Completeness of Iris-Based Program Logics

Johannes Hostert[1], Zichen Zhang[2], Puming Liu[3],
Simon Oddershede Gregersen[4], Ralf Jung[1], Joseph Tassarotti[2]

ICFP 2026, Indianapolis, USA
27 August 2026

1: ETH Zurich, Switzerland 2: New York University, USA 3: NYU Shanghai, China 4: CISPA Helmholtz Center for Information Security, Germany

What is Completeness?

let $x :=$ alloc 40 in
$x \leftarrow !x + 2\,;;$
$!x$

What is Completeness?

$\{\mathsf{True}\}$

$\quad \mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}$

$\quad\quad x \leftarrow\ !x + 2\ ;;$

$\quad\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

What is Completeness?

$\{\mathsf{True}\}$

$\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}$

$\quad x \leftarrow\ !x + 2\ ;;$

$\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

Syntactic

What is Completeness?

$\{\mathsf{True}\}$

$$\left.\begin{array}{l}\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}\\ \quad x \leftarrow\ !x + 2\ ;;\\ \quad !x\end{array}\right\} e$$

$$\{v.\ \exists \ell.\ \ulcorner \underbrace{v = 42}_{\varphi(v)} \urcorner * \ell \mapsto 42\}$$

Syntactic

What we actually want is:

$$\mathit{safe}_{\varphi}(e)$$

What is Completeness?

$\{\mathsf{True}\}$

$$\left.\begin{array}{l}\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}\\ \quad x \leftarrow\ !x + 2\ ;;\\ \quad !x\end{array}\right\} e$$

$$\{v.\ \exists \ell.\ \ulcorner \underbrace{v = 42}_{\varphi(v)} \urcorner * \ell \mapsto 42\}$$

Syntactic

What we actually want is:

$$\mathit{safe}_{\varphi}(e)$$

- e never gets stuck
- If e terminates, φ holds
- e can also diverge

What is Completeness?

$\{\mathsf{True}\}$

$$\left.\begin{array}{l}\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}\\ \quad x \leftarrow\ !x + 2\ ;;\\ \quad !x\end{array}\right\} e$$

$$\{v.\ \exists \ell.\ \ulcorner \underbrace{v = 42}_{\varphi(v)} \urcorner * \ell \mapsto 42\}$$

Syntactic

What we actually want is:

$\mathit{safe}_{\varphi}(e)$

- e never gets stuck
- If e terminates, φ holds
- e can also diverge

Semantic

What is Completeness?

$\{\mathsf{True}\}$

$$\left.\begin{array}{l}\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}\\ \quad x \leftarrow\ !x + 2\ ;;\\ \quad !x\end{array}\right\} e$$

$$\{v.\ \exists \ell.\ \ulcorner \underbrace{v = 42}_{\varphi(v)} \urcorner * \ell \mapsto 42\}$$

What we actually want is:

$$\mathit{safe}_{\varphi}(e)$$

- e never gets stuck
- If e terminates, φ holds
- e can also diverge

Syntactic —Soundness→ Semantic

What is Completeness?

$\{\mathsf{True}\}$

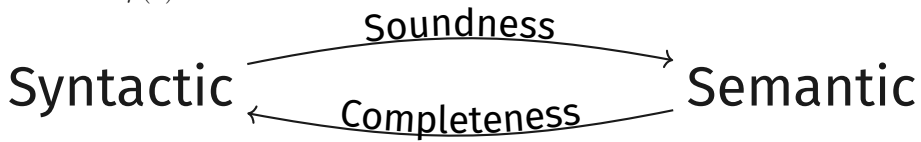
$$\left.\begin{array}{l}\mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}\\ \quad x \leftarrow\ !x + 2\ ;;\\ \quad !x\end{array}\right\} e$$

$$\{v.\ \exists \ell.\ \ulcorner \underbrace{v = 42}_{\varphi(v)} \urcorner * \ell \mapsto 42\}$$

What we actually want is:

$$\mathit{safe}_{\varphi}(e)$$

- e never gets stuck
- If e terminates, φ holds
- e can also diverge



What is Completeness?

$\{\text{True}\}$

let $x := \text{alloc } 40$ in
$x \leftarrow !x + 2$;;
$!x$

$\}e$

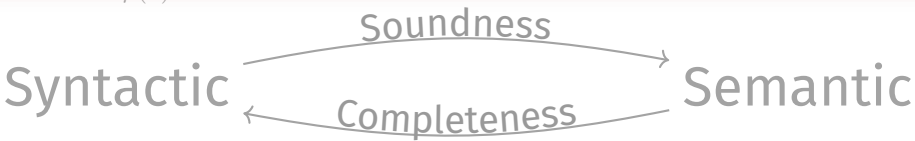
$\varphi(v)$

What we actually want is:

$\mathit{safe}_{\varphi}(e)$

- e never gets stuck

Completeness: All Correct Programs Can Be Verified



A Note On Gödel

Any
sufficiently expressive system
is incomplete!

A Note On Gödel

Any
sufficiently expressive system
is incomplete!

Side-step issue with **pure embedding** $\ulcorner\psi\urcorner$:

A Note On Gödel

Any
sufficiently expressive system
is incomplete!

Side-step issue with **pure embedding** $\ulcorner\psi\urcorner$:

$$\frac{\psi \text{ holds in meta-logic}}{P \vdash \ulcorner\psi\urcorner}$$

A Note On Gödel

Any
sufficiently expressive system
is incomplete!

