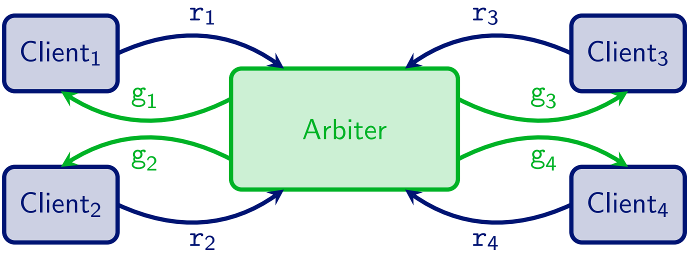
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$

Reactive Systems

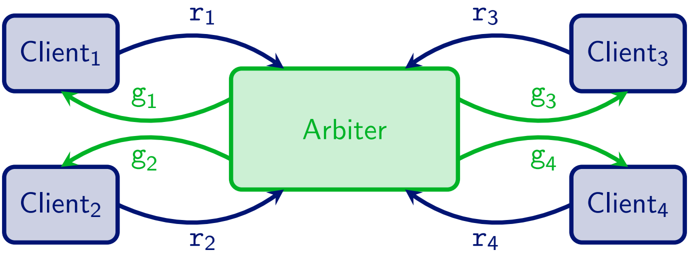
- Setting: an arbiter with n clients
- requests $\texttt{r}_i$ from client i controlled by the environment
- grants $\texttt{g}_i$ for client i controlled by the system



Env:	$\{\texttt{r}_1, \texttt{r}_2\}$	$\{\texttt{r}_1\}$	$\{\texttt{r}_1, \texttt{r}_4\}$	$\emptyset$
Sys:	$\{\texttt{g}_1\}$	$\{\texttt{g}_2\}$	$\{\texttt{g}_3\}$	

Reactive Systems

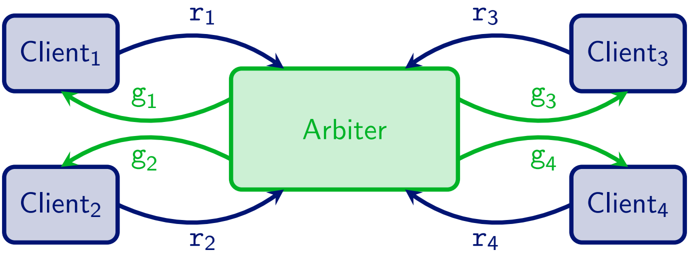
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$

Reactive Systems

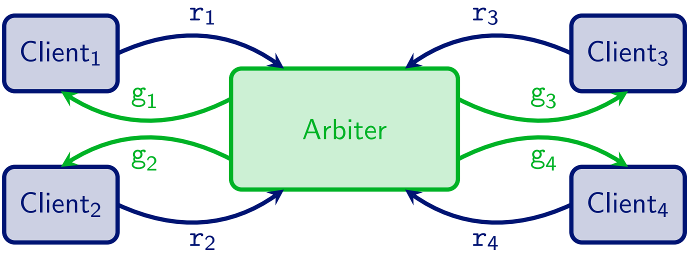
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	

Reactive Systems

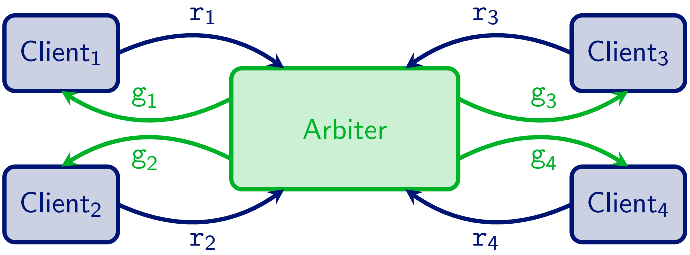
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$

