Iris Tutorial

Tej Chajed[1] Ralf Jung[2] Joseph Tassarotti[3]

[1]Massachusetts Institute of Technology, USA

[2]MPI-SWS, Germany

[3]Boston College, USA

January 18, 2021 @ POPL Tutorials

Preparation for this tutorial

- Clone the tutorial lecture material
  `https://iris-project.org/tutorial-popl21`
- Follow README to install Iris

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

- **Higher-order separation logic:** Supports modular reasoning about higher-order stateful programs

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

- **Higher-order separation logic:** Supports modular reasoning about higher-order stateful programs
- **Fine-grained concurrent programs:** Programs that use low-level synchronization primitives for more parallelism

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

- **Higher-order separation logic:** Supports modular reasoning about higher-order stateful programs
- **Fine-grained concurrent programs:** Programs that use low-level synchronization primitives for more parallelism
- **Language-independent:** Parameterized by the language

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

- **Higher-order separation logic:** Supports modular reasoning about higher-order stateful programs
- **Fine-grained concurrent programs:** Programs that use low-level synchronization primitives for more parallelism
- **Language-independent:** Parameterized by the language
- **Simple foundations:** Small, “canonical” set of primitive rules

What is Iris?

Language-independent higher-order separation logic with simple foundations for verifying fine-grained concurrent programs in Coq.

- **Higher-order separation logic:** Supports modular reasoning about higher-order stateful programs
- **Fine-grained concurrent programs:** Programs that use low-level synchronization primitives for more parallelism
- **Language-independent:** Parameterized by the language
- **Simple foundations:** Small, "canonical" set of primitive rules
- **Coq:** Provides practical support for machine-checked proof

The versatility of Iris

Iris has been used to formalize many projects, ranging from program logics to logical relations to program proofs.

- RustBelt
- Perennial
- Many other examples

RustBelt: formalizing the Rust type system

Rust is a safe systems programming language with a sophisticated type system based on **ownership**, **borrowing**, and **lifetimes**.

- Safety of high-level Rust code relies on safe encapsulation of **unsafe code** in the lower layers.
- RustBelt uses Iris to build a **logical relation** for the Rust type system, formalizing the invariants encoded by the types.
- Borrowing and lifetimes are formalized by the lifetime logic, which puts Iris' flexibility to the test.
- RustBelt is able to verify the safety of `Mutex` and other Rust standard library abstractions.

Perennial: logic for crash-safety reasoning

Storage systems need proofs of correctness both under failures (due to kernel panic or disconnecting disk) and normal execution.

- Perennial uses Iris to build a variant of Hoare logic with a **crash condition** that holds at all intermediate points, even on failure.
- Iris gives the Perennial logic the flexibility to combine concurrency and failure reasoning.
- Perennial is built on top of a custom language which models the executable code written in **Go**.

Many other diverse projects using Iris

- Concurrent Search Templates uses Iris to prove some data structures correct
- Aneris is a program logic for distributed systems built using Iris
- Scala Step-by-Step formalizes soundness of the Scala type system using Iris to handle step-indexing
- Hazel is a sequential separation logic for effect handlers that uses Iris to handle recursive predicates (at this POPL 2021, first session on Friday!)

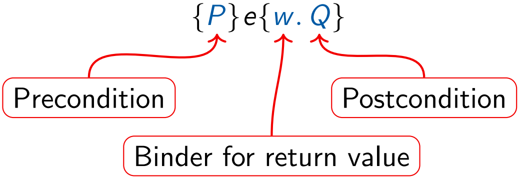
Outline

- **The Iris story, Part 1:** Working with invariants and ghost state
- **The Iris story, Part 2:** Modeling ghost state via “PCMs”
- **Iris in Coq:** The Interactive Proof Mode (IPM), live demo
- **Hands-on Iris:** Work on the exercises (we will be available for help throughout the conference) `https://iris-project.org/tutorial-popl21/`

The Iris story, Part 1:
Working with invariants and ghost state

Hoare triples

Hoare triples for partial program correctness:



$$\{P\}\,e\,\{w.\,Q\}$$

Precondition

Binder for return value

Postcondition

If the initial state satisfies P, then:

- e does not get stuck/crash
- if e terminates with value v, the final state satisfies $Q[v/w]$

