UNIVERSITY
OF TRENTO

FONDAZIONE
BRUNO KESSLER

Compositional Reasoning and Model Checking of Asynchronous Systems on Variations of LTL

PhD candidate: **Alberto Bombardelli**

University of Trento, Italy

27-Oct-2025

PhD Advisor: **Stefano Tonetta**

Motivation

Safety Critical systems are everywhere:

Motivation

Safety Critical systems are everywhere:

Bugs and failures cause distasters:

Model checking

High level idea Does a system M satisfy a specification Φ?

$$\underbrace{M}_{System} \models \underbrace{\Phi}_{Specification}$$

Model checking

High level idea Does a system M satisfy a specification Φ?

$$\underbrace{M}_{System} \models \underbrace{\Phi}_{Specification}$$

Yes: Our system is correct

Model checking

High level idea Does a system M satisfy a specification Φ?

$$\underbrace{M}_{System} \models \underbrace{\Phi}_{Specification}$$

Yes: Our system is correct **No:** We get a counter-example

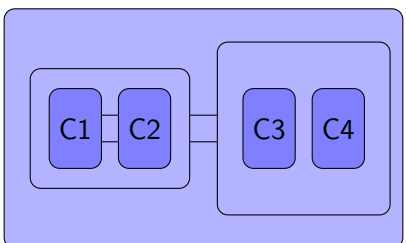
Challenge: Scalability

State space explosion

Increasing size of the system makes unfeasible the verification

Type of specification

Expressive specifications are significantly harder to verify



$\models \varphi$

Challenge: Property Expressiveness

Real time properties

Will my system respond in 3 seconds?

Security properties

Can an intruder learn secret data?

Diagnosability

Can I detect a failure in my system with partial observability?

Challenge: System Complexity

Model Asynchronicity

Components run in interleaving

Shared Data

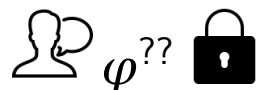
Component A sends messages to component B

Timed System

The system obeys some real time timing constraints

Problem statement

1. Scalability
   - state space explosion
   - complex type of specification
2. Expressiveness
   - Security properties
   - Diagnosability
   - Real time properties
3. System Complexity
   - Scheduling
   - Shared data
   - timed system



Papers

NFM22 [Asynchronous Composition of Local Interface LTL Properties - A. Bombardelli, S. Tonetta]

DATE23 [Metric Temporal Logic with Resettable Skewed Clocks (Extended abstract) - A. Bombardelli, S. Tonetta]

NFM23 [Reasoning with Metric Temporal Logic and Resettable Skewed Clocks - A. Bombardelli, S. Tonetta - A. Bombardelli, S. Tonetta]

iFM23 [Symbolic Model Checking of Relative Safety Properties - A. Bombardelli, A. Cimatti, S. Tonetta, M. Zamboni]

JPK60 [Another Look at LTL Modulo Theory over Finite and Infinite Traces - A. Bombardelli, A. Cimatti, A. Griggio, S. Tonetta]

FSTTCS24 [Unifying Asynchronous Logics for Hyperproperties - A. Bombardelli, A. Bozzelli, C. Sánchez, S. Tonetta]

SPIN25 [Asynchronous) Temporal Logics for Hyperproperties on Finite Traces - A. Bombardelli, A. Bozzelli, C. Sánchez, S. Tonetta]

LMCS25 (Minor revision UR)[Asynchronous Composition of LTL Properties over Infinite and Finite Traces - A. Bombardelli, S. Tonetta]

Structure of the talk

LTL Model Checking: iFM23,JPK60

Compositional Verification:NFM22,LMCS25(UR?) DATE23,NFM23

Hyperproperty Verification:FSTTCS24,SPIN25

Conclusion

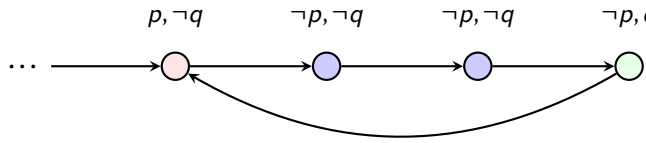
LTL Model Checking

Linear Temporal Logic (LTL)[Pnueli77]

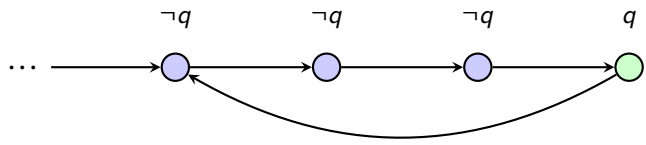
- Used for verification of reactive systems
- Represents time as sequence of actions

$$\varphi ::= p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\phi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi \mid \mathbf{F}\varphi$$

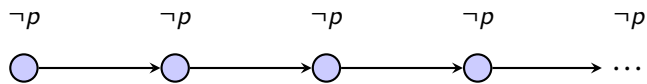
Response: $\mathbf{G}(p \rightarrow \mathbf{F}q)$



Fairness: $\mathbf{GF}q$



Absence $\mathbf{G}\neg p$



LTL modulo theory: $\mathbf{G}(in \geq 5 \rightarrow \mathbf{F}o = f(in))$

LTL Model Checking

Symbolic Transition System:

$$M = \langle \underbrace{V}_{\text{variables}} , \underbrace{I(V)}_{\text{Initial conditions}} , \underbrace{T(V, V')}_{\text{transition relation}} \rangle$$

LTL Model Checking

Symbolic Transition System:

$$M = \langle \underbrace{V}_{\text{variables}} , \underbrace{I(V)}_{\text{Initial conditions}} , \underbrace{T(V, V')}_{\text{transition relation}} \rangle$$

Reduction to liveness

$$M \models \varphi \Leftrightarrow M \times M_{\neg\varphi} \models \mathbf{FG}q$$

LTL Model Checking

Symbolic Transition System:

$$M = \langle \underbrace{V}_{\text{variables}} , \underbrace{I(V)}_{\text{Initial conditions}} , \underbrace{T(V, V')}_{\text{transition relation}} \rangle$$

Reduction to liveness

$$M \models \varphi \Leftrightarrow M \times M_{\neg\varphi} \models \mathbf{FG}q$$

⚠ Reduction to liveness: Liveness is hard!

