CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Coinductive Proofs for Temporal (Hyper)Liveness

Arthur Correnson | Cambium Seminar | September 26, 2025

Context and Motivation

Context

This talk is about **temporal** properties of **reactive programs**.

$$\Box(req \rightarrow \Diamond res)$$

Context

This talk is about **temporal** properties of **reactive programs**.

$$\Box(req \rightarrow \Diamond res)$$

Specifically, about temporal **hyperproperties**...

$$\forall \pi_1 . \forall \pi_2 . \; \Box(in_{\pi_1} = in_{\pi_2} \rightarrow out_{\pi_1} = out_{\pi_2})$$

Context

This talk is about **temporal** properties of **reactive programs**.

$$\Box(req \rightarrow \Diamond res)$$

Specifically, about temporal **hyperproperties**...

$$\forall \pi_1 . \forall \pi_2 .\ \Box(in_{\pi_1} = in_{\pi_2} \rightarrow out_{\pi_1} = out_{\pi_2})$$

... and how to verify them using **interactive proof assistants** (Rocq).

A Motivating Example

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

A Motivating Example

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

Question: Is this program *secure* in the sense that it never *reveals* the value of secret inputs via outputs?

A Motivating Example

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

Question: Is this program *secure* in the sense that it never *reveals* the value of secret inputs via outputs?

Answer: It depends on your point of view!

Attempt 1: Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

Attempt 1: Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

We could require outputs to be strictly independent from secrets (non-interference).

$$\forall \pi_1.\, \forall \pi_2.\, \square(\mathtt{pub}_{\pi_1} = \mathtt{pub}_{\pi_2} \rightarrow \mathtt{out}_{\pi_1} = \mathtt{out}_{\pi_2})$$

Attempt 1: Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

We could require outputs to be strictly independent from secrets (non-interference).

$$\forall \pi_1.\ \forall \pi_2.\ \square(\mathtt{pub}_{\pi_1} = \mathtt{pub}_{\pi_2} \rightarrow \mathtt{out}_{\pi_1} = \mathtt{out}_{\pi_2})$$

This program **violates** non-interference: it can produce different outputs for the same public inputs.

Attempt 2: A Weaker Notion of Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

Attempt 2: A Weaker Notion of Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

We could allow dependencies, so long as they are “not informative” (non-inference)

$$\forall \pi_1.\, \exists \pi_2.\, \Box(\mathtt{pub}_{\pi_1} = \mathtt{pub}_{\pi_2} \wedge \mathtt{sec}_{\pi_2} = 0 \wedge \mathtt{out}_{\pi_1} = \mathtt{out}_{\pi_2})$$

> *“Any input/output difference can be observed with the secret fixed to be* 0*”*

Attempt 2: A Weaker Notion of Non-interference

```
loop {
  pub ← input()
  sec ← input()
  noi ← havoc()
  out ← pub + sec + noi
  output(out)
}
```

We could allow dependencies, so long as they are "not informative" (non-inference)

$$\forall \pi_1.\, \exists \pi_2.\, \Box(\mathtt{pub}_{\pi_1} = \mathtt{pub}_{\pi_2} \wedge \mathtt{sec}_{\pi_2} = 0 \wedge \mathtt{out}_{\pi_1} = \mathtt{out}_{\pi_2})$$

> *"Any input/output difference can be observed with the secret fixed to be* 0*"*

This program **satisfies** non-inference.

Observations

Observation 1. Temporal hyperproperties of the form ∀∃ are extremely difficult to verify fully automatically...

Observations

Observation 1. Temporal hyperproperties of the form ∀∃ are extremely difficult to verify fully automatically...

Observation 2. ... but there is still a lack of deductive proof techniques for such hyperproperties.

Observations

Observation 1. Temporal hyperproperties of the form ∀∃ are extremely difficult to verify fully automatically...

Observation 2. ... but there is still a lack of deductive proof techniques for such hyperproperties.

Contribution: We present HyCo, a framework for the deductive verification of ∀∃ hyperproperties based on coinduction.

Preliminaries

System Models

We are modelling programs as event-labeled transition systems.

$$(\mathcal{S}, \mathcal{E}, \rightarrow) \qquad \begin{cases} \mathcal{S} \text{ set of states} \\ \mathcal{E} \text{ set of events} \\ \rightarrow \subseteq \mathcal{S} \times \mathcal{E} \times \mathcal{S} \text{ transition relation} \end{cases}$$

System Models

We are modelling programs as event-labeled transition systems.

$$(\mathcal{S}, \mathcal{E}, \rightarrow) \qquad \begin{cases} \mathcal{S} \text{ set of states} \\ \mathcal{E} \text{ set of events} \\ \rightarrow \subseteq \mathcal{S} \times \mathcal{E} \times \mathcal{S} \text{ transition relation} \end{cases}$$

$Traces(s) \subseteq \mathcal{E}^{\omega}$ is the set of *infinite traces* emanating from a state $s \in \mathcal{S}$.