Side-step issue with **pure embedding** $\ulcorner\psi\urcorner$:

$$\frac{\psi \text{ holds in meta-logic}}{P \vdash \ulcorner\psi\urcorner}$$

Proof system is not entirely syntactic!

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Proof will be easy!

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Proof will be easy!

In λ_{Rust}, which tries to model Rust:

$\{\ell \mapsto_q v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto_q v\}$ is complete for:

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Proof will be easy!

In λ_{Rust}, which tries to model Rust:

$\{\ell \mapsto_q v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto_q v\}$ is complete for:

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Proof will be easy!

In λ_{Rust}, which tries to model Rust:

$\{\ell \mapsto_q v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto_q v\}$ is complete for:

Data Race Detection:

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$$\frac{\sigma(\ell) = v}{\langle !\ell, \sigma\rangle \to_b \langle v, \sigma\rangle}$$

Simple semantics, close to Hoare rule

Proof will be easy!

In λ_{Rust}, which tries to model Rust:

$\{\ell \mapsto_q v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto_q v\}$ is complete for:

Data Race Detection:

Administrative Redexes, Reader-Writer Locks, …

Motivation: Resource Rules And Perfect Abstractions

In our simple example language:

$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$ is complete for:

$\sigma(\ell) = v$ Simple semantics, close to Hoare rule

Program Logic is **Perfect Abstraction** over the Semantics!

In λ_{Rust}, which tries to model Rust:

$\{\ell \mapsto_q v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto_q v\}$ is complete for:

Data Race Detection:

Administrative Redexes, Reader-Writer Locks, …

The Setup

Iris Logic

Example Language

The Setup

Iris Logic

Example Language

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$
- separation logic: $*, -\!\!*, \ldots$

Example Language

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$
- separation logic: $*, -\!\!*, \ldots$
- program logic: $\{P\}\ e\ \{Q\}$

Example Language

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$
- separation logic: $*, -\!\!*, \ldots$
- program logic: $\{P\}\ e\ \{Q\}$

Example Language

- ML-like lambda calculus

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$
- separation logic: $*, -\!\!*, \ldots$
- program logic: $\{P\}\ e\ \{Q\}$

Example Language

- ML-like lambda calculus
- higher-order state, recursion

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner \psi \urcorner$
- separation logic: $*, -\!\!*, \ldots$
- program logic: $\{P\}\ e\ \{Q\}$

Example Language

- ML-like lambda calculus
- higher-order state, recursion
- contextual OpSem

The Setup

Iris Logic

- syntactic-ish system: $\vdash, \ulcorner\psi\urcorner$
- separation logic: $*, -\!\!*, \ldots$
- program logic: $\{P\}\ e\ \{Q\}$

Example Language

- ML-like lambda calculus
- higher-order state, recursion
- contextual OpSem

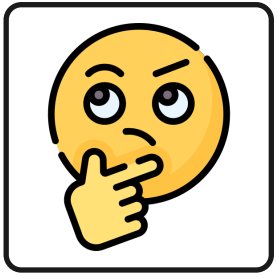
Goal will be: $\{\ulcorner \mathit{safe}_{\varphi}(e)\urcorner\}\ e\ \{v.\ \ulcorner\varphi(v)\urcorner\}$

Outline

Review the Laws

Do the Proof

Workshop the Proof



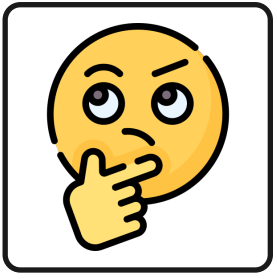
Reflect on the Proof

Outline

Review the Laws

Do the Proof

Workshop the Proof



Reflect on the Proof

The Proof System Laws, by Example

$\{\mathsf{True}\}$

$\quad \mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}$

$\quad\quad x \leftarrow\ !x + 2\ ;;$

$\quad\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

The Proof System Laws, by Example

$\{\mathsf{True}\}$

$\quad \mathsf{let}\ x := \mathsf{alloc}\ 40\ \mathsf{in}$

$\quad\quad x \leftarrow\ !x + 2\ ;;$

$\quad\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\ \exists \ell.\ \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

The Proof System Laws, by Example

$\{\mathsf{True}\}$

$\quad \mathsf{let}\ x := \boxed{\mathsf{alloc}\ 40}\ \mathsf{in}$

$\quad\quad x \leftarrow\ !x + 2\ ;;$

$\quad\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\ \exists \ell.\ \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e\ \{v.\ \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

let $x :=$ ℓ in

$x \leftarrow !x + 2\,;;$

$!x$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$\mathsf{let}\ x := \quad \ell \quad \mathsf{in}$

$\quad x \leftarrow !x + 2\ ;;$

$\quad !x$

$\{v.\ \exists \ell.\ \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\ \exists \ell.\ \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

$$\frac{\{P\}\ e\ \{v.\ \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$$\ell \leftarrow !\ell + 2 \,;;$$
$$!\ell$$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$$\begin{array}{l} \ell \leftarrow !\ell + 2\,;; \\ !\ell \end{array}$$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$\ell \leftarrow 40 + 2\,;;$
$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$\ell \leftarrow 40 + 2\,;;$
$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$\ell \leftarrow 42 \quad ;;$
$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 40\}$

$\ell \leftarrow 42 \quad ;;$
$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w.\, \ulcorner w = () \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 42\}$