Safety and Finite fragments of LTL

SafetyLTL:

- LTL without positive until. e.g. $\mathbf{G}(a \rightarrow \mathbf{X}p)$
- Infinite semantics, can be checked with finite semantics

Truncated Semantics of LTL [CAV03]

- Useful to represent prefixes of infinite traces

LTL_f [IJCAI13]:

- Interpret semantics over finite traces.

Safety and Finite fragments of LTL

SafetyLTL:

- LTL without positive until. e.g. $\mathbf{G}(a \rightarrow \mathbf{X}p)$
- Infinite semantics, can be checked with finite semantics

Truncated Semantics of LTL [CAV03]

- Useful to represent prefixes of infinite traces

LTL_f [IJCAI13]:

- Interpret semantics over finite traces.

$$M \models_f \varphi \Leftrightarrow M \times M_{\neg\varphi} \models_f \mathbf{G}q$$

Reduction to invariant!

Unifying Constructions for LTL Modulo Theory

	Infinite Traces $M \models \varphi$		Finite Traces $M \models_f \varphi$
	LTL	Safety Fragments	LTL_f/Truncated
Propositional	$M \times M_{\neg\varphi} \models \mathbf{FG}q$	$M \times M_{\neg\varphi} \models_f \mathbf{G}q$	
Modulo Theories	$M \times M_{\neg\varphi} \times M_{p_i \leftrightarrow \gamma_i(V)} \models \mathbf{FG}q$	$M \times M_{\neg\varphi} \times M_{p_i \leftrightarrow \gamma_i(V)} \models_f \mathbf{G}q$	
Engine	Liveness Checker	Invariant Checker	

Minor Contribution to building block: Unified automata construction for finite trace logics modulo theory in the `PyVMT` library over `vmt-lib`

Going beyond Safety

$$M \stackrel{?}{\models} \mathbf{G}p \rightarrow \mathbf{G}q$$

Assuming p *always holds, then* q *will always hold as well*

Going beyond Safety

$$M \stackrel{?}{\models} \mathbf{G}p \rightarrow \mathbf{G}q$$

Assuming p *always holds, then* q *will always hold as well*

We need to reduce to **Liveness!**

Going beyond Safety

$$M \overset{?}{\models} \mathbf{G}p \rightarrow \mathbf{G}q$$

Assuming p *always holds, then* q *will always hold as well*

We need to reduce to **Liveness!**

$$M \times \underbrace{M_{\mathbf{G}p}}_{\langle \{p\}, p, p \wedge p' \rangle} \models_f \mathbf{G}q$$

Going beyond Safety

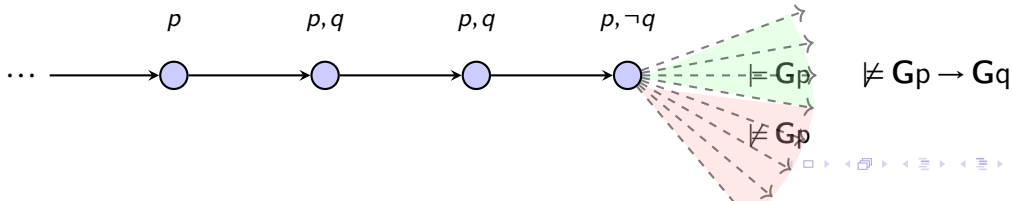
$$M \stackrel{?}{\models} \mathbf{G}p \rightarrow \mathbf{G}q$$

Assuming p *always holds, then* q *will always hold as well*

We need to reduce to **Liveness!**

$$M \times \underbrace{M_{\mathbf{G}p}}_{\langle\{p\},p,p\wedge p'\rangle} \models_f \mathbf{G}q$$

Relative Safety[Henzinger94]: *Assuming that* $\mathbf{G}p$ *holds,* $\mathbf{G}p \rightarrow \mathbf{G}q$ *can be treated as a safety property!*



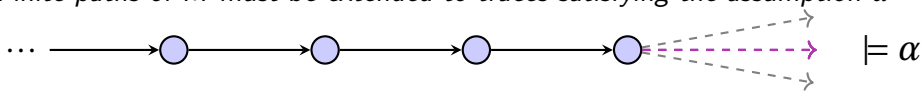
Model Checking of Relative Safety Properties (I)

I: Identify formal conditions to reduce $\overbrace{\alpha}^{\alpha_S \wedge \alpha_L} \rightarrow \overbrace{\varphi_S}^{\text{safety}}$ to invariant checking

Model Checking of Relative Safety Properties (I)

I: Identify formal conditions to reduce $\overbrace{\alpha}^{\alpha_S \wedge \alpha_L} \rightarrow \overbrace{\varphi_S}^{\text{safety}}$ to invariant checking

Finite paths of M must be extended to traces satisfying the assumption α



Reduction to invariant!

Model Checking of Relative Safety Properties (II)

II: Iterative algorithm

- Relax requirements: $(\alpha \rightarrow \varphi_S)$
- Use invariant reduction $(M \times M_\alpha \models_f \varphi_S)$
- block non-extendible counter-examples

Model Checking of Relative Safety Properties (II)

II: Iterative algorithm

- Relax requirements: $(\alpha \rightarrow \varphi_S)$
- Use invariant reduction $(M \times M_\alpha \models_f \varphi_S)$
- block non-extendible counter-examples

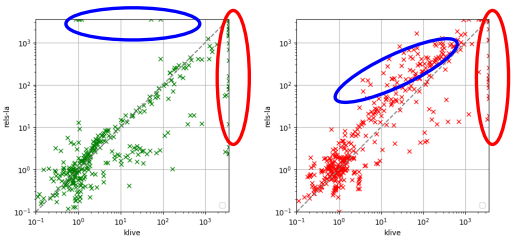
Compared with `k-liveness` and `liveness-to-safety` over models of form $\alpha \rightarrow \varphi_S$

Model Checking of Relative Safety Properties (II)

II: Iterative algorithm

- Relax requirements: $(\alpha \rightarrow \varphi_S)$
- Use invariant reduction $(M \times M_\alpha \models_f \varphi_S)$
- block non-extendible counter-examples