Specifications

Syntax

(Quantifier Prefixes) $\underbrace{\forall \ldots \forall}_{n} \underbrace{\exists \ldots \exists}_{m} \psi$

(Trace Relations) $\psi \subseteq (\mathcal{E}^{\omega})^{n+m}$

Specifications

Syntax

(Quantifier Prefixes) $\underbrace{\forall \ldots \forall}_{n} \underbrace{\exists \ldots \exists}_{m} \psi$

(Trace Relations) $\psi \subseteq (\mathcal{E}^{\omega})^{n+m}$

Semantics (Example with $\forall\exists$)

$$(s_1, s_2) \models \forall\exists\psi \iff \forall \pi_1 \in \mathit{Traces}(s_1).\exists \pi_2 \in \mathit{Traces}(s_2).\psi(\pi_1, \pi_2)$$

Trace Relations Expressed With Modalities

Syntax

(Temporal Relations) $\psi ::= \Box\psi \mid \Diamond\psi \mid \bigcirc\psi \mid \neg\psi \mid \psi \vee \psi \mid \varphi \mid \dots$

(Event Relations) $\varphi \subseteq \mathcal{E}^{n+m}$

Trace Relations Expressed With Modalities

Syntax

$$\text{(Temporal Relations)} \quad \psi ::= \Box\psi \mid \Diamond\psi \mid \bigcirc\psi \mid \neg\psi \mid \psi \vee \psi \mid \varphi \mid \dots$$

$$\text{(Event Relations)} \quad \varphi \subseteq \mathcal{E}^{n+m}$$

Semantics (Example of the binary case)

$$\begin{aligned}
\psi(\pi_1, \pi_2) &\iff (\pi_1, \pi_2) \models^0 \psi \\
(\pi_1, \pi_2) \models^i \Box\psi &\iff \forall j \in \mathbb{N}.\ (\pi_1, \pi_2) \models^{i+j} \psi \\
(\pi_1, \pi_2) \models^i \Diamond\psi &\iff \exists j \in \mathbb{N}.\ (\pi_1, \pi_2) \models^{i+j} \psi \\
(\pi_1, \pi_2) \models^i \bigcirc\psi &\iff (\pi_1, \pi_2) \models^{i+1} \psi \\
(\pi_1, \pi_2) \models^i \varphi &\iff \varphi(\pi_1(i), \pi_2(i))
\end{aligned}$$

Challenges

The challenge(s) of $\forall\exists$

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

The challenge(s) of ∀∃

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

Formally, we need three ingredients:

The challenge(s) of ∀∃

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

Formally, we need three ingredients:

1. A function f producing some π_2 from an arbitrary π_1

The challenge(s) of ∀∃

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

Formally, we need three ingredients:

1. A function f producing some π_2 from an arbitrary π_1
2. A proof that $f(\mathit{Traces}(s_1)) \subseteq \mathit{Traces}(s_2)$

The challenge(s) of ∀∃

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

Formally, we need three ingredients:

1. A function f producing some π_2 from an arbitrary π_1
2. A proof that $f(\mathit{Traces}(s_1)) \subseteq \mathit{Traces}(s_2)$
3. A proof that for any $\pi_1 \in \mathit{Traces}(s_1)$, $\psi(\pi_1, f(\pi_1))$

The challenge(s) of ∀∃

Suppose we wish to (constructively) prove

$$\forall \pi_1 \in \mathit{Traces}(s_1).\ \exists \pi_2 \in \mathit{Traces}(s_2).\ \psi(\pi_1, \pi_2)$$

Formally, we need three ingredients:

1. A function f producing some π_2 from an arbitrary π_1
2. A proof that $f(\mathit{Traces}(s_1)) \subseteq \mathit{Traces}(s_2)$
3. A proof that for any $\pi_1 \in \mathit{Traces}(s_1)$, $\psi(\pi_1, f(\pi_1))$

Ideally, a good proof technique should be **incremental**, enabling to simultaneously construct f and proofs of $2.$ and $3.$

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

$\Longrightarrow$ provides confidence in the correctness of proofs

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

- $\Longrightarrow$ provides confidence in the correctness of proofs
- $\Longrightarrow$ facilitates reasoning about programs with *hard* features (concurrency, higher-order, etc)

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

- $\Longrightarrow$ provides confidence in the correctness of proofs
- $\Longrightarrow$ facilitates reasoning about programs with *hard* features (concurrency, higher-order, etc)

However, most of the focus is on Pre/Post specifications for concurrent or sequential programs.

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

- $\Longrightarrow$ provides confidence in the correctness of proofs
- $\Longrightarrow$ facilitates reasoning about programs with *hard* features (concurrency, higher-order, etc)

However, most of the focus is on Pre/Post specifications for concurrent or sequential programs.

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

- ⟹ provides confidence in the correctness of proofs
- ⟹ facilitates reasoning about programs with *hard* features (concurrency, higher-order, etc)

However, most of the focus is on Pre/Post specifications for concurrent or sequential programs.

- ⟹ limited support for reactive systems

Mechanized Program Logics

There is a large body of work on the interactive verification of programs using proof assistants (Iris, VST, CFML, etc).

- ⟹ provides confidence in the correctness of proofs
- ⟹ facilitates reasoning about programs with *hard* features (concurrency, higher-order, etc)

However, most of the focus is on Pre/Post specifications for concurrent or sequential programs.

- ⟹ limited support for reactive systems
- ⟹ no support for temporal (hyper)properties

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

✓ Embedded in the Rocq proof assistant

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

- ✓ Embedded in the Rocq proof assistant
- ✓ Supports incremental proofs

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

- ✓ Embedded in the Rocq proof assistant
- ✓ Supports incremental proofs
- ✓ Programming language agnostic

Contribution

We present HyCo (**Hy**perproperties, **Co**inductively), a framework for deductive proofs of temporal hyperproperties [CF25].

- ✓ Embedded in the Rocq proof assistant
- ✓ Supports incremental proofs
- ✓ Programming language agnostic
- ✓ Specification language agnostic

The Case of Safety

The Case of Invariants

We start by focusing on the case of **relational invariants** $\varphi \subseteq \mathcal{E} \times \mathcal{E}$.

$$\forall\exists\Box\varphi$$

The Case of Invariants

We start by focusing on the case of **relational invariants** $\varphi \subseteq \mathcal{E} \times \mathcal{E}$.

$$\forall\exists\Box\varphi$$

In this case, a simple **simulation** argument suffices!