Reactive Systems

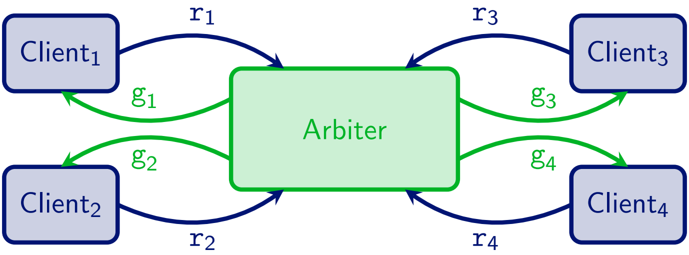
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$	

Reactive Systems

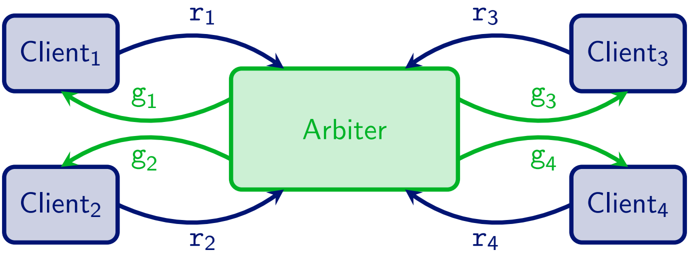
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$

Reactive Systems

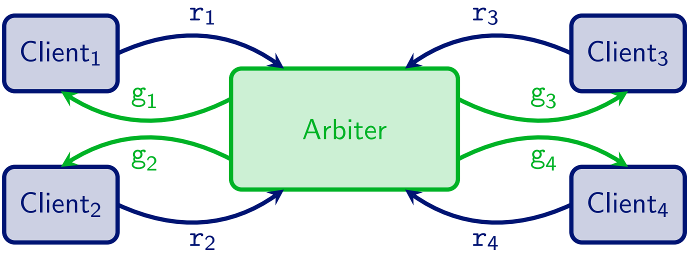
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathtt{r}_2\}$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	

Reactive Systems

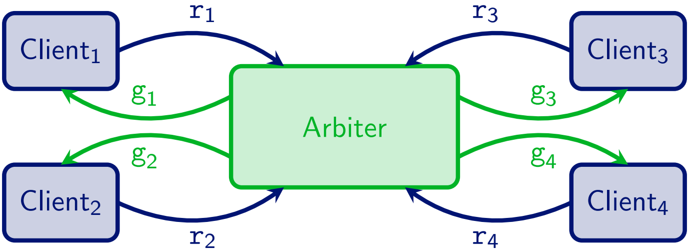
- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathtt{r}_2\}$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$

Reactive Systems

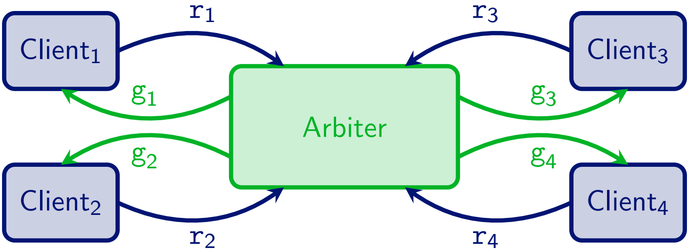
- Setting: an arbiter with n clients
- requests $\texttt{r}_i$ from client i controlled by the environment
- grants $\texttt{g}_i$ for client i controlled by the system



Env:	$\{\texttt{r}_1, \texttt{r}_2\}$	$\{\texttt{r}_1\}$	$\{\texttt{r}_1, \texttt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\texttt{r}_2\}$	$\{\texttt{r}_1\}$
Sys:	$\{\texttt{g}_1\}$	$\{\texttt{g}_2\}$	$\{\texttt{g}_3\}$	$\{\texttt{g}_4\}$	$\{\texttt{g}_1\}$	$\{\texttt{g}_2\}$	$\{\texttt{g}_3\}$	

Reactive Systems

- Setting: an arbiter with n clients
- requests $\mathtt{r}_i$ from client i controlled by the environment
- grants $\mathtt{g}_i$ for client i controlled by the system