Compared with `k-liveness` and `liveness-to-safety` over models of form $\alpha \rightarrow \varphi_S$



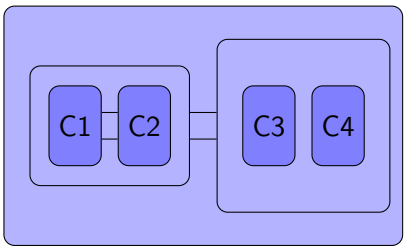
Compositional Verification

Context

Challenges: Scalability and system complexity

Context

Challenges: Scalability and system complexity



$$\models \underbrace{\varphi}_{LTL}$$

- Large component-based systems
- Component interconnected
- Complex interactions

Composition

$$\frac{M_1 \models \varphi_1, \ldots, M_n \models \varphi_n}{\gamma_S(M_1, \ldots, M_n) \models \gamma_P(\varphi_1, \ldots, \varphi_n) \quad \gamma_P(\varphi_1, \ldots, \varphi_n) \models \varphi}$$
$$\gamma_S(M_1, \ldots, M_n) \models \varphi$$

- M_i: local components
- φ_i: local properties
- φ: global property
- γ_S: component composition
- γ_P: property composition

Composition

$$\frac{M_1 \models \varphi_1, \ldots, M_n \models \varphi_n}{\gamma_S(M_1, \ldots, M_n) \models \gamma_P(\varphi_1, \ldots, \varphi_n) \quad \gamma_P(\varphi_1, \ldots, \varphi_n) \models \varphi}{\gamma_S(M_1, \ldots, M_n) \models \varphi}$$

- M_i: local components
- φ_i: local properties
- φ: global property
- γ_S: component composition
- γ_P: property composition

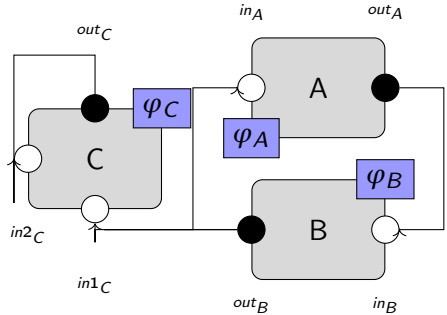
Sync case:

$$\gamma_S(M_1, \ldots, M_n) = M_1 \times \cdots \times M_n$$

$$\gamma_P(\varphi_1, \ldots, \varphi_n) = \varphi_1 \wedge \cdots \wedge \varphi_n$$

Asynchronous Composition of Interface LTL properties

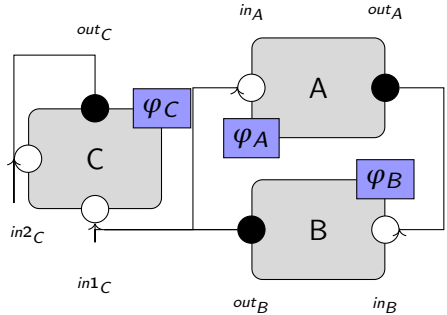
Type of composition		
Variable setup	Sync	Async
Local Vars		
I/O		X



Asynchronous Composition of Interface LTL properties

Type of composition		
Variable setup	Sync	Async
Local Vars		
I/O		X

$$\underbrace{\gamma_P(\overbrace{\varphi_C, \varphi_B, \varphi_A}^{\text{Local props}})}_{\text{Async composition}} \rightarrow \overbrace{\varphi}^{\text{global prop}}$$



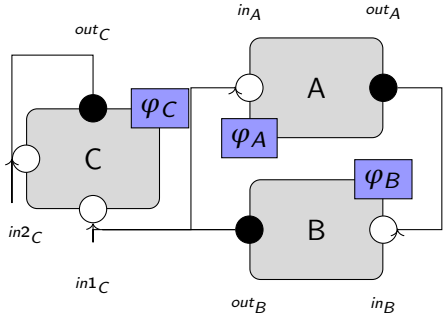
Challenges:

- Composition complicated by interleaving
- Input not controlled by the component

Asynchronous Composition of Interface LTL properties

Type of composition		
Variable setup	Sync	Async
Local Vars		
I/O		X

$$\underbrace{\gamma_P(\overbrace{\varphi_C, \varphi_B, \varphi_A}^{\text{Local props}})}_{\text{Async composition}} \rightarrow \overbrace{\varphi}^{\text{global prop}}$$



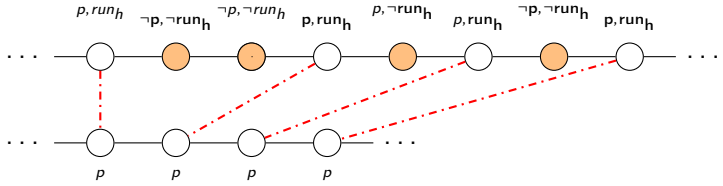
Challenges:

- Composition complicated by interleaving
- Input not controlled by the component

Related Works:

- [The operators of TLA - L. Lamport]
- [Definition of a Temporal Clock Operator - C. Eisner et al.]
- [Assume-guarantee reasoning with scheduled components. - C. Liu et al.]

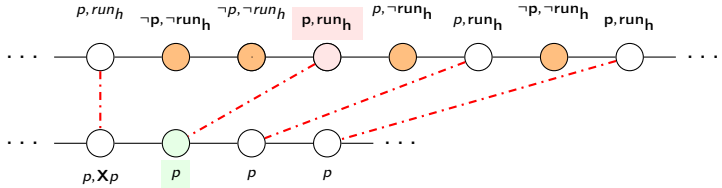
Local/ Global trace View



$\sigma \in \mathscr{L}(\gamma_S(M_1, \ldots, M_n))$

$\sigma_h \in \mathscr{L}(M_h)$

Local/ Global trace View



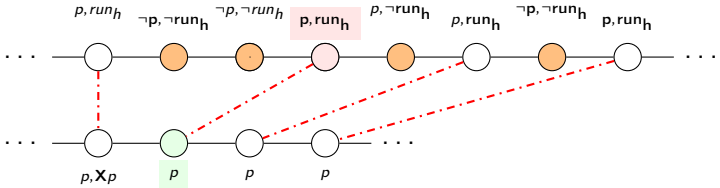
$\sigma \in \mathscr{L}(\gamma_S(M_1, \ldots, M_n))$