$()$;;
$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w.\, \ulcorner w = () \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$\{\ell \mapsto 42\}$

$!\ell$

$\{v.\, \exists \ell.\, \ulcorner v = 42 \urcorner * \ell \mapsto 42\}$

$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$

$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w.\, \ulcorner w = () \urcorner * \ell \mapsto v\}$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$$\ell \mapsto 42 \vdash \exists \ell.\ulcorner 42 = 42 \urcorner * \ell \mapsto 42$$

$$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\exists \ell.\ulcorner w = \ell \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto v\}\ !\ell\ \{w.\ulcorner w = v \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w.\ulcorner w = () \urcorner * \ell \mapsto v\}$$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let} \qquad \textsc{Add}, \textsc{Seq}, \ldots$$

$$\frac{\{P\}\ e\ \{v.\{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$$\ell \mapsto 42$$

$$\vdash \exists \ell.\, \ulcorner 42 = 42 \urcorner * \ell \mapsto 42$$

using ~22 assertion logic rules (not shown here)

$$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w.\, \exists \ell.\, \ulcorner w = \ell \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto v\}\ !\ell\ \{w.\, \ulcorner w = v \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w.\, \ulcorner w = () \urcorner * \ell \mapsto v\}$$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof System Laws, by Example

$$\ell \mapsto 42$$

$$\vdash \exists \ell. \ulcorner 42 = 42 \urcorner * \ell \mapsto 42$$

using ~22 assertion logic rules (not shown here)

language-specific rules

$$\{\mathsf{True}\}\ \mathsf{alloc}\ v\ \{w. \exists \ell. \ulcorner w = \ell \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto v\}\ !\ell\ \{w. \ulcorner w = v \urcorner * \ell \mapsto v\}$$

$$\{\ell \mapsto \_\}\ \ell \leftarrow v\ \{w. \ulcorner w = () \urcorner * \ell \mapsto v\}$$

$$\frac{\{P\}\ e[x/v]\ \{Q\}}{\{P\}\ \mathsf{let}\ x := v\ \mathsf{in}\ e\ \{Q\}}\ \textsc{Let}$$

Add, Seq, ...

structural rules

Value, Weaken, Frame

$$\frac{\{P\}\ e\ \{v.\ \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

Brace Yourselves – We Do the Proof!

The Proof, Part 1

$$\{\ulcorner \mathit{safe}_{\varphi}(e)\urcorner\}$$

$$e$$

$$\{v.\ulcorner\varphi(v)\urcorner\}$$

The Proof, Part 1

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(e, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$e$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- Generalize over state σ

The Proof, Part 1

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(e, \sigma)\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\, \ell \mapsto v\end{array}\right\}$$

$$e$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ

$\vee$ $\langle e, \sigma\rangle$ reducible

$\wedge\ \forall e', \sigma'.\ \langle e, \sigma\rangle \to \langle e', \sigma'\rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 1

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(e, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$e$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ
$\vee$ $\langle e, \sigma \rangle$ reducible
$\wedge \forall e', \sigma'. \langle e, \sigma \rangle \to \langle e', \sigma' \rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 1

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(w, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

w

$\{v. \ulcorner \varphi(v) \urcorner\}$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible
- If e is some value w:

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ
$\vee$ $\langle e, \sigma \rangle$ reducible
$\wedge \forall e', \sigma'. \langle e, \sigma \rangle \to \langle e', \sigma' \rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 1

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(w, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$w$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- Generalize over state σ
- Analyze $\mathit{safe}_\varphi(e, \sigma)$:
  e is value or reducible
- If e is some value w:
  From $\mathit{safe}_\varphi(w, \sigma)$: $\varphi(w)$ holds

$\mathit{safe}_\varphi(e, \sigma) \cong e$ is value satisfying φ

$\quad \vee \langle e, \sigma \rangle$ reducible

$\quad\quad \wedge \forall e', \sigma'. \langle e, \sigma \rangle \to \langle e', \sigma' \rangle \Rightarrow \mathit{safe}_\varphi(e', \sigma')$

The Proof, Part 1

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(w, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$w$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible
- If e is some value w:
  From $\mathit{safe}_{\varphi}(w, \sigma)$: $\varphi(w)$ holds

The Proof, Part 1

$$\ulcorner \mathit{safe}_{\varphi}(w, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \\ \vdash \\ \ulcorner \varphi(w) \urcorner$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible
- If e is some value w:
  From $\mathit{safe}_{\varphi}(w, \sigma)$: $\varphi(w)$ holds
  Apply VALUE, WEAKEN rules

The Proof, Part 1

$$\ulcorner \mathit{safe}_{\varphi}(w, \sigma) \urcorner * * _{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \vdash \ulcorner \varphi(w) \urcorner$$

$$\frac{\psi \text{ holds in meta-logic}}{P \vdash \ulcorner \psi \urcorner}$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible
- If e is some value w:
  From $\mathit{safe}_{\varphi}(w, \sigma)$: $\varphi(w)$ holds
  Apply VALUE, WEAKEN rules
  Conclude, as $\varphi(w)$ holds!