Env:	$\{\mathtt{r}_1, \mathtt{r}_2\}$	$\{\mathtt{r}_1\}$	$\{\mathtt{r}_1, \mathtt{r}_4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathtt{r}_2\}$	$\{\mathtt{r}_1\}$
Sys:	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$	$\{\mathtt{g}_1\}$	$\{\mathtt{g}_2\}$	$\{\mathtt{g}_3\}$	$\{\mathtt{g}_4\}$

Trace Properties

Goal: specify properties of (transition) systems $\mathfrak{T}$ in terms of their set $\mathrm{Tr}(\mathfrak{T}) \subseteq (2^{\mathrm{AP}})^\omega$ of traces, where AP is a finite set of atomic propositions.

Trace Properties

Goal: specify properties of (transition) systems $\mathfrak{T}$ in terms of their set $\mathrm{Tr}(\mathfrak{T}) \subseteq (2^{\mathrm{AP}})^\omega$ of traces, where AP is a finite set of atomic propositions.

Typical properties:

- Answer every request: every $\mathtt{r}_i$ is eventually followed by some $\mathtt{g}_i$.

Trace Properties

Goal: specify properties of (transition) systems $\mathfrak{T}$ in terms of their set $\mathrm{Tr}(\mathfrak{T}) \subseteq (2^{\mathrm{AP}})^\omega$ of traces, where AP is a finite set of atomic propositions.

Typical properties:

- Answer every request: every $\mathtt{r}_i$ is eventually followed by some $\mathtt{g}_i$.
- At most one grant at a time: if $i \neq j$, then $\mathtt{g}_i$ and $\mathtt{g}_j$ are never true at the same time.

Trace Properties

Goal: specify properties of (transition) systems $\mathfrak{T}$ in terms of their set $\mathrm{Tr}(\mathfrak{T}) \subseteq (2^{\mathrm{AP}})^\omega$ of traces, where AP is a finite set of atomic propositions.

Typical properties:

- Answer every request: every $\mathtt{r}_i$ is eventually followed by some $\mathtt{g}_i$.
- At most one grant at a time: if $i \neq j$, then $\mathtt{g}_i$ and $\mathtt{g}_j$ are never true at the same time.
- No spurious grants: $\mathtt{g}_i$ is true only if $\mathtt{r}_i$ has been true before and $\mathtt{g}_i$ has not been true in the meantime.

Trace Properties

Goal: specify properties of (transition) systems $\mathfrak{T}$ in terms of their set $\mathrm{Tr}(\mathfrak{T}) \subseteq (2^{\mathrm{AP}})^\omega$ of traces, where AP is a finite set of atomic propositions.

Typical properties:

- Answer every request: every $\mathtt{r}_i$ is eventually followed by some $\mathtt{g}_i$.
- At most one grant at a time: if $i \neq j$, then $\mathtt{g}_i$ and $\mathtt{g}_j$ are never true at the same time.
- No spurious grants: $\mathtt{g}_i$ is true only if $\mathtt{r}_i$ has been true before and $\mathtt{g}_i$ has not been true in the meantime.

All these properties φ are trace properties, i.e., each φ is a set of traces and $\mathfrak{T}$ satisfies φ if $\mathrm{Tr}(\mathfrak{T}) \subseteq \varphi$.

LTL in a Nutshell

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

where p ranges over AP.

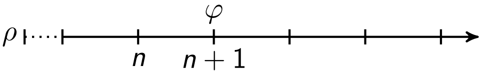
LTL in a Nutshell

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

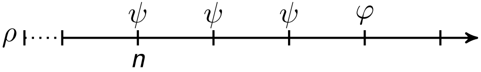
where p ranges over AP.