$\sigma_h \in \mathscr{L}(M_h)$

LOCAL REASONING: $M_h \models \varphi_h \Leftrightarrow \forall \sigma_h \in \mathscr{L}(M_h) : \sigma_h \models \varphi_h$

$\mathbf{X}p$ is local to the trace

Local/ Global trace View



$\sigma \in \mathscr{L}(\gamma_S(M_1, \ldots, M_n))$

$\sigma_h \in \mathscr{L}(M_h)$

LOCAL REASONING: $M_h \models \varphi_h \Leftrightarrow \forall \sigma_h \in \mathscr{L}(M_h) : \sigma_h \models \varphi_h$
$X p$ is local to the trace

SYNTACTIC APPROACH:

$\mathscr{R}(\varphi_h)$ s.t. $\sigma_h, i \models \varphi_h \Leftrightarrow \sigma, m_i \models \mathscr{R}(\varphi_h)$

EXAMPLE:

$\mathscr{R}(\mathbf{X}p) := \mathbf{X}(\neg run_h \, \mathsf{U} \, (run_h \wedge p))$

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}_h^*(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V_h^O}(v'=v)\right)\right)$$

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}_h^*(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V_h^O}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}_h^*(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V_h^O}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

Possibily finite semantics: Eisner-Fisman LTL Truncated Semantics

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}^*_h(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V^O_h}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

Possibily finite semantics: Eisner-Fisman LTL Truncated Semantics

Linear Size Rewriting $\mathscr{R}$ is linear w.r.t. the size of φ_h!

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}^*_h(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V^O_h}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

Possibily finite semantics: Eisner-Fisman LTL Truncated Semantics

Linear Size Rewriting $\mathscr{R}$ is linear w.r.t. the size of φ_h!

Rewriting Optimization: Systematically exploit stutter invariance

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}^*_h(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V^O_h}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

Possibily finite semantics: Eisner-Fisman LTL Truncated Semantics

Linear Size Rewriting $\mathscr{R}$ is linear w.r.t. the size of φ_h!

Rewriting Optimization: Systematically exploit stutter invariance

Implemented in OCRA: Tool for Contract Refinement

Rewriting Based Approach

$$\gamma_P(\varphi_1,\ldots,\varphi_n) := \bigwedge_{1\le h\le n}\left(\mathscr{R}^*_h(\varphi_h)\wedge\left(\neg run_h \rightarrow \bigwedge_{v\in V^O_h}(v'=v)\right)\right)$$

Support LTL modulo theory: Ad-hoc approach for v'

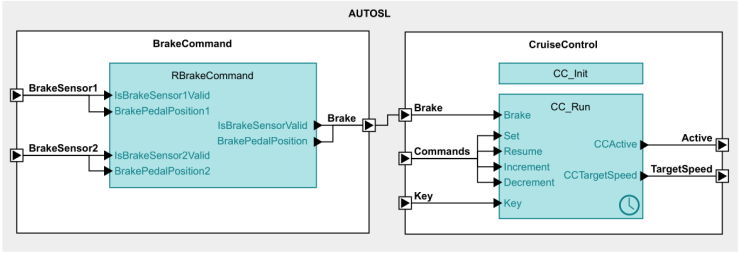
Possibily finite semantics: Eisner-Fisman LTL Truncated Semantics

Linear Size Rewriting $\mathscr{R}$ is linear w.r.t. the size of φ_h!

Rewriting Optimization: Systematically exploit stutter invariance

Implemented in OCRA: Tool for Contract Refinement

Applications



EVA [EVA: [Cimatti et al. TACAS23]:]

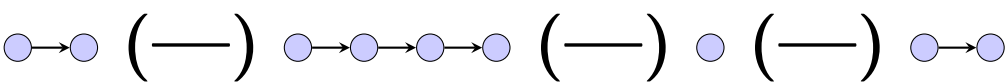
- Tool for compositional verification of *AUTOSAR* (automotive) systems
- Contract refinement using asynchronous composition
- Event-based scheduling: $\mathbf{G}(change(BrakeSens1) \rightarrow \mathbf{X}RBrakeComd.run)$
- Periodic scheduling

What about continuous time?

- **discrete time:** Time is defined as a sequence of actions
- **continuous time:** Actions are discrete while time passes densely

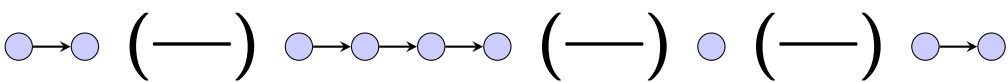
What about continuous time?

- **discrete time:** Time is defined as a sequence of actions
- **continuous time:** Actions are discrete while time passes densely



What about continuous time?

- **discrete time:** Time is defined as a sequence of actions
- **continuous time:** Actions are discrete while time passes densely



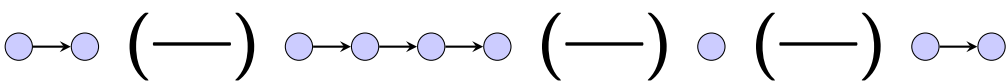
MTL:

- Extend LTL with bounds on modalities, e.g. $\mathbf{F}_{\leq 5} a$
- $\mathbf{F}_{\leq 5} a$ means *"a will become true once in at most 5 time units"*

We can lift composition to timed systems.

What about continuous time?