The Proof, Part 1

$$\left\{\begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(e, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array}\right\}$$

$$e$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- Generalize over state σ
- Analyze $\mathit{safe}_{\varphi}(e, \sigma)$:
  e is value or reducible
- If e is some value w:
  From $\mathit{safe}_{\varphi}(w, \sigma)$: $\varphi(w)$ holds
  Apply VALUE, WEAKEN rules
  Conclude, as $\varphi(w)$ holds!
- If $\langle e, \sigma \rangle$ is reducible…

The Proof, Part 2

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(e, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$e$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- If $\langle e, \sigma \rangle$ is reducible

The Proof, Part 2

$$\left\{\begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(e, \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array}\right\}$$

$$e$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- If $\langle e, \sigma \rangle$ is reducible

$$\frac{}{\langle e, \sigma \rangle \to \langle e', \sigma' \rangle}$$

The Proof, Part 2

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_{\varphi}(K[e], \sigma) \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$K[e]$$

$$\{v. \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \rightarrow_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \rightarrow \langle K[e'], \sigma' \rangle}$$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e], \sigma)\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$
$$K[e]$$
$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ
$\vee$ $\langle e, \sigma\rangle$ reducible
$\wedge\ \forall e', \sigma'.\ \langle e, \sigma\rangle \to \langle e', \sigma'\rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e], \sigma)\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$
$$K[e]$$
$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ
$\vee\ \langle e, \sigma\rangle$ reducible
$\wedge\ \forall e', \sigma'.\ \langle e, \sigma\rangle \to \langle e', \sigma'\rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e'], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[e]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

$\mathit{safe}_{\varphi}(e, \sigma) \cong e$ is value satisfying φ

$\vee\ \langle e, \sigma\rangle$ reducible

$\wedge\ \forall e', \sigma'.\ \langle e, \sigma\rangle \to \langle e', \sigma'\rangle \Rightarrow \mathit{safe}_{\varphi}(e', \sigma')$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e'], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\, \ell \mapsto v\end{array}\right\}$$
$$K[e]$$
$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e'], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\, \ell \mapsto v\end{array}\right\}$$

$$K[e]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[v], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[!\ell]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[v], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[!\ell]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

The Proof, Part 2

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(K[v], \sigma') \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$!\ell$$

$$\{w.\ \{\mathsf{True}\}\ K[w]\ \{v.\ \ulcorner \varphi(v) \urcorner\}\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \to_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \to \langle K[e'], \sigma' \rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

  Apply BIND rule

The Proof, Part 2

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(K[v], \sigma') \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$!\ell$

$$\{w.\, \{\mathsf{True}\}\ K[w]\ \{v.\, \ulcorner \varphi(v) \urcorner\}\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \to_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \to \langle K[e'], \sigma' \rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

  Apply BIND rule

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[v], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[v]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

  Apply BIND rule

  Apply LOAD rule

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[v], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[v]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

  Apply BIND rule

  Apply LOAD rule

  Observe: we are back where we started!

The Proof, Part 2

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[v], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[v]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = !\ell$: $\sigma(\ell) = v,\ e' = v, \sigma' = \sigma$

  Apply BIND rule — Observe: we are back where we started!

  Apply LOAD rule — Conclude by induction on steps.

  (assuming termination, for now)

The Proof, Part 3

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e'], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\, \ell \mapsto v\end{array}\right\}$$

$$K[e]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

The Proof, Part 3

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e_1[v/x]], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[\text{let } x := v \text{ in } e_1]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \text{let } x := v \text{ in } e_1$: $e' = e_1[v/x], \sigma' = \sigma$

The Proof, Part 3

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(K[e_1[v/x]], \sigma') \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

let $x := v$ in e_1

$$\{w.\ \{\mathsf{True}\}\ K[w]\ \{v.\ \ulcorner \varphi(v) \urcorner\}\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \to_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \to \langle K[e'], \sigma' \rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e =$ let $x := v$ in e_1: $e' = e_1[v/x], \sigma' = \sigma$

  Apply BIND rule

The Proof, Part 3

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(K[e_1[v/x]], \sigma') \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$e_1[v/x]$

$$\{w.\, \{\mathsf{True}\}\ K[w]\ \{v. \ulcorner \varphi(v) \urcorner\}\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \to_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \to \langle K[e'], \sigma' \rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \mathsf{let}\ x := v\ \mathsf{in}\ e_1$: $e' = e_1[v/x], \sigma' = \sigma$

  Apply Bind rule

  Apply Let rule

The Proof, Part 3

$$\left\{ \begin{array}{l} \ulcorner \mathit{safe}_\varphi(K[e_1[v/x]], \sigma') \urcorner * \\ *_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \end{array} \right\}$$

$$e_1[v/x]$$

$$\{w.\, \{\mathsf{True}\}\ K[w]\ \{v. \ulcorner \varphi(v) \urcorner\}\}$$

- If $\langle K[e], \sigma \rangle$ is reducible

$$\frac{\langle e, \sigma \rangle \to_b \langle e', \sigma' \rangle}{\langle K[e], \sigma \rangle \to \langle K[e'], \sigma' \rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \mathsf{let}\ x := v\ \mathsf{in}\ e_1$: $e' = e_1[v/x], \sigma' = \sigma$

  Apply Bind rule

  Apply Let rule

$$\frac{\{P\}\ e\ \{v.\, \{\mathsf{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof, Part 3

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_\varphi(K[e_1[v/x]], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma}\ \ell \mapsto v\end{array}\right\}$$

$$K[e_1[v/x]]$$

$$\{v.\ \ulcorner \varphi(v) \urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_\varphi(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \text{let } x := v \text{ in } e_1$: $e' = e_1[v/x], \sigma' = \sigma$