Separation logic [O'Hearn, Reynolds, Yang]

The points-to connective $x \mapsto v$

- provides the knowledge that location x has value v, and
- provides exclusive ownership of x

Separating conjunction $P * Q$: the state consists of *disjoint parts* satisfying P and Q

Separation logic [O'Hearn, Reynolds, Yang]

The points-to connective $x \mapsto v$

- provides the knowledge that location x has value v, and
- provides exclusive ownership of x

Separating conjunction $P * Q$: the state consists of *disjoint parts* satisfying P and Q

Example:

$$\{x \mapsto v_1 * y \mapsto v_2\}\mathit{swap}(x, y)\{w.\, w = () \wedge x \mapsto v_2 * y \mapsto v_1\}$$

the $*$ ensures that x and y are different

Concurrent separation logic [O'Hearn, Brookes]

The *par* rule:

$$\frac{\{P_1\}e_1\{Q_1\} \qquad \{P_2\}e_2\{Q_2\}}{\{P_1 * P_2\}e_1||e_2\{Q_1 * Q_2\}}$$

Concurrent separation logic [O'Hearn, Brookes]

The *par* rule:

$$\frac{\{P_1\}\, e_1 \{Q_1\} \qquad \{P_2\}\, e_2 \{Q_2\}}{\{P_1 * P_2\}\, e_1 || e_2 \{Q_1 * Q_2\}}$$

For example:

$$\begin{array}{c}
\{x \mapsto 4 * y \mapsto 6\} \\
x := !\,x + 2 \;\Bigg\|\; y := !\,y + 2 \\
\{x \mapsto 6 * y \mapsto 8\}
\end{array}$$

Concurrent separation logic [O'Hearn, Brookes]

The *par* rule:

$$\frac{\{P_1\}\, e_1 \{Q_1\} \qquad \{P_2\}\, e_2 \{Q_2\}}{\{P_1 * P_2\}\, e_1 || e_2 \{Q_1 * Q_2\}}$$

For example:

$$\begin{array}{c} \{x \mapsto 4 * y \mapsto 6\} \\ \begin{array}{l||l} \{x \mapsto 4\} & \{y \mapsto 6\} \\ x := !\, x + 2 & y := !\, y + 2 \\ & \end{array} \\ \{x \mapsto 6 * y \mapsto 8\} \end{array}$$

Concurrent separation logic [O'Hearn, Brookes]

The *par* rule:

$$\frac{\{P_1\}e_1\{Q_1\} \qquad \{P_2\}e_2\{Q_2\}}{\{P_1 * P_2\}e_1||e_2\{Q_1 * Q_2\}}$$

For example:

$$\begin{array}{c}
\{x \mapsto 4 * y \mapsto 6\} \\
\begin{array}{l||l}
\{x \mapsto 4\} & \{y \mapsto 6\} \\
x := !\,x + 2 & y := !\,y + 2 \\
\{x \mapsto 6\} & \{y \mapsto 8\}
\end{array} \\
\{x \mapsto 6 * y \mapsto 8\}
\end{array}$$

Concurrent separation logic [O'Hearn, Brookes]

The *par* rule:

$$\frac{\{P_1\}\, e_1 \{Q_1\} \qquad \{P_2\}\, e_2 \{Q_2\}}{\{P_1 * P_2\}\, e_1 || e_2 \{Q_1 * Q_2\}}$$

For example:

$$\begin{array}{c}
\{x \mapsto 4 * y \mapsto 6\} \\
\begin{array}{l||l}
\{x \mapsto 4\} & \{y \mapsto 6\} \\
x := \,!\, x + 2 & y := \,!\, y + 2 \\
\{x \mapsto 6\} & \{y \mapsto 8\}
\end{array} \\
\{x \mapsto 6 * y \mapsto 8\}
\end{array}$$

Works great for concurrent programs without shared memory: concurrent quick sort, . . .

What about shared state/racy programs?

A classic problem:

$$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$$

$$\begin{array}{l|l} \texttt{fetchandadd}(x, 2) & \texttt{fetchandadd}(x, 2) \end{array}$$

$$!\, x$$

Where $\texttt{fetchandadd}(x, y)$ is the atomic version of $x := !\, x + y$.

What about shared state/racy programs?