The Case of Invariants

We start by focusing on the case of **relational invariants** $\varphi \subseteq \mathcal{E} \times \mathcal{E}$.

$$\forall\exists\Box\varphi$$

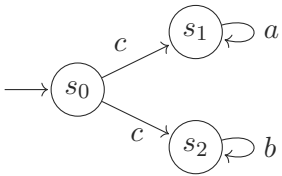
In this case, a simple **simulation** argument suffices!

To prove that $(s_1, s_2) \models \forall\exists\Box\varphi$, it suffices to find a relation $R \subseteq \mathcal{S} \times \mathcal{S}$ such that:

$$\begin{array}{ll} \text{(init)} & R(s_1, s_2) \\ \text{(step)} & R(s_1, s_2) \implies \forall s_1 \xrightarrow{e_1} s_1'.\exists s_2 \xrightarrow{e_2} s_2'.\ \varphi(e_1, e_2) \wedge R(s_1', s_2') \end{array}$$

A First Example

System:

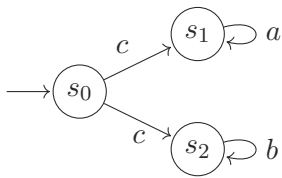


Specification: $(s_0, s_0) \models \forall\exists \Box \varphi$

$$\varphi(e_1, e_2) \triangleq e_1 = a \iff e_2 = b$$

A First Example

System:



Specification: $(s_0, s_0) \models \forall\exists \square \varphi$

$$\varphi(e_1, e_2) \triangleq e_1 = a \iff e_2 = b$$

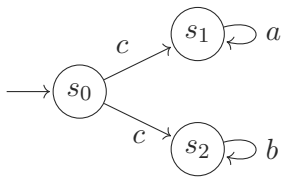
Proof: Pick R such that

$$R(s_0, s_0) \wedge R(s_1, s_2) \wedge R(s_2, s_1)$$

In terms of transitions

A First Example

System:



Specification: $(s_0, s_0) \models \forall\exists \Box\varphi$

$$\varphi(e_1, e_2) \triangleq e_1 = a \iff e_2 = b$$

Proof: Pick R such that

$$R(s_0, s_0) \wedge R(s_1, s_2) \wedge R(s_2, s_1)$$

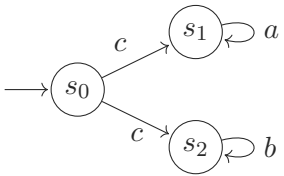
In terms of transitions

$s_0 \xrightarrow{c} s_1$ is matched with $s_0 \xrightarrow{c} s_2$

$s_0 \xrightarrow{c} s_2$ is matched with $s_0 \xrightarrow{c} s_1$

A First Example

System:



Specification: $(s_0, s_0) \models \forall\exists \Box\varphi$

$$\varphi(e_1, e_2) \triangleq e_1 = a \iff e_2 = b$$

Proof: Pick R such that

$$R(s_0, s_0) \wedge R(s_1, s_2) \wedge R(s_2, s_1)$$

In terms of transitions

$s_0 \xrightarrow{c} s_1$ is matched with $s_0 \xrightarrow{c} s_2$

$s_0 \xrightarrow{c} s_2$ is matched with $s_0 \xrightarrow{c} s_1$

$s_1 \xrightarrow{a} s_1$ is matched with $s_2 \xrightarrow{b} s_2$

$s_2 \xrightarrow{b} s_2$ is matched with $s_1 \xrightarrow{a} s_1$

A Deductive System for the Case of Invariants

The simulation approach is still very tedious: we need to guess a suitable R *up-front*. Instead, we wish to construct R *incrementally*.

A Deductive System for the Case of Invariants

The simulation approach is still very tedious: we need to guess a suitable R *up-front*. Instead, we wish to construct R *incrementally*.

To achieve this, we propose an incremental proof system based on parameterized coinduction [Hur+13].

$$\boxed{H} \vdash_{\mathtt{inv}} s_1 \;{}^{\forall} \mid \; s_2 \;{}^{\exists} \{ \varphi \}$$
$$\lceil \bar{H} \rfloor \vdash_{\mathtt{inv}} s_1 \;{}^{\forall} \mid \; s_2 \;{}^{\exists} \{ \varphi \}$$

proof quadruples

$$\begin{cases} H \subseteq \mathcal{S} \times \mathcal{S} & \text{is a } \textit{guarded context} \\ s_1, s_2 \in \mathcal{S} & \text{are states} \\ \varphi \subseteq \mathcal{E} \times \mathcal{E} & \text{is the desired invariant} \end{cases}$$

Intuition

Intuition behind $\boxed{H} \vdash_{\texttt{inv}} s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \varphi \}$:

Intuition

Intuition behind $\boxed{H} \vdash_{\texttt{inv}} s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \varphi \}$:

For every $s_1 \xrightarrow{e_1} s_1'$, there exists some $s_2 \xrightarrow{e_2} s_2'$ such that:

1. φ is preserved by the current transition: $\varphi(e_1, e_2)$
2. the quadruple $\lceil \bar{H} \rceil \vdash_{\texttt{inv}} s_1' \;^{\forall} \mid \; s_2' \;^{\exists} \{ \varphi \}$ holds

Intuition

Intuition behind $\boxed{H} \vdash_{\texttt{inv}} s_1 \ {}^{\forall} \mid \ s_2 \ {}^{\exists} \{ \varphi \}$:

For every $s_1 \xrightarrow{e_1} s_1'$, there exists some $s_2 \xrightarrow{e_2} s_2'$ such that:

1. φ is preserved by the current transition: $\varphi(e_1, e_2)$
2. the quadruple $\lceil \bar{H} \rceil \vdash_{\texttt{inv}} s_1' \ {}^{\forall} \mid \ s_2' \ {}^{\exists} \{ \varphi \}$ holds

Intuition behind $\lceil \bar{H} \rceil \vdash_{\texttt{inv}} s_1 \ {}^{\forall} \mid \ s_2 \ {}^{\exists} \{ \varphi \}$:

Either $(s_1, s_2) \in H$, or $\boxed{H} \vdash_{\texttt{inv}} s_1 \ {}^{\forall} \mid \ s_2 \ {}^{\exists} \{ \varphi \}$

Proof Rules

STEP

$$\frac{\forall s_1 \xrightarrow{e_1} s_1'.\ \exists s_2 \xrightarrow{e_2} s_2'.\ \varphi(e_1, e_2) \land \lceil \bar{H} \rceil \vdash_{\texttt{inv}} s_1'\ {}^{\forall}\ |\ s_2'\ {}^{\exists}\ \{\ \varphi\ \}}{\boxed{H} \vdash_{\texttt{inv}} s_1\ {}^{\forall}\ |\ s_2\ {}^{\exists}\ \{\ \varphi\ \}}$$

Proof Rules

Step

$$\frac{\forall s_1 \xrightarrow{e_1} s_1'.\ \exists s_2 \xrightarrow{e_2} s_2'.\ \varphi(e_1, e_2) \land \boxed{H} \vdash_{\texttt{inv}} s_1'\ ^{\forall}\ |\ \ s_2'\ ^{\exists}\ \{\ \varphi\ \}}{\boxed{H} \vdash_{\texttt{inv}} s_1\ ^{\forall}\ |\ \ s_2\ ^{\exists}\ \{\ \varphi\ \}}$$

Invariant

$$\frac{(s_1, s_2) \in H' \qquad \forall (s_1', s_2') \in H', \boxed{H \cup H'} \vdash_{\texttt{inv}} s_1'\ ^{\forall}\ |\ \ s_2'\ ^{\exists}\ \{\ \varphi\ \}}{\boxed{H} \vdash_{\texttt{inv}} s_1\ ^{\forall}\ |\ \ s_2\ ^{\exists}\ \{\ \varphi\ \}}$$

Proof Rules

Step

$$\frac{\forall s_1 \xrightarrow{e_1} s_1'.\ \exists s_2 \xrightarrow{e_2} s_2'.\ \varphi(e_1, e_2) \land \boxed{\bar{H}} \vdash_{\texttt{inv}} s_1'\ ^{\forall}\ |\ \ s_2'\ ^{\exists}\ \{\ \varphi\ \}}{\boxed{H} \vdash_{\texttt{inv}} s_1\ ^{\forall}\ |\ \ s_2\ ^{\exists}\ \{\ \varphi\ \}}$$

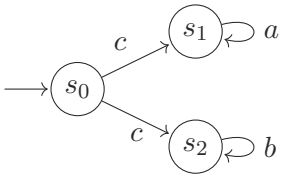
Invariant

$$\frac{(s_1, s_2) \in H' \qquad \forall (s_1', s_2') \in H', \boxed{H \cup H'} \vdash_{\texttt{inv}} s_1'\ ^{\forall}\ |\ \ s_2'\ ^{\exists}\ \{\ \varphi\ \}}{\boxed{H} \vdash_{\texttt{inv}} s_1\ ^{\forall}\ |\ \ s_2\ ^{\exists}\ \{\ \varphi\ \}}$$

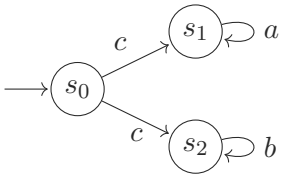
Cycle

$$\frac{(s_1, s_2) \in H}{\boxed{\bar{H}} \vdash_{\texttt{inv}} s_1\ ^{\forall}\ |\ \ s_2\ ^{\exists}\ \{\ \varphi\ \}}$$

Example Proof

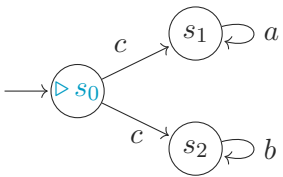


Example Proof

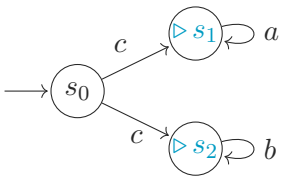


Example Proof

$$\boxed{\emptyset} \vdash_{\texttt{inv}} s_0 \,{}^{\forall} \mid \; s_0 \,{}^{\exists} \{ \varphi \}$$



Example Proof

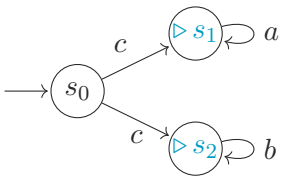


$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_0 \,{}^{\forall} \mid \; s_0 \,{}^{\exists} \{\, \varphi \,\}$$

By STEP

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{\, \varphi \,\} \wedge \ldots$$

Example Proof



$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_0 \,{}^{\forall} \mid \; s_0 \,{}^{\exists} \{\, \varphi \,\}$$

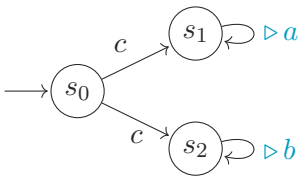
By STEP

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{\, \varphi \,\} \wedge \ldots$$