Semantics: $\rho \in (2^{\mathrm{AP}})^\omega$, $n \in \mathbb{N}$

- $(\rho, n) \models \mathbf{X}\,\varphi$:



- $(\rho, n) \models \psi\,\mathbf{U}\,\varphi$:



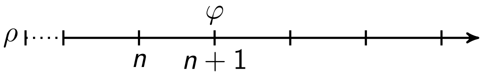
LTL in a Nutshell

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

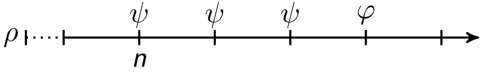
where p ranges over AP.

Semantics: $\rho \in (2^{\mathrm{AP}})^\omega$, $n \in \mathbb{N}$

- $(\rho, n) \models \mathbf{X}\,\varphi$:



- $(\rho, n) \models \psi\,\mathbf{U}\,\varphi$:



Syntactic sugar:

- $\mathbf{F}\,\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$: φ holds eventually (finally φ)
- $\mathbf{G}\,\varphi = \neg\,\mathbf{F}\,\neg\varphi$: φ holds always (generally φ)

Examples

LTL expresses trace properties.

Examples

LTL expresses trace properties.

- Answer every request: $\bigwedge_i \mathbf{G}(\mathtt{r}_i \rightarrow \mathbf{F}\,\mathtt{g}_i)$

Examples

LTL expresses trace properties.

- Answer every request: $\bigwedge_i \mathbf{G}(\mathtt{r}_i \rightarrow \mathbf{F}\, \mathtt{g}_i)$
- At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$

Examples

LTL expresses trace properties.

- Answer every request: $\bigwedge_i \mathbf{G}(\mathtt{r}_i \rightarrow \mathbf{F}\, \mathtt{g}_i)$
- At most one grant at a time: $\mathbf{G} \bigwedge_{i \neq j} \neg(\mathtt{g}_i \wedge \mathtt{g}_j)$
- No spurious grants:

$$\bigwedge_i \neg[\,(\neg \mathtt{r}_i\, \mathbf{U}(\neg \mathtt{r}_i \wedge \mathtt{g}_i))\,] \wedge \neg[\,\mathbf{F}(\mathtt{g}_i \wedge \mathbf{X}(\neg \mathtt{r}_i\, \mathbf{U}(\neg \mathtt{r}_i \wedge \mathtt{g}_i)))\,]$$

But not Everything is a Trace Property

- Input determinism: projection to the $\mathtt{r}_i$ uniquely determines the projection to the $\mathtt{g}_i$.

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_3\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_4\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

But not Everything is a Trace Property

- Input determinism: projection to the $\mathtt{r}_i$ uniquely determines the projection to the $\mathtt{g}_i$.

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_3\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_4\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

- This property (and many others) cannot be expressed by considering single traces in isolation. Instead one needs to reason about pairs of traces.

But not Everything is a Trace Property

- Input determinism: projection to the $\mathtt{r}_i$ uniquely determines the projection to the $\mathtt{g}_i$.

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_3\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

$$\{\mathtt{r}_1, \mathtt{g}_1\} \quad \{\mathtt{r}_2\} \quad \{\mathtt{r}_3, \mathtt{g}_2\} \quad \{\mathtt{r}_4\} \quad \{\} \quad \{\mathtt{r}_1, \mathtt{g}_4\} \quad \{\mathtt{r}_1\} \quad \{\mathtt{g}_4\} \quad \cdots$$

- This property (and many others) cannot be expressed by considering single traces in isolation. Instead one needs to reason about pairs of traces.
- Clarkson and Schneider termed such properties hyperproperties: formally, they are sets of sets of traces.
- $\mathfrak{T}$ satisfies a hyperproperty $H \subseteq 2^{(2^{\mathrm{AP}})^\omega}$ if $\mathrm{Tr}(\mathfrak{T}) \in H$.