- **discrete time:** Time is defined as a sequence of actions
- **continuous time:** Actions are discrete while time passes densely



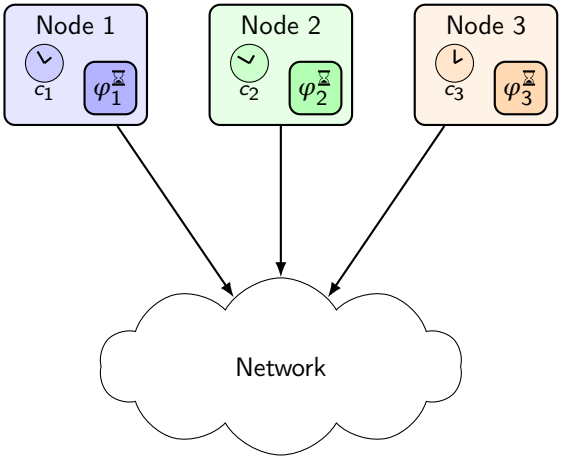
MTL:

- Extend LTL with bounds on modalities, e.g. $\mathbf{F}_{\leq 5} a$
- $\mathbf{F}_{\leq 5} a$ means *"a will become true once in at most 5 time units"*

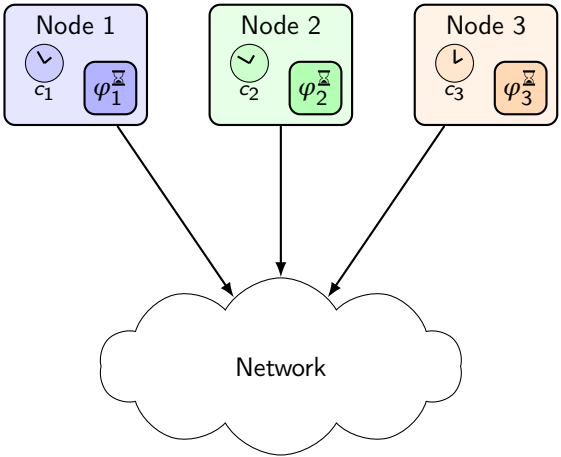
We can lift composition to timed systems.

What if local time is skewed?

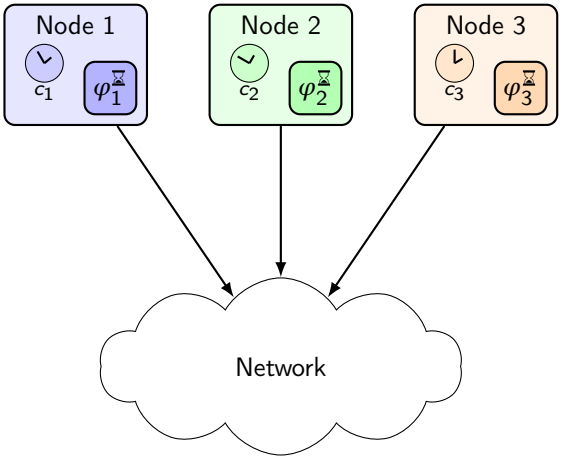
Compositional Verification of Distributed Real Time Systems



Compositional Verification of Distributed Real Time Systems

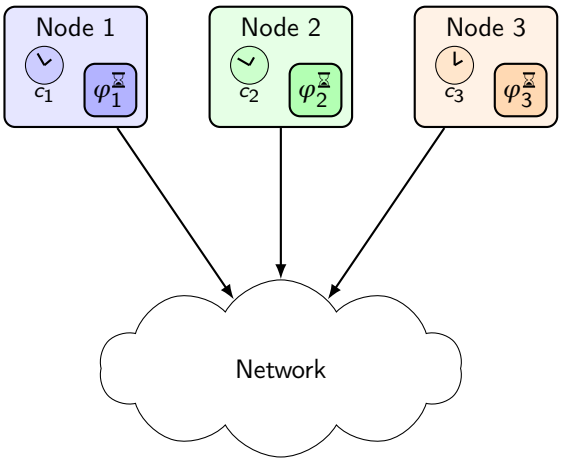


Compositional Verification of Distributed Real Time Systems



MTL with local clocks $\mathbf{G}(p \rightarrow \mathbf{F}^{c_1}_{\leq 5} q)$

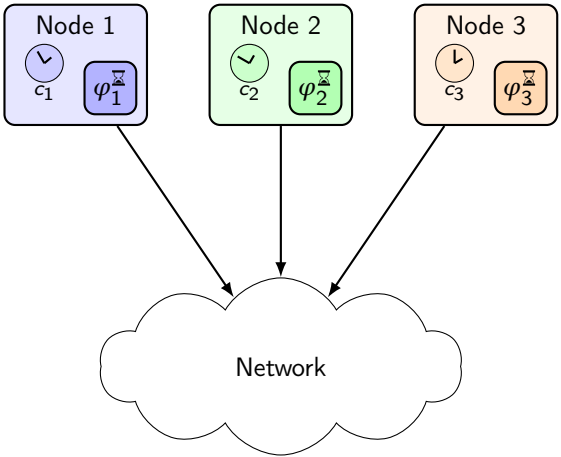
Compositional Verification of Distributed Real Time Systems



MTL with local clocks $\mathbf{G}(p \rightarrow \mathbf{F}^{c_1}_{\leq 5} q)$

Novel logics supporting resettable clock:
MTLSK ⚠non-monotonic "time"!

Compositional Verification of Distributed Real Time Systems



MTL with local clocks $\mathbf{G}(p \rightarrow \mathbf{F}^{c_1}_{\leq 5} q)$

Novel logics supporting resettable clock: `MTLSK` ⚠non-monotonic "time"!

Re-define γ_P for timed traces with $\mathscr{R}^\tau$

Hyperproperties

Motivation

Challenges: Expressiveness and Scalability

Motivation

Challenges: Expressiveness and Scalability

Information Flow Properties

- Non interference
- Non inference
- Observational Determinism
- ...

Diagnosability *With partial observability, can I detect an internal fault in my Plant?*

Motivation

Challenges: Expressiveness and Scalability

Information Flow Properties

- Non interference
- Non inference
- Observational Determinism
- ...

Diagnosability *With partial observability, can I detect an internal fault in my Plant?*

⚠**We need to reason over multiple traces at the same time!**

HyperLTL

[Clarkson, Finkbeiner, Koleini, Micinski, Rabe and Sánchez,"Temporal Logics for Hyperproperties. POST'14]

Quantifiers with trace variables: $\forall x.\varphi \quad \exists x.\varphi$

Syntax:

$$\varphi ::= \forall x.\varphi \mid \exists x.\varphi \mid \psi$$

$$\psi ::= p[x] \mid X\psi \mid G\psi \mid F\psi \mid \psi U\psi \mid \psi R\psi$$

Notes:

- HyperLTL starts with quantifiers, then temporal formula (prenex form).
- Model Checking decidable.