By INVARIANT with $H' = \{\, (s_1, s_2) \,\}$

$$\boxed{(s_1, s_2)} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{\, \varphi \,\} \wedge \ldots$$

Example Proof



$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_0 \,^{\forall} \mid\ s_0 \,^{\exists} \{\, \varphi \,\}$$

By Step

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid\ s_2 \,^{\exists} \{\, \varphi \,\} \wedge \ldots$$

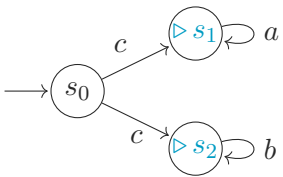
By Invariant with $H' = \{\, (s_1, s_2) \,\}$

$$\boxed{(s_1, s_2)} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid\ s_2 \,^{\exists} \{\, \varphi \,\} \wedge \ldots$$

By Step

$$\boxed{(s_1, s_2)} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid\ s_2 \,^{\exists} \{\, \varphi \,\} \wedge \ldots$$

Example Proof



$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_0 \,^{\forall} \mid \; s_0 \,^{\exists} \{ \varphi \}$

By Step

$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \varphi \} \wedge \ldots$

By Invariant with $H' = \{ (s_1, s_2) \}$

$\boxed{(s_1, s_2)} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \varphi \} \wedge \ldots$

By Step

$\boxed{(s_1, s_2)} \vdash_{\mathtt{inv}} s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \varphi \} \wedge \ldots$

By Cycle

$(s_1, s_2) \in \{(s_1, s_2)\}$

Soundness

Theorem (Soundness)

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{ \varphi \} \implies (s_1, s_2) \models \forall\exists \square \varphi$$

Soundness

Theorem (Soundness)

$$\boxed{\emptyset} \vdash_{\texttt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{ \varphi \} \implies (s_1, s_2) \models \forall\exists \Box \varphi$$

Remark. The proof is particularly difficult to mechanize in Rocq, as it requires extracting a witness from a quadruple (in `Prop`).

Soundness

Theorem (Soundness)

$$\boxed{\emptyset} \vdash_{\texttt{inv}} s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{ \varphi \} \implies (s_1, s_2) \models \forall\exists \Box \varphi$$

Remark. The proof is particularly difficult to mechanize in Rocq, as it requires extracting a witness from a quadruple (in `Prop`).

Soundness

Theorem (Soundness)

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall}\, |\; s_2 \,{}^{\exists}\, \{\, \varphi \,\} \implies (s_1, s_2) \models \forall\exists \Box \varphi$$

Remark. The proof is particularly difficult to mechanize in Rocq, as it requires extracting a witness from a quadruple (in `Prop`).

→ The original proof uses the axiom of functional choice.

Soundness

Theorem (Soundness)

$$\boxed{\emptyset} \vdash_{\mathtt{inv}} s_1 \,{}^{\forall}\, |\; s_2 \,{}^{\exists}\, \{\, \varphi \,\} \implies (s_1, s_2) \models \forall\exists \Box \varphi$$

Remark. The proof is particularly difficult to mechanize in Rocq, as it requires extracting a witness from a quadruple (in `Prop`).

→ The original proof uses the axiom of functional choice.

→ Functional choice can be shown to be a **necessary condition**.

Beyond Invariants: Handling Safety via Derivatives

The technique can be easily generalized to arbitrary safety relations via the notion of **derivatives**.

Beyond Invariants: Handling Safety via Derivatives

The technique can be easily generalized to arbitrary safety relations via the notion of **derivatives**.

Let $\psi \subseteq \mathcal{E}^\omega \times \mathcal{E}^\omega$ be a relation between traces. The derivative of ψ with respect to $e_1, e_2 \in \mathcal{E}$ is

$$\Delta_{e_1,e_2}(\psi) \triangleq \{\ (\pi_1, \pi_2) \mid \psi(e_1 \cdot \pi_1, e_2 \cdot \pi_2)\ \}$$

Intuition: "What remains to be satisfied if the first events are (e_1, e_2)"

Generalized Quadruples

We generalize our quadruples to support safety relations:

$$\boxed{H} \vdash s_1 \,{}^{\forall}\!\mid\, s_2 \,{}^{\exists}\, \{\, \psi \,\}$$

$$\overline{\boxed{H}} \vdash s_1 \,{}^{\forall}\!\mid\, s_2 \,{}^{\exists}\, \{\, \psi \,\}$$

$$\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall}\!\mid\, e_2 \triangleright s_2 \,{}^{\exists}\, \{\, \psi \,\}$$

$$\begin{cases} H \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{P}(\mathcal{E}^\omega \times \mathcal{E}^\omega) \\ s_1, s_2 \in \mathcal{S} \\ \psi \subseteq \mathcal{E}^\omega \times \mathcal{E}^\omega \end{cases}$$

Intuition

Intuition behind $\boxed{H} \vdash e_1 \triangleright s_1 \,^{\forall} \mid \; e_2 \triangleright s_2 \,^{\exists} \{ \psi \}$: Holds under the following two conditions:

1. ψ is not immediately violated by e_1, e_2:

$$\Delta_{e_1,e_2}(\psi) \neq \emptyset$$

2. ψ is not violated later:

$$\boxed{\boxed{H}} \vdash s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \Delta_{e_1,e_2}(\psi) \}$$

A Deductive System for Safety Relations

$$\textsc{Step}\quad \frac{\forall s_1 \xrightarrow{e_1} s_1'.\exists s_2 \xrightarrow{e_2} s_2'.\ \boxed{H} \vdash e_1 \triangleright s_1'\ ^{\forall} \mid\ e_2 \triangleright s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