HyperLTL in a Nutshell

- To express hyperproperties, Clarkson et al. introduced HyperLTL, LTL + trace quantification (in prenex normal form).
- Input determinism:

$$\forall \pi.\ \forall \pi'.\ (\mathbf{G} \bigwedge_i (\mathtt{r}_i)_\pi \leftrightarrow (\mathtt{r}_i)_{\pi'}) \rightarrow (\mathbf{G} \bigwedge_i (\mathtt{g}_i)_\pi \leftrightarrow (\mathtt{g}_i)_{\pi'})$$

HyperLTL in a Nutshell

- To express hyperproperties, Clarkson et al. introduced HyperLTL, LTL + trace quantification (in prenex normal form).
- Input determinism:

$$\forall \pi.\ \forall \pi'.\ (\mathbf{G} \bigwedge_i (\mathtt{r}_i)_\pi \leftrightarrow (\mathtt{r}_i)_{\pi'}) \rightarrow (\mathbf{G} \bigwedge_i (\mathtt{g}_i)_\pi \leftrightarrow (\mathtt{g}_i)_{\pi'})$$

- HyperLTL is able to express many information-flow properties from security and privacy.
- Model-checking is decidable and successfully implemented (for a small number of quantifier alternations).
- Much more exciting work!

But HyperLTL Cannot Express Everything

Consider common knowledge in multi-agent systems:

- An agent knows that a property φ holds if it holds on all traces that are indistinguishable in the agent's view.
- φ is common knowledge among the agents if all agents know φ, all agents know that all agents know φ, and so on.

But HyperLTL Cannot Express Everything

Consider common knowledge in multi-agent systems:

- An agent knows that a property φ holds if it holds on all traces that are indistinguishable in the agent's view.
- φ is common knowledge among the agents if all agents know φ, all agents know that all agents know φ, and so on.
- Bozelli et al. proved that common knowledge is not expressible in HyperLTL.

Second-order HyperLTL

- Beutner et al. introduced second-order HyperLTL (written $\mathrm{Hyper}^2\mathrm{LTL}$), HyperLTL + quantification over sets of traces.
- Can express common knowledge, asynchronous hyperproperties, and more.

$$\forall \pi.\, \exists X.$$
$$\pi \in X \wedge \left(\forall \pi' \in X.\, \forall \pi''.\, \left(\bigvee_{i=1}^{n} \pi' \sim_i \pi''\right) \rightarrow \pi'' \in X\right) \wedge$$
$$\forall \pi' \in X.\, \varphi(\pi')$$

Second-order HyperLTL

- Beutner et al. introduced second-order HyperLTL (written $\mathrm{Hyper}^2\mathrm{LTL}$), HyperLTL + quantification over sets of traces.
- Can express common knowledge, asynchronous hyperproperties, and more.

$$\forall \pi. \exists X.$$
$$\pi \in X \land \left(\forall \pi' \in X. \forall \pi''. \left(\bigvee_{i=1}^{n} \pi' \sim_i \pi''\right) \to \pi'' \in X\right) \land$$
$$\forall \pi' \in X.\ \varphi(\pi')$$

Open problem:
What is the complexity of satisfiability (SAT) and model-checking (MC) for second-order HyperLTL?

Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
Hyper^{2}LTL	?	Σ_1^1-hard

Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
$\mathrm{Hyper}^2\mathrm{LTL}$	?	Σ_1^1-hard

We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$

Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
Hyper^{2}LTL	?	Σ_1^1-hard

We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$

Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
Hyper^{2}LTL	?	Σ_1^1-hard

We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$

Known Results

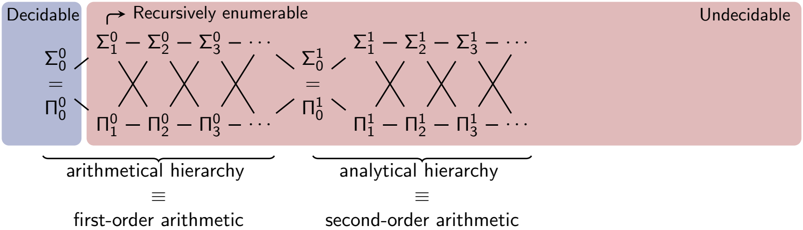
Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
Hyper^2LTL	?	Σ_1^1-hard

We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$

Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ^1_1-complete	TOWER-complete
Hyper^2LTL	?	Σ^1_1-hard