  Apply Bind rule

  Apply Let rule

  Apply Bind rule in reverse

$$\frac{\{P\}\ e\ \{v.\ \{\text{True}\}\ K[v]\ \{Q\}\}}{\{P\}\ K[e]\ \{Q\}}\ \textsc{Bind}$$

The Proof, Part 3

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e_1[v/x]], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma'} \ell \mapsto v\end{array}\right\}$$

$$K[e_1[v/x]]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \text{let } x := v \text{ in } e_1$: $e' = e_1[v/x], \sigma' = \sigma$

  Apply Bind rule

  Apply Let rule

  Apply Bind rule in reverse

The Proof, Part 3

$$\left\{\begin{array}{l}\ulcorner \mathit{safe}_{\varphi}(K[e_1[v/x]], \sigma')\urcorner * \\ *_{(\ell \leftarrow v) \in \sigma'} \ell \mapsto v\end{array}\right\}$$

$$K[e_1[v/x]]$$

$$\{v.\ulcorner \varphi(v)\urcorner\}$$

- If $\langle K[e], \sigma\rangle$ is reducible

$$\frac{\langle e, \sigma\rangle \to_b \langle e', \sigma'\rangle}{\langle K[e], \sigma\rangle \to \langle K[e'], \sigma'\rangle}$$

We know $\mathit{safe}_{\varphi}(K[e'], \sigma')$

Case distinction on $\to_b$:

- Case $e = \text{let } x := v \text{ in } e_1$: $e' = e_1[v/x], \sigma' = \sigma$
  - Apply BIND rule — We are back where we started!
  - Apply LET rule
  - Apply BIND rule in reverse

We did it!

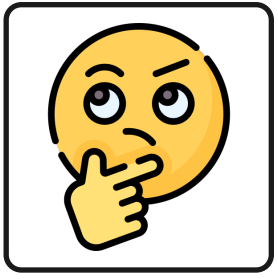
We did it!

Outline

Review the Laws

Do the Proof

Workshop the Proof



Reflect on the Proof

The Proof, Summary

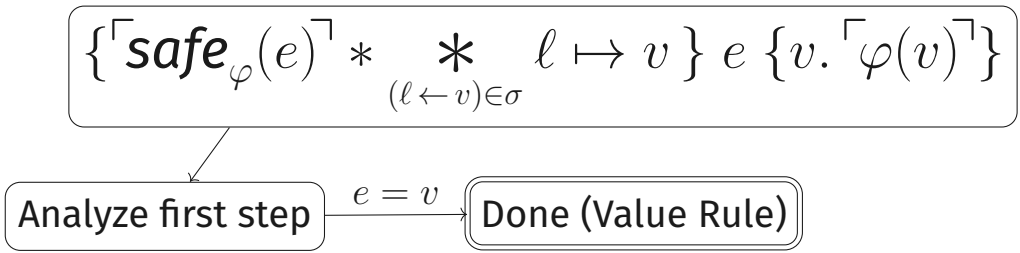
$$\{\ulcorner \mathit{safe}_{\varphi}(e) \urcorner * \mathop{\Large *}_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v \}\ e\ \{v. \ulcorner \varphi(v) \urcorner\}$$

The Proof, Summary

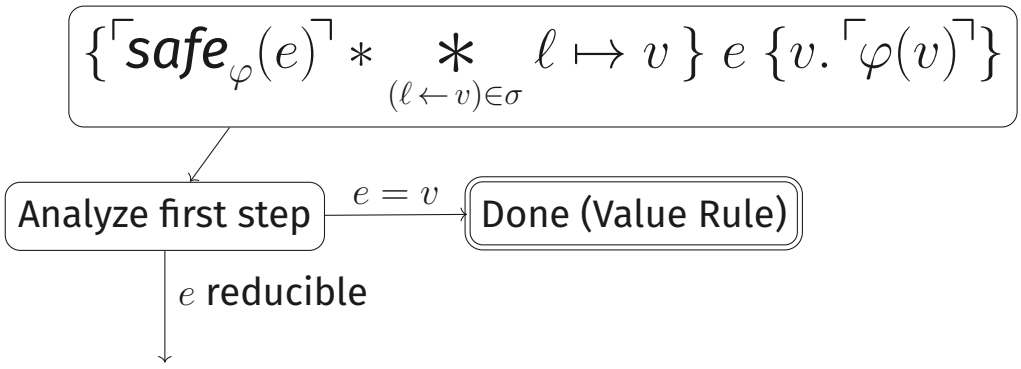
$$\{\ulcorner \mathit{safe}_{\varphi}(e) \urcorner * \mathop{\Large *}_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v\}\ e\ \{v.\ \ulcorner \varphi(v) \urcorner\}$$