A Deductive System for Safety Relations

Step

$$\frac{\forall s_1 \xrightarrow{e_1} s_1'.\exists s_2 \xrightarrow{e_2} s_2'.\ \boxed{H} \vdash e_1 \triangleright s_1'\ ^{\forall} \mid\ e_2 \triangleright s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Invariant

$$\frac{(s_1, s_2, \psi) \in H' \qquad \forall (s_1', s_2', \psi) \in H', \boxed{H \cup H'} \vdash s_1'\ ^{\forall} \mid\ s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Cycle

$$\frac{(s_1, s_2, \psi) \in H}{\boxed{\boxed{H}} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

A Deductive System for Safety Relations

Step

$$\frac{\forall s_1 \xrightarrow{e_1} s_1'.\exists s_2 \xrightarrow{e_2} s_2'.\ \boxed{H} \vdash e_1 \triangleright s_1'\ ^{\forall} \mid\ e_2 \triangleright s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Invariant

$$\frac{(s_1, s_2, \psi) \in H' \qquad \forall (s_1', s_2', \psi) \in H', \boxed{H \cup H'} \vdash s_1'\ ^{\forall} \mid\ s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Cycle

$$\frac{(s_1, s_2, \psi) \in H}{\lceil \bar{H} \rceil \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Derive

$$\frac{\Delta_{e_1,e_2}(\psi) \neq \emptyset \qquad \lceil \bar{H} \rceil \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \Delta_{e_1,e_2}(\psi)\ \}}{\boxed{H} \vdash e_1 \triangleright s_1\ ^{\forall} \mid\ e_2 \triangleright s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Soundness

Theorem (Soundness)

If ψ is a safety relation

$$\boxed{\emptyset} \vdash s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{ \psi \} \implies (s_1, s_2) \models \forall\exists\psi$$

Soundness

Theorem (Soundness)

If ψ is a safety relation

$$\boxed{\emptyset} \vdash s_1 \,^{\forall} \mid \; s_2 \,^{\exists} \{ \, \psi \, \} \implies (s_1, s_2) \models \forall\exists\psi$$

Proof (intuition). Safety relations are exactly relations ψ that are satisfied by a pair (π_1, π_2) iff the successive derivatives are non-empty

$$\psi_0 = \Delta_{\pi_1(0),\pi_2(0)}(\psi) \neq \emptyset \qquad \psi_1 = \Delta_{\pi_1(1),\pi_2(1)}(\psi_0) \neq \emptyset \qquad \dots$$

Calculating Derivatives

Derivatives can be **calculated** for relations expressed with LTL modalities.

$$\Delta_{e_1,e_2}(\Box\psi) \equiv \Delta_{e_1,e_2}(\psi) \wedge \Box\psi$$

$$\Delta_{e_1,e_2}(\psi_1 \mathsf{W}\, \psi_2) \equiv \Delta_{e_1,e_2}(\psi_2) \vee [\Delta_{e_1,e_2}(\psi_1) \wedge (\psi_1 \mathsf{W}\, \psi_2)]$$

$$\Delta_{e_1,e_2}(\bigcirc\psi) \equiv \psi$$

$$\Delta_{e_1,e_2}(\varphi) \equiv \begin{cases} \mathit{true} & \text{if } \varphi(e_1, e_1) \\ \mathit{false} & \text{otherwise} \end{cases}$$

Derived Rules for Temporal Modalities

This allows us to derive more ergonomic reasoning rules.

DERIVE-W-NOW

$$\frac{\varphi_2(e_1, e_2) \qquad \lceil H \rceil \vdash s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{\; true \;\}}{\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall} \mid \; e_2 \triangleright s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}$$

DERIVE-W-LATER

$$\frac{\varphi_1(e_1, e_2) \qquad \lceil H \rceil \vdash s_1 \,{}^{\forall} \mid \; s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}{\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall} \mid \; e_2 \triangleright s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}$$

Derived Rules for Temporal Modalities

This allows us to derive more ergonomic reasoning rules.

DERIVE-W-NOW

$$\frac{\varphi_2(e_1, e_2) \qquad \boxed{\boxed{H}} \vdash s_1 \,{}^{\forall} \mid\; s_2 \,{}^{\exists} \{\; true \;\}}{\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall} \mid\; e_2 \triangleright s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}$$

DERIVE-W-LATER

$$\frac{\varphi_1(e_1, e_2) \qquad \boxed{\boxed{H}} \vdash s_1 \,{}^{\forall} \mid\; s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}{\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall} \mid\; e_2 \triangleright s_2 \,{}^{\exists} \{\; \varphi_1 \;\mathsf{W}\; \varphi_2 \;\}}$$

DERIVE-◯

$$\frac{\psi \neq false \qquad \boxed{H} \vdash s_1 \,{}^{\forall} \mid\; s_2 \,{}^{\exists} \{\; \psi \;\}}{\boxed{H} \vdash e_1 \triangleright s_1 \,{}^{\forall} \mid\; e_2 \triangleright s_2 \,{}^{\exists} \{\; \bigcirc \psi \;\}}$$

Derived Rules for Temporal Modalities

This allows us to derive more ergonomic reasoning rules.