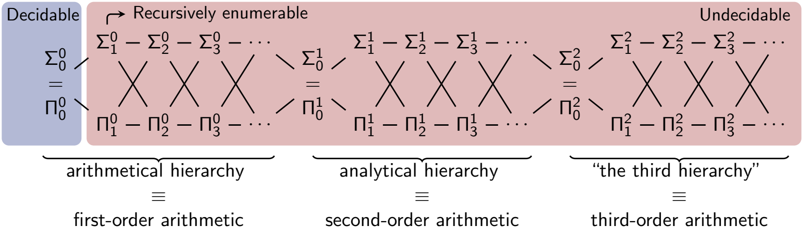
We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$



Known Results

Logic	Satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete
HyperLTL	Σ_1^1-complete	TOWER-complete
$\mathrm{Hyper}^2\mathrm{LTL}$	?	Σ_1^1-hard

We classify the complexity of undecidable problems using arithmetic, i.e., predicate logic with signature $(+, \cdot, <, \in)$, evaluated over $(\mathbb{N}, +, \cdot, <, \in)$



The Complexity of Second-order HyperLTL

Theorem

Second-order HyperLTL SAT and MC are polynomial-time equivalent to truth in third-order arithmetic.

The Complexity of Second-order HyperLTL

Theorem

Second-order HyperLTL SAT and MC are polynomial-time equivalent to truth in third-order arithmetic.

Proof sketch

Lower bound:

- Traces can encode (characteristic sequences of) sets of natural numbers, so sets of traces encode sets of sets of natural numbers.
- Fortin et al. showed that addition and multiplication can be *implemented* in HyperLTL.

The Complexity of Second-order HyperLTL

Theorem

Second-order HyperLTL SAT and MC are polynomial-time equivalent to truth in third-order arithmetic.

Proof sketch

Upper bound:

- Sets of natural numbers can encode traces (and encoding is implementable in first-order arithmetic), so sets of sets of natural numbers can encode sets of traces.
- Semantics of temporal operators can be expressed in arithmetic.

Least Fixed Points

Recall the formula for common knowledge:

$$\forall\pi.\,\exists X.\,\overbrace{\pi \in X \wedge \left(\forall\pi' \in X.\,\forall\pi''.\,\left(\bigvee_{i=1}^{n} \pi' \sim_i \pi''\right) \rightarrow \pi'' \in X\right)}^{\psi} \wedge \forall\pi' \in X.\ \varphi(\pi')$$

- We are only interested in the smallest X that satisfies ψ.

Least Fixed Points

Recall the formula for common knowledge:

$$\forall \pi.\, \exists X.\, \overbrace{\pi \in X \wedge \left(\forall \pi' \in X.\, \forall \pi''.\, \left(\bigvee_{i=1}^{n} \pi' \sim_i \pi''\right) \to \pi'' \in X\right)}^{\psi} \wedge \forall \pi' \in X.\, \varphi(\pi')$$

- We are only interested in the smallest X that satisfies ψ.
- This set can be captured by a least fixed point computation.
- Such least fixed points are sufficient for many other applications of second-order HyperLTL.
- Beutner et al. presented a partial model checking algorithm for this fragment that relies on approximations of the fixed points.

Least Fixed Points

Recall the formula for common knowledge:

$$\forall\pi.\,\exists X.\,\overbrace{\pi \in X \wedge \left(\forall\pi' \in X.\,\forall\pi''.\,\left(\bigvee_{i=1}^{n} \pi' \sim_i \pi''\right) \rightarrow \pi'' \in X\right)}^{\psi} \wedge$$
$$\forall\pi' \in X.\,\varphi(\pi')$$

- We are only interested in the smallest X that satisfies ψ.
- This set can be captured by a least fixed point computation.
- Such least fixed points are sufficient for many other applications of second-order HyperLTL.
- Beutner et al. presented a partial model checking algorithm for this fragment that relies on approximations of the fixed points.