∀-∃ safety hyperproperties

GNI: *The high-security input of a system does not influence the observable output.*

$$\forall x \forall y \exists z. \mathbf{G} \left(\bigwedge_{h \in H} (h[x] \leftrightarrow h[z]) \wedge \bigwedge_{o \in L \cup O} (o[y] \leftrightarrow o[z]) \right)$$

∀-∃ safety hyperproperties

GNI: *The high-security input of a system does not influence the observable output.*

$$\forall x \forall y \exists z. \mathbf{G}\left(\bigwedge_{h \in H} (h[x] \leftrightarrow h[z]) \wedge \bigwedge_{o \in L \cup O} (o[y] \leftrightarrow o[z])\right)$$

Path planning: robustness, Shortest Path

$$\exists x. \forall y. (\neg goal[y])\, \mathsf{U}\, goal[x]$$

∀-∃ safety hyperproperties

GNI: *The high-security input of a system does not influence the observable output.*

$$\forall x \forall y \exists z. \mathbf{G}\left(\bigwedge_{h \in H} (h[x] \leftrightarrow h[z]) \wedge \bigwedge_{o \in L \cup O} (o[y] \leftrightarrow o[z])\right)$$

Path planning: robustness, Shortest Path

$$\exists x. \forall y. (\neg goal[y])\, \mathsf{U}\, goal[x]$$

⚠**Cannot be verified using self-composition!**

∀-∃ safety hyperproperties

GNI: *The high-security input of a system does not influence the observable output.*

$$\forall x \forall y \exists z. \mathbf{G}\left(\bigwedge_{h \in H} (h[x] \leftrightarrow h[z]) \wedge \bigwedge_{o \in L \cup O} (o[y] \leftrightarrow o[z])\right)$$

Path planning: robustness, Shortest Path

$$\exists x. \forall y. (\neg goal[y])\, \mathsf{U}\, goal[x]$$

⚠**Cannot be verified using self-composition!**

RW:MCHyper, AutoHyper, HyPro, HyperQB

Inductive reasoning for hyperproperties

Inductive invariant

1. $I(V) \rightarrow P(V)$
2. $P(V) \wedge T(V, V') \rightarrow P(V')$

Inductive reasoning for hyperproperties

Inductive invariant

1. $I(V) \rightarrow P(V)$
2. $P(V) \wedge T(V, V') \rightarrow P(V')$

Inductive Simulation

1. $I(V[x]) \rightarrow \exists V[y].I(V[y]) \wedge P(V[x], V[y])$
2. $P(V[x], V[y]) \wedge T(V[x], V'[x]) \rightarrow$
   $\exists V'[y].T(V[y], V'[y]) \wedge P(V'[x], V'[y])$

Inductive reasoning for hyperproperties

Inductive invariant

1. $I(V) \rightarrow P(V)$
2. $P(V) \wedge T(V, V') \rightarrow P(V')$

Inductive Simulation

1. $I(V[x]) \rightarrow \exists V[y].I(V[y]) \wedge P(V[x], V[y])$
2. $P(V[x], V[y]) \wedge T(V[x], V'[x]) \rightarrow$
   $\exists V'[y].T(V[y], V'[y]) \wedge P(V'[x], V'[y])$

$$M = \langle \overbrace{\{p, q, r\}}^{V}, \overbrace{p \wedge r}^{I}, \overbrace{(p \rightarrow p')}^{T} \rangle$$

Inductive reasoning for hyperproperties

Inductive invariant

1. $I(V) \rightarrow P(V)$
2. $P(V) \wedge T(V, V') \rightarrow P(V')$

Inductive Simulation

1. $I(V[x]) \rightarrow \exists V[y].I(V[y]) \wedge P(V[x], V[y])$
2. $P(V[x], V[y]) \wedge T(V[x], V'[x]) \rightarrow$
   $\exists V'[y].T(V[y], V'[y]) \wedge P(V'[x], V'[y])$

$$M = \langle \overbrace{\{p, q, r\}}^{V}, \overbrace{p \wedge r}^{I}, \overbrace{(p \rightarrow p')}^{T} \rangle$$

p is an inductive invariant

Inductive reasoning for hyperproperties

Inductive invariant

1. $I(V) \rightarrow P(V)$
2. $P(V) \wedge T(V, V') \rightarrow P(V')$

Inductive Simulation

1. $I(V[x]) \rightarrow \exists V[y].I(V[y]) \wedge P(V[x], V[y])$
2. $P(V[x], V[y]) \wedge T(V[x], V'[x]) \rightarrow$
   $\exists V'[y].T(V[y], V'[y]) \wedge P(V'[x], V'[y])$

$$M = \langle \overbrace{\{p, q, r\}}^{V}, \overbrace{p \wedge r}^{I}, \overbrace{(p \rightarrow p')}^{T} \rangle$$

p is an inductive invariant

$p[x] \wedge q[y]$ is an inductive simulation

HyperK-induction

k-inductive case: Use BMC[TACAS21] as base case and a $\forall$-$\exists$ k-inductive proof obligation. **COMPLETE!**

HyperK-induction

k-inductive case: Use BMC[TACAS21] as base case and a ∀-∃ k-inductive proof obligation. **COMPLETE!**

Generalisation:counter-example to induction + interpolation

Incremental solving (⚠limited)