DERIVE-W-NOW

$$\frac{\varphi_2(e_1, e_2) \qquad \boxed{\boxed{H}} \vdash s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \; true \; \}}{\boxed{H} \vdash e_1 \triangleright s_1 \;^{\forall} \mid \; e_2 \triangleright s_2 \;^{\exists} \{ \; \varphi_1 \; \mathsf{W} \; \varphi_2 \; \}}$$

DERIVE-W-LATER

$$\frac{\varphi_1(e_1, e_2) \qquad \boxed{\boxed{H}} \vdash s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \; \varphi_1 \; \mathsf{W} \; \varphi_2 \; \}}{\boxed{H} \vdash e_1 \triangleright s_1 \;^{\forall} \mid \; e_2 \triangleright s_2 \;^{\exists} \{ \; \varphi_1 \; \mathsf{W} \; \varphi_2 \; \}}$$

DERIVE-◯

$$\frac{\psi \neq false \qquad \boxed{H} \vdash s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \; \psi \; \}}{\boxed{H} \vdash e_1 \triangleright s_1 \;^{\forall} \mid \; e_2 \triangleright s_2 \;^{\exists} \{ \; \bigcirc \psi \; \}}$$

. . . and more

Substituting Equivalent Relations

Remark 1. Deriving these rules requires substituting relations with equivalent ones in quadruples

$$\textsc{Derive}\ \frac{\dots \qquad \dfrac{\dots \qquad \Delta_{e_1,e_2}\psi \equiv \psi' \qquad \boxed{H} \vdash s_1\ {}^{\forall} \mid\ s_2\ {}^{\exists}\ \{\ \psi'\ \}}{\boxed{H} \vdash s_1\ {}^{\forall} \mid\ s_2\ {}^{\exists}\ \{\ \Delta_{e_1,e_2}(\psi)\ \}}}{\boxed{H} \vdash e_1 \triangleright s_1\ {}^{\forall} \mid\ e_2 \triangleright s_2\ {}^{\exists}\ \{\ \psi\ \}}$$

Substituting Equivalent Relations

Remark 1. Deriving these rules requires substituting relations with equivalent ones in quadruples

$$\textsc{Derive}\ \frac{\dots \qquad \dfrac{\dots \qquad \Delta_{e_1,e_2}\psi \equiv \psi' \qquad \boxed{H} \vdash s_1\ {}^{\forall}|\ s_2\ {}^{\exists}\{\ \psi'\ \}}{\boxed{H} \vdash s_1\ {}^{\forall}|\ s_2\ {}^{\exists}\{\ \Delta_{e_1,e_2}(\psi)\ \}}}{\boxed{H} \vdash e_1 \triangleright s_1\ {}^{\forall}|\ e_2 \triangleright s_2\ {}^{\exists}\{\ \psi\ \}}$$

This is not immediate in Rocq without the axiom of predicate extensionality

$$\psi_1 \equiv \psi_2 \implies \psi_1 = \psi_2$$

Exploiting Stronger Relations

Remark 2. When computing successive derivatives, we often find ourselves with a pair of states that is already in the context, but associated with a *stronger* relation.

$$\dfrac{\dfrac{(s_1, s_2, \mathit{true}) \in \{\ (s_1, s_2, \psi)\ \}}{(s_1, s_2, \psi) \vdash s_1\ {}^{\forall}\ |\ s_2\ {}^{\exists}\ \{\ \mathit{true}\ \}}\ \textsc{Cycle}}{\vdots}$$

$$\dfrac{\boxed{(s_1, s_2, \psi)} \vdash s_1\ {}^{\forall}\ |\ s_2\ {}^{\exists}\ \{\ \psi\ \}}{\boxed{\emptyset} \vdash s_1\ {}^{\forall}\ |\ s_2\ {}^{\exists}\ \{\ \psi\ \}}\ \textsc{Invariant}$$

Reasoning Up-to Stronger Trace Relations

To solve the two problems without axioms, we define our quadruples "*up-to stronger trace relations*".

Reasoning Up-to Stronger Trace Relations

To solve the two problems without axioms, we define our quadruples "*up-to stronger trace relations*".

The soundness of this generalization is justified by framing it as a "compatible up-to technique" [Pou16; Zak+20].

Reasoning Up-to Stronger Trace Relations

To solve the two problems without axioms, we define our quadruples "*up-to stronger trace relations*".

The soundness of this generalization is justified by framing it as a "compatible up-to technique" [Pou16; Zak+20].

This generalization gives access to an additional *strengthening* rule

$$\textsc{Strengthen}\quad \frac{\psi' \subseteq \psi \qquad \boxed{H} \vdash s_1 \;{}^{\forall} \mid\; s_2 \;{}^{\exists} \{\, \psi' \,\}}{\boxed{H} \vdash s_1 \;{}^{\forall} \mid\; s_2 \;{}^{\exists} \{\, \psi \,\}}$$

Other Contributions: Programs

Additional rules can be derived when LTSs are generated by **programs** (we can construct program logics on top of HyCo).

$$\cdots \quad \frac{\forall v.\ \boxed{H} \vdash \langle P_1, m_1[x \mapsto v] \rangle^{\forall} \mid \langle P_2, m_2 \rangle^{\exists} \{ \psi \}}{\boxed{H} \vdash \langle \textbf{havoc}\ x\ \textbf{;}\ P_1, m_1 \rangle^{\forall} \mid \langle P_2, m_2 \rangle^{\exists} \{ \psi \}}\ \textsc{Havoc-L}$$

$$\frac{\exists v.\ \boxed{H} \vdash \langle P_1, m_1 \rangle^{\forall} \mid \langle P_2, m_2[x \mapsto v] \rangle^{\exists} \{ \psi \}}{\boxed{H} \vdash \langle P_1, m_1 \rangle^{\forall} \mid \langle \textbf{havoc}\ x\ \textbf{;}\ P_2, m_2 \rangle^{\exists} \{ \psi \}}\ \textsc{Havoc-R} \quad \cdots$$

Selection of rules to handle **havoc** statements

Other Contributions: Tau-Steps

HyCo supports tau-steps (i.e., silent transitions) and non-lock-step *alignments*.