Open problem:

Does this fragment have better complexity?

"Small" Models

Lemma

Every satisfiable lfp-second-order HyperLTL formula has a countable model.

“Small” Models

Lemma

Every satisfiable lfp-second-order HyperLTL formula has a countable model.

Proof sketch

- Let T be an uncountable model of φ and let $t_0 \in T$ be an arbitrary trace in T.
- Let R be the smallest subset of T that
  - contains t_0,
  - is closed under application of Skolem functions, and
  - contains witnesses for all traces being in the required fixed points.
- Then, R is a countable model of φ.

Main Theorem

Theorem

lfp-second-order HyperLTL SAT is Σ_1^1-complete.

Main Theorem

Theorem

lfp-second-order HyperLTL SAT is Σ^1_1-complete.

Proof sketch

The lower bound already holds for the fragment HyperLTL, so let us consider the upper bound.

- Express the existence of a countable model, Skolem functions, and more; all encoded as sets of natural numbers.
- Use first-order arithmetic to ensure correctness.

Main Theorem

Theorem

lfp-second-order HyperLTL SAT is Σ_1^1-complete.

Proof sketch

The lower bound already holds for the fragment HyperLTL, so let us consider the upper bound.

- Express the existence of a countable model, Skolem functions, and more; all encoded as sets of natural numbers.
- Use first-order arithmetic to ensure correctness.

So, you can add least fixed points to HyperLTL for free (at least for satisfiability).

Also in the Paper

- Model checking lfp-second-order HyperLTL
- Finite-state satisfiability
- Two semantics

Logic	Satisfiability	Finite-state satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete	PSPACE-complete
HyperLTL	Σ^1_1-complete	Σ^0_1-complete	TOWER-complete
$\mathrm{Hyper}^2\mathrm{LTL}$	**T3A-equivalent**	**T3A-equivalent**	**T3A-equivalent**
$\mathrm{Hyper}^2\mathrm{LTL}_{\mathrm{mm}}$	**T3A-equivalent**	**T3A-equivalent**	**T3A-equivalent**
$\mathrm{lfp}\text{-}\mathrm{Hyper}^2\mathrm{LTL}_{\mathrm{mm}}$	**Σ^1_1-complete***	**Σ^1_1-hard/in Σ^2_2**	**Σ^1_1-hard/in Σ^2_2**

Meanwhile

- Regaud and Zimmermann: closed all gaps (arxiv.org/abs/2501.19046)
- Regaud and Zimmermann: the complexity of HyperQPTL (arxiv.org/abs/2412.07341)

Logic	Satisfiability	Finite-state satisfiability	Model-checking
LTL	PSPACE-complete	PSPACE-complete	PSPACE-complete
HyperLTL	Σ^1_1-complete	Σ^0_1-complete	TOWER-complete
$\mathrm{Hyper}^2\mathrm{LTL}$	T3A-equivalent	T3A-equivalent	T3A-equivalent
$\mathrm{Hyper}^2\mathrm{LTL}_{\mathrm{mm}}$	T3A-equivalent	T3A-equivalent	T3A-equivalent
$\mathrm{Hyper}^2\mathrm{LTL}^{\curlywedge}_{\mathrm{mm}}$	**T3A-equivalent**	**T3A-equivalent**	**T3A-equivalent**
$\mathrm{Hyper}^2\mathrm{LTL}^{\curlyvee}_{\mathrm{mm}}$	**T3A-equivalent**	**T3A-equivalent**	**T3A-equivalent**
lfp-$\mathrm{Hyper}^2\mathrm{LTL}_{\mathrm{mm}}$	**Σ^2_1-complete (STD)**/ Σ^1_1-complete (CW)	**T2A-equivalent**	**T2A-equivalent**
HyperQPTL	**T2A-equivalent**	**Σ^0_1-complete**	**Tower-complete**
$\mathrm{HyperQPTL}^+$	**T3A-equivalent**	**T3A-equivalent**	**T3A-equivalent**