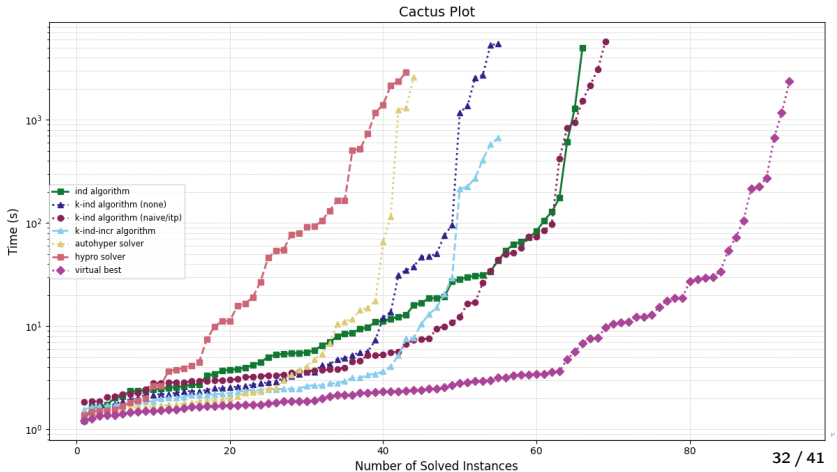
HyperK-induction

k-inductive case: Use BMC[TACAS21] as base case and a ∀-∃ k-inductive proof obligation. **COMPLETE!**

Generalisation: counter-example to induction + interpolation

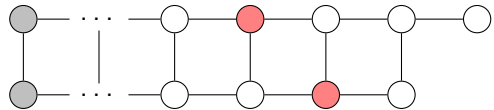
Incremental solving (⚠ limited)

Competitive tool: (⚠ generalisation still limited)



From Sync to Async

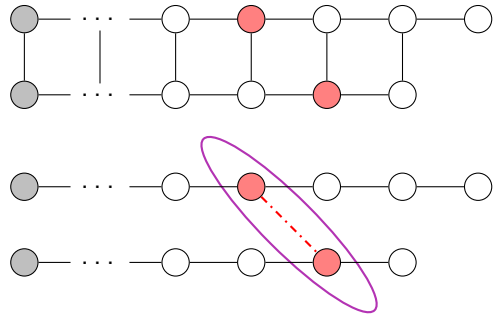
HyperLTL considers trace running in lockstep.



From Sync to Async

HyperLTL considers trace running in lockstep.

Often we are interesting in relating different observation points of the executions.



Asynchronous Hyperlogics: various logics have been defined to model asynchrony in Hyperproperties: e.g., A-HLTL, H_μ, ObsHyperLTL

Combining Stuttering and Context: $GHLTL_{SC}$

Stuttering HyperLTL[LICS21]

$$\forall x \exists y.(\mathbf{G}_{\Gamma}(p[x] \leftrightarrow q[y]))$$

Γ *restricts evaluation over observation points*

Combining Stuttering and Context: $GHLTL_{SC}$

Stuttering HyperLTL[LICS21]

$$\forall x \exists y.(\mathbf{G}_{\Gamma}(p[x] \leftrightarrow q[y]))$$

Γ *restricts evaluation over observation points*

Context HyperLTL[LICS21]

$$\forall x \exists y \exists z.\mathbf{G}(p[x] \rightarrow \langle x, y \rangle \mathbf{F}(p[x] \leftrightarrow q[y]))$$

Synchronous over "context trace variables" $C = \{x, \ldots\}$

Combining Stuttering and Context: $GHLTL_{SC}$

Stuttering HyperLTL[LICS21]

$$\forall x \exists y.(\mathbf{G}_{\Gamma}(p[x] \leftrightarrow q[y]))$$

Γ *restricts evaluation over observation points*

Context HyperLTL[LICS21]

$$\forall x \exists y \exists z.\mathbf{G}(p[x] \rightarrow \langle x, y \rangle \mathbf{F}(p[x] \leftrightarrow q[y]))$$

Synchronous over "context trace variables" $C = \{x, \ldots\}$

Generalized HyperLTL$_{SC}$

Combining Stuttering and Context: $GHLTL_{SC}$

Stuttering HyperLTL[LICS21]

$$\forall x \exists y.(\mathbf{G}_{\Gamma}(p[x] \leftrightarrow q[y]))$$

Γ *restricts evaluation over observation points*

Context HyperLTL[LICS21]

$$\forall x \exists y \exists z.\mathbf{G}(p[x] \rightarrow \langle x, y \rangle \mathbf{F}(p[x] \leftrightarrow q[y]))$$

Synchronous over "context trace variables" $C = \{x, \ldots\}$

Generalized HyperLTL$_{SC}$

- With Past Operators
- Non prenex quantification
- Expressive Decidable fragment: Simple GHyperLTL$_{S+C}$

Combining Stuttering and Context: $GHLTL_{SC}$

Stuttering HyperLTL[LICS21]

$$\forall x \exists y.(\mathbf{G}_{\Gamma}(p[x] \leftrightarrow q[y]))$$

Γ *restricts evaluation over observation points*

Context HyperLTL[LICS21]

$$\forall x \exists y \exists z.\mathbf{G}(p[x] \rightarrow \langle x, y \rangle \mathbf{F}(p[x] \leftrightarrow q[y]))$$

Synchronous over "context trace variables" $C = \{x, \ldots\}$

Generalized HyperLTL$_{SC}$

- With Past Operators
- Non prenex quantification
- Expressive Decidable fragment: Simple GHyperLTL$_{S+C}$

- Can express non-regular properties
- Subsumes KLTL (one agent) with both Synchronous and asynchronous semantics.
- Express **Finite Delay** diagnosability.

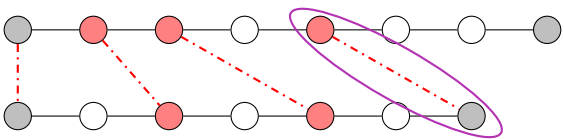
SC-HyperLTL

Considered a prenex fragment of Simple GHyperLTL$_{S+C}$

SC-HyperLTL

Considered a prenex fragment of Simple GHyperLTL$_{S+C}$

Studied with finite trace semantics



Traces can have different length

SC-HyperLTL

Considered a prenex fragment of Simple GHyperLTL$_{S+C}$

Studied with finite trace semantics

Practical Model Checking:

Stuttering extension:

- **Idea:** Reduce $\forall^*/\exists^*$ to asynchronous composition with $\mathscr{R}$
- In principle similar to the approach used for A-HLTL [Baumeister, Coenen, Bonakdarpour, Finkbeiner and Sánchez. "A Temporal Logic for Asynchronous Hyperproperties." CAV'21.]