STEPS-L

$$\frac{s_1 \to^* s_1' \text{ determinstically} \qquad \boxed{H} \vdash s_1'\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

STEPS-R

$$\frac{s_2 \to^* s_2' \qquad \boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2'\ ^{\exists}\ \{\ \psi\ \}}{\boxed{H} \vdash s_1\ ^{\forall} \mid\ s_2\ ^{\exists}\ \{\ \psi\ \}}$$

Other Contributions: Reasoning Up-to Simulation

HyCo supports reasoning **up-to simulation**, enabling a form of over/under-approximation of the universally/existentially quantified behaviors.

SIM-L

$$\frac{s_1 \preceq s_1' \qquad \boxed{H} \vdash s_1'^{\,\forall} \mid s_2^{\,\exists} \{ \psi \}}{\boxed{H} \vdash s_1^{\,\forall} \mid s_2^{\,\exists} \{ \psi \}}$$

SIM-R

$$\frac{s_2' \preceq s_2 \qquad \boxed{H} \vdash s_1^{\,\forall} \mid s_2'^{\,\exists} \{ \psi \}}{\boxed{H} \vdash s_1^{\,\forall} \mid s_2^{\,\exists} \{ \psi \}}$$

Other Contributions: Reasoning Up-to Simulation

In fact, SIM-L and SIM-R can be *internalized*.

SIM-L

$$\frac{\boxed{\emptyset} \vdash s_1 \;^{\forall} \mid \; s_1' \;^{\exists} \{ \Box\, eq \} \qquad \boxed{H} \vdash s_1' \;^{\forall} \mid \; s_2 \;^{\exists} \{ \psi \}}{\boxed{H} \vdash s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \psi \}}$$

SIM-R

$$\frac{\boxed{\emptyset} \vdash s_2' \;^{\forall} \mid \; s_2 \;^{\exists} \{ \Box\, eq \} \qquad \boxed{H} \vdash s_1 \;^{\forall} \mid \; s_2' \;^{\exists} \{ \psi \}}{\boxed{H} \vdash s_1 \;^{\forall} \mid \; s_2 \;^{\exists} \{ \psi \}}$$

Future Work

Future Work

? Currently limited to *safety* relations. What about *liveness*?

$\Longrightarrow$ e.g., using *fair* notions of simulation [CKF25].

Future Work

? Currently limited to *safety* relations. What about *liveness*?

$\Longrightarrow$ e.g., using *fair* notions of simulation [CKF25].

? HyCo is provably *incomplete*. How to make it complete?

$\Longrightarrow$ e.g., using *prophecy variables* [BF22].

Future Work

? Currently limited to *safety* relations. What about *liveness*?

$\Longrightarrow$ e.g., using *fair* notions of simulation [CKF25].

? HyCo is provably *incomplete*. How to make it complete?

$\Longrightarrow$ e.g., using *prophecy variables* [BF22].

? How about reasoning about realistic programs manipulating data-structures?

$\Longrightarrow$ e.g., using separation logic à la Simuliris [Gäh+22].

Bibliography i

[BF22] R. Beutner and B. Finkbeiner. **Prophecy Variables for Hyperproperty Verification.** In: IEEE Computer Security Foundations Symposium, CSF 2022. IEEE, 2022.

[CF25] A. Correnson and B. Finkbeiner. **Coinductive Proofs for Temporal Hyperliveness.** In: Proc. ACM Program. Lang. 9.POPL (Jan. 2025). DOI: 10.1145/3704889. URL: https://doi.org/10.1145/3704889.

[CKF25] A. Correnson, I. Kuhn, and B. Finkbeiner. **Almost Fair Simulations.** In: Proc. ACM Program. Lang. 9.ICFP (Aug. 2025). DOI: 10.1145/3747512. URL: https://doi.org/10.1145/3747512.

Bibliography ii

[Gäh+22] L. Gäher, M. Sammler, S. Spies, R. Jung, H.-H. Dang, R. Krebbers, J. Kang, and D. Dreyer. **Simuliris: a separation logic framework for verifying concurrent program optimizations.** In: Proc. ACM Program. Lang. 6.POPL (Jan. 2022). DOI: 10.1145/3498689. URL: https://doi.org/10.1145/3498689.

Bibliography iii

[Hur+13] C.-K. Hur, G. Neis, D. Dreyer, and V. Vafeiadis. **The power of parameterization in coinductive proof.** In: Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '13. Rome, Italy: Association for Computing Machinery, 2013, pp. 193–206. ISBN: 9781450318327. DOI: 10.1145/2429069.2429093. URL: https://doi.org/10.1145/2429069.2429093.

Bibliography iv

[Pou16] D. Pous. **Coinduction All the Way Up.** In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 307–316. ISBN: 9781450343916. DOI: 10.1145/2933575.2934564. URL: https://doi.org/10.1145/2933575.2934564.

Bibliography v

[Zak+20] Y. Zakowski, P. He, C.-K. Hur, and S. Zdancewic. **An equational theory for weak bisimulation via generalized parameterized coinduction.** In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. CPP 2020. New Orleans, LA, USA: Association for Computing Machinery, 2020, pp. 71–84. ISBN: 9781450370974. DOI: 10.1145/3372885.3373813. URL: https://doi.org/10.1145/3372885.3373813.