Analyze first step

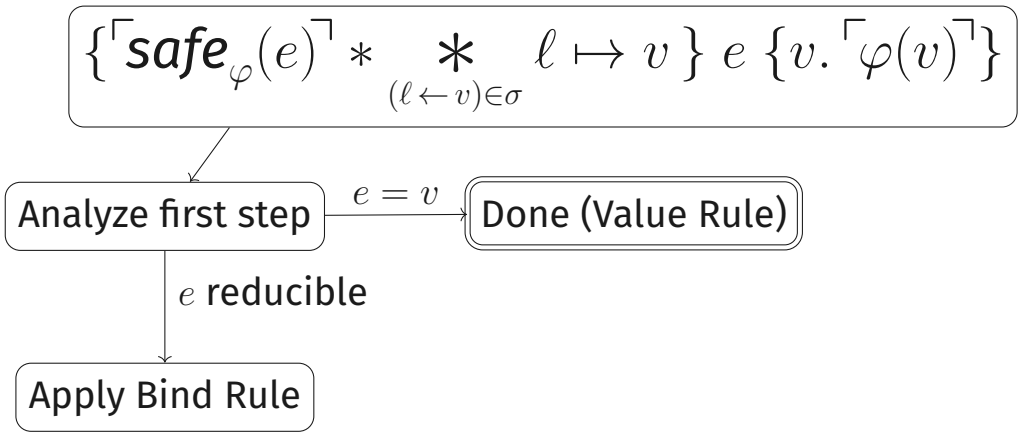
The Proof, Summary



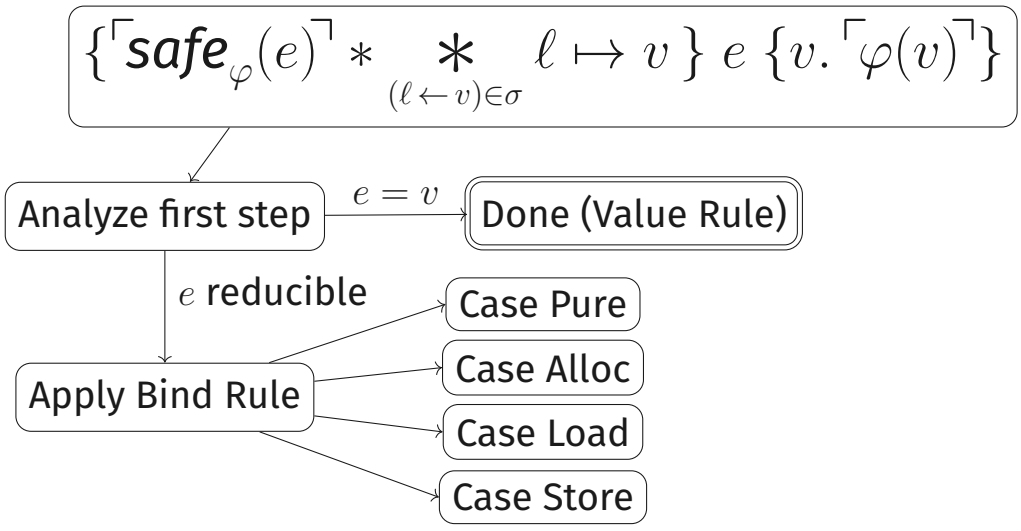
The Proof, Summary



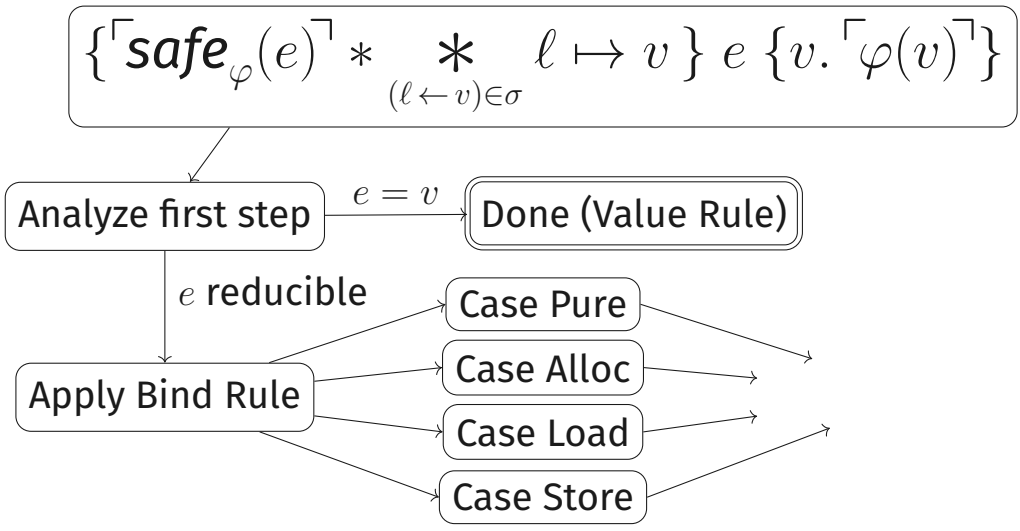
The Proof, Summary



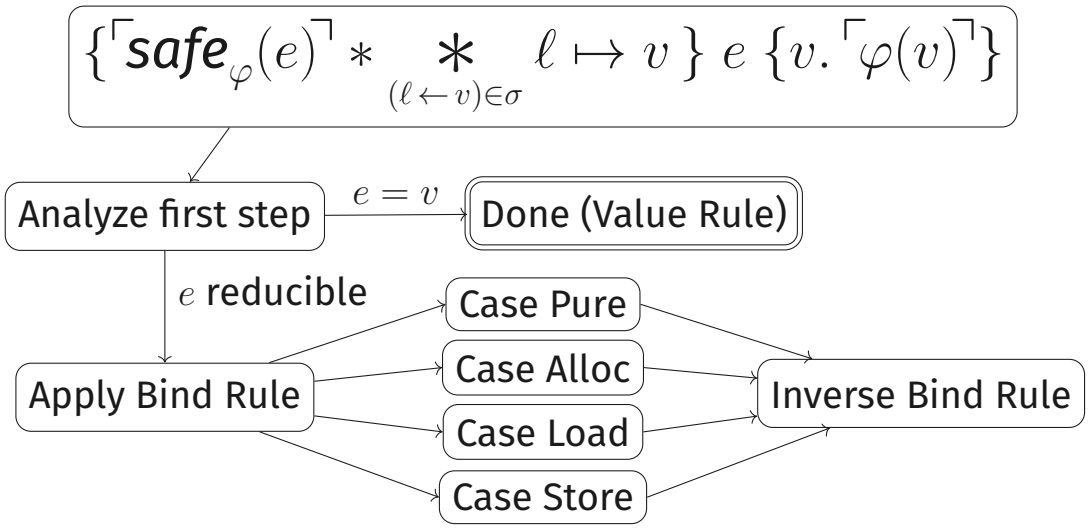
The Proof, Summary



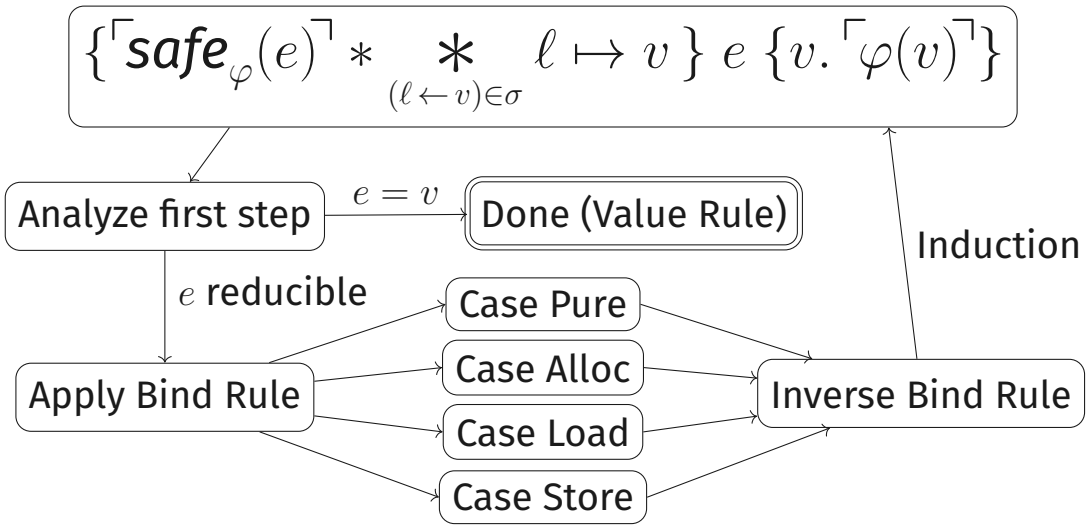
The Proof, Summary



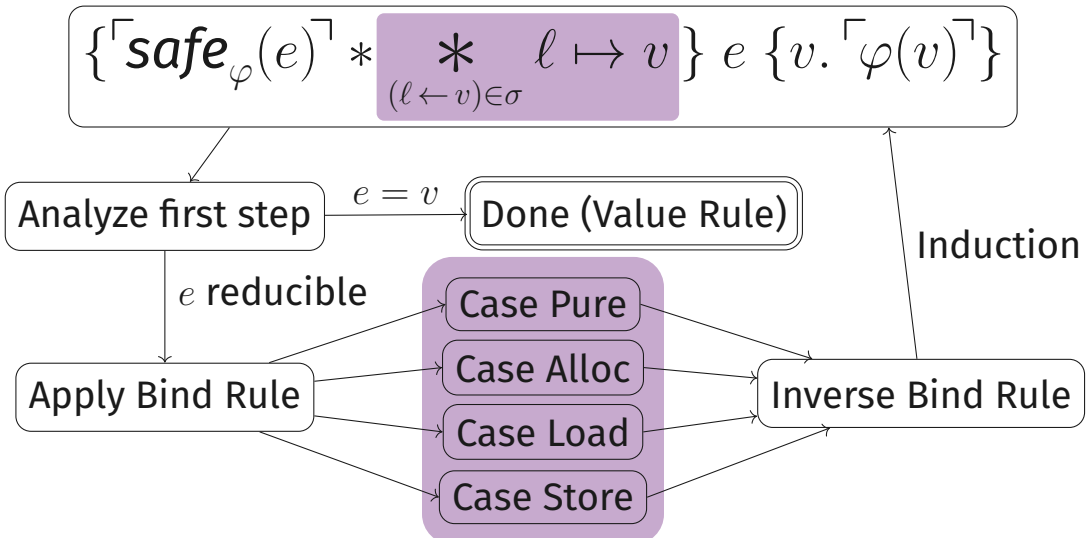
The Proof, Summary



The Proof, Summary



The Proof, Summary and Generalizations



Strengthenings

So far: Completeness for sequential, terminating programs.

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination:

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are also co-inductive.

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Concurrency:

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Concurrency: We must share resources among threads.

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Concurrency: We must share resources among threads.

Solution: Invariants

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Concurrency: We must share resources among threads.

Solution: Invariants

$$\exists \vec{e}, \sigma.\ \ulcorner \text{safe-tp}_{\varphi}(\vec{e}, \sigma) \urcorner * \bullet^{\gamma} \vec{e} * \ast_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v$$

Strengthenings

So far: Completeness for sequential, terminating programs.

Non-termination: safe is co-inductive.