Bounded observations:

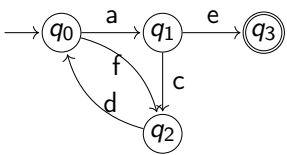
- Support $\forall\exists$ fixing at most k observation points
- Traces can be unbounded
- Reduces to HyperLTL

Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

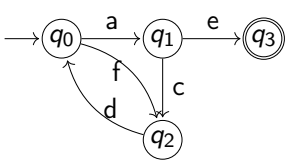
Model Checking via stuttering encoding: Overall idea

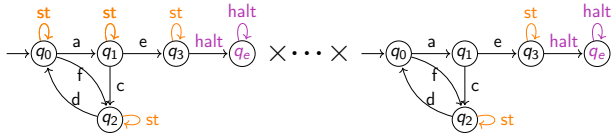
Asynchronous self-composition of M



Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

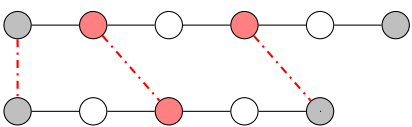




Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

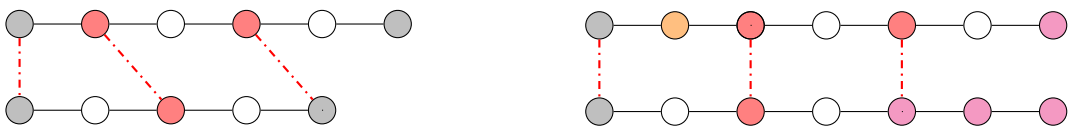
Align traces over Γ-points



Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

Align traces over Γ-points



$$\mathbf{G}(\neg end[x] \wedge \neg end[y] \rightarrow (\Phi_\Gamma[x] \leftrightarrow \Phi_\Gamma[y]))$$

Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

Align traces over Γ-points

Rewrite temporal operators to evaluate temporal operators at observation points synchronously

$\langle x \rangle \beta[x] \Rightarrow \mathscr{R}_{st[x] \vee halt[x]}(\beta[x])$
$\varphi_1 U \varphi_2 \Rightarrow \mathscr{R}_{\Phi_\Gamma}(\varphi_1 U \varphi_2)$

$\Phi_\Gamma := \bigvee_{\theta \in \Gamma}(Y\theta \leftrightarrow \neg\theta) \vee init \vee last$

Model Checking via stuttering encoding: Overall idea

Asynchronous self-composition of M

Align traces over Γ-points

Rewrite temporal operators to evaluate temporal operators at observation points synchronously

Conclusions

Summary

Framework for LTL($\mathcal{T}$) with finite and infinite traces with *relative safety* properties

Compositional rewriting approach for asynchronous systems over *discrete* and *timed* domain

- Inductive construction for the verification of $\forall\exists$ hyperproperties
- Novel expressive and *decidable* asynchronous hyperlogic over infinite and finite traces.

Contextualized Contributions

LTL Model Checking:
iFM23,JPK60

Compositional Verification:
NFM22,LMCS25 DATE23,NFM23

Hyperproperty Verification:
KIND, FSTTCS24, SPIN25

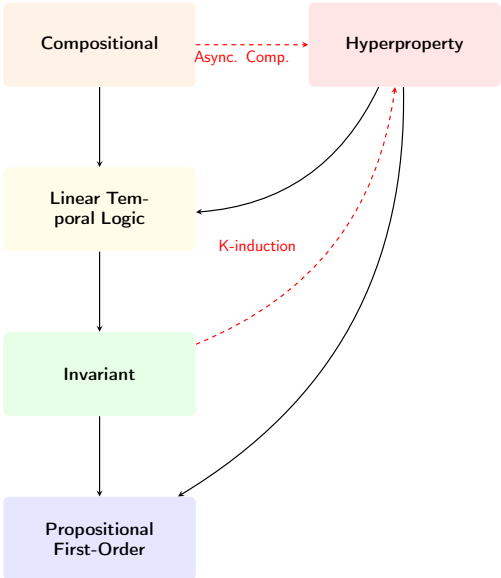
Contextualized Contributions

LTL Model Checking:
iFM23,JPK60

Compositional Verification:
NFM22,LMCS25 DATE23,NFM23

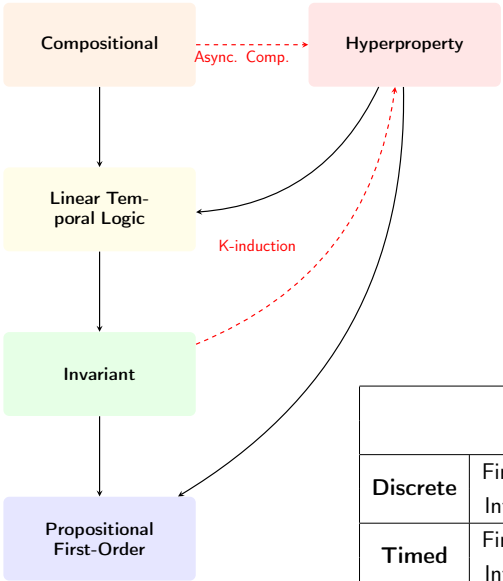
Hyperproperty Verification:
KIND, FSTTCS24, SPIN25

	LTLMC	Compositional Verif.	Hyperproperty
Scalability	✓	✓	✓
Expressiveness	Indirect	✓	✓
System Complexity	Indirect	✓	✓



Contribution relations

- Contributions viewable in a larger SMT-based framework
- Compositional reasoning reduces to LTL verification
- Certain fragments of hyperproperties reduces to LTL
- (red dashed lines) Conceptual inspirations



Contribution relations

- Contributions viewable in a larger SMT-based framework
- Compositional reasoning reduces to LTL verification
- Certain fragments of hyperproperties reduces to LTL
- (red dashed lines) Conceptual inspirations

		Trace Properties		Hyperproperties	
		Sync MC	Async Comp.	Sync MC	Async MC
Discrete	Fin. Trace	✓	✓	✓	✓
	Inf. Trace	✓	✓	✓	✓
Timed	Fin. Trace				
	Inf. Trace	✓	✓		

Future Works

Improve algorithms:

- relative safety
- hyper K-induction

Study settings that were not considered