Solution: Hoare triples are ~~also co-inductive~~ *step-indexed*.

Concurrency: We must share resources among threads.

Solution: Invariants and more technical difficulties…

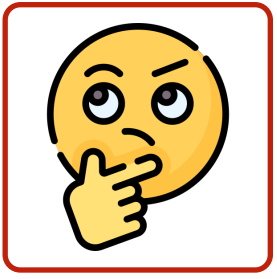
$$\exists \vec{e}, \sigma.\ \ulcorner \mathsf{safe\text{-}tp}_{\varphi}(\vec{e}, \sigma) \urcorner * \bullet^{\gamma} \vec{e} * \ast_{(\ell \leftarrow v) \in \sigma} \ell \mapsto v$$

Outline

Review the Laws

Do the Proof

Workshop the Proof



Reflect on the Proof

Discussion: Case Studies

What can we apply our framework to?

Discussion: Case Studies

What can we apply our framework to?

- HeapLang: total + partial completeness
  - Prophecies

Discussion: Case Studies

What can we apply our framework to?

- HeapLang: total + partial completeness
  - Prophecies
- λ_{Rust}: partial completeness
  - Data Races

Discussion: Case Studies

What can we apply our framework to?

- HeapLang: total + partial completeness
  - Prophecies
- λ_{Rust}: partial completeness
  - Data Races
- Time Credits: partial completeness

Discussion: Case Studies

What can we apply our framework to?

- HeapLang: total + partial completeness
  - Prophecies
- λ_{Rust}: partial completeness
  - Data Races
- Time Credits: partial completeness
- Eris: partial + total completeness
  - probabilities (prevents framework use, but approach still works)

Discussion: Case Studies

What can we apply our framework to?

- HeapLang: total + partial completeness
  - Prophecies
- λ_{Rust}: partial completeness
  - Data Races
- Time Credits: partial completeness
- Eris: partial + total completeness
  - probabilities (prevents framework use, but approach still works)

Small incompletenesses found and fixed!

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)
- First-order invariants (for concurrency)

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)
- First-order invariants (for concurrency)
- Custom ghost state, specifically *ghost maps* (concurrency)

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)
- First-order invariants (for concurrency)
- Custom ghost state, specifically *ghost maps* (concurrency)

Not needed:

- Impredicate invariants

Discussion: Necessary Features for Completeness

- VALUE, BIND, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)
- First-order invariants (for concurrency)
- Custom ghost state, specifically *ghost maps* (concurrency)

Not needed:

- Impredicate invariants
- Higher-order ghost state

Discussion: Necessary Features for Completeness

- Value, Bind, and their *inverses*
- Löb induction a.k.a. co-induction (for non-termination)
- First-order invariants (for concurrency)
- Custom ghost state, specifically *ghost maps* (concurrency)

Not needed:

- Impredicate invariants
- Higher-order ghost state
- Later credits, > 1 laters per step, Transfinite Iris

Contributions and Future Work

- ✓ General Framework for partial or total completeness

Contributions and Future Work

✓ General Framework for partial or total completeness

HeapLang, λ_{Rust}, Time Credits, …

Contributions and Future Work

✓ General Framework for partial or total completeness

HeapLang, λ_{Rust}, Time Credits, … Your logic?

Contributions and Future Work

- ✓ General Framework for partial or total completeness

  HeapLang, λ_{Rust}, Time Credits, … Your logic?

- ✓ Completeness of Eris (probabilistic)

Contributions and Future Work

- ✓ General Framework for partial or total completeness

  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic

Contributions and Future Work

- ✓ General Framework for partial or total completeness

  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic
- ✓ Requisiteness of wp definition (in the model)

Contributions and Future Work

- ✓ General Framework for partial or total completeness
  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic
- ✓ Requisiteness of wp definition (in the model)

- ◉ Further Work Remains: Eliminate use of Inverse BIND rule

Contributions and Future Work

- ✓ General Framework for partial or total completeness
  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic
- ✓ Requisiteness of wp definition (in the model)

- ◉ Further Work Remains: Eliminate use of Inverse BIND rule
- ◉ Follow-Up Work on Logical Atomicity: `arXiv:2607.11435`
- ◉ Follow-Up Work on Incorrectness Logic: `arXiv:2607.11611`

Contributions and Future Work

- ✓ General Framework for partial or total completeness
  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic
- ✓ Requisiteness of wp definition (in the model)

- Further Work Remains: Eliminate use of Inverse BIND rule
- Follow-Up Work on Logical Atomicity: `arXiv:2607.11435`
- Follow-Up Work on Incorrectness Logic: `arXiv:2607.11611`

Contributions and Future Work

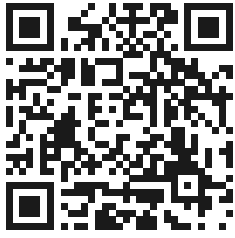
- ✓ General Framework for partial or total completeness
  HeapLang, λ_{Rust}, Time Credits, … Your logic?
- ✓ Completeness of Eris (probabilistic)
- ✓ Completeness of a ReLoC-like relational logic
- ✓ Requisiteness of wp definition (in the model)

- Further Work Remains: Eliminate use of Inverse BIND rule
- Follow-Up Work on Logical Atomicity: `arXiv:2607.11435`
- Follow-Up Work on Incorrectness Logic: `arXiv:2607.11611`

The End

Thanks for your attention!

More info:



All icons by https://www.flaticon.com/authors/freepik