A classic problem:

$$
\begin{array}{l}
\{\mathsf{True}\} \\
\texttt{let}\, x = \texttt{ref}(0)\,\texttt{in} \\
\left.\texttt{fetchandadd}(x, 2)\;\right\|\; \texttt{fetchandadd}(x, 2) \\
!\,x \\
\{w.\, w = 4\}
\end{array}
$$

Where $\texttt{fetchandadd}(x, y)$ is the atomic version of $x := !\,x + y$.

What about shared state/racy programs?

A classic problem:

$$
\begin{array}{l}
\{\mathsf{True}\} \\
\texttt{let}\, x = \texttt{ref}(0)\,\texttt{in} \\
\{x \mapsto 0\} \\
\texttt{fetchandadd}(x, 2) \; \Bigg\| \; \texttt{fetchandadd}(x, 2) \\
!\, x \\
\{w.\, w = 4\}
\end{array}
$$

Where $\texttt{fetchandadd}(x, y)$ is the atomic version of $x := !\, x + y$.

What about shared state/racy programs?

A classic problem:

$$
\begin{array}{l}
\{\mathsf{True}\}\\
\texttt{let}\, x = \texttt{ref}(0)\,\texttt{in}\\
\{x \mapsto 0\}\\
\begin{array}{l}\{??\}\\ \texttt{fetchandadd}(x, 2)\\ \{??\}\end{array} \;\left\|\; \begin{array}{l}\{??\}\\ \texttt{fetchandadd}(x, 2)\\ \{??\}\end{array}\right.\\
!\,x\\
\{w.\, w = 4\}
\end{array}
$$

Where $\texttt{fetchandadd}(x, y)$ is the atomic version of $x := !\,x + y$.

Problem: can only give ownership of x to one thread

Invariants

The invariant assertion $\boxed{R}$ expresses that R is maintained as an invariant on the state

Invariants

The invariant assertion $\boxed{R}$ expresses that R is maintained as an invariant on the state

Invariant opening:

$$\frac{\{R * P\}\, e\, \{R * Q\} \qquad e \text{ atomic}}{\left\{\boxed{R} * P\right\}\, e\, \left\{\boxed{R} * Q\right\}}$$

Invariants

The invariant assertion $\boxed{R}$ expresses that R is maintained as an invariant on the state

Invariant opening:

$$\frac{\{R * P\}\, e\, \{R * Q\} \qquad e \text{ atomic}}{\left\{\boxed{R} * P\right\} e \left\{\boxed{R} * Q\right\}}$$

Invariant allocation:

$$\frac{\left\{\boxed{R} * P\right\} e\, \{Q\}}{\{R * P\}\, e\, \{Q\}}$$

Invariants

The invariant assertion $\boxed{R}^{\mathcal{N}}$ expresses that R is maintained as an invariant on the state

Invariant opening:

$$\frac{\{R * P\}\, e\, \{R * Q\} \qquad e \text{ atomic}}{\left\{\boxed{R} * P\right\} e \left\{\boxed{R} * Q\right\}}$$

Invariant allocation:

$$\frac{\left\{\boxed{R} * P\right\} e\, \{Q\}}{\{R * P\}\, e\, \{Q\}}$$

Invariant duplication: $\boxed{R} \vdash \boxed{R} * \boxed{R}$

Invariants

The invariant assertion $\boxed{R}^{\mathcal{N}}$ expresses that R is maintained as an invariant on the state

Invariant opening:

$$\frac{\{R * P\}\, e\, \{R * Q\}_{\mathcal{E}} \qquad e \text{ atomic}}{\left\{\boxed{R}^{\mathcal{N}} * P\right\}\, e\, \left\{\boxed{R}^{\mathcal{N}} * Q\right\}_{\mathcal{E} \uplus \mathcal{N}}}$$

Invariant allocation:

$$\frac{\left\{\boxed{R}^{\mathcal{N}} * P\right\}\, e\, \{Q\}_{\mathcal{E}}}{\{R * P\}\, e\, \{Q\}_{\mathcal{E}}}$$

Invariant duplication: $\boxed{R}^{\mathcal{N}} \vdash \boxed{R}^{\mathcal{N}} * \boxed{R}^{\mathcal{N}}$

Technicalities: names prevent opening the same invariant twice

Invariants

The invariant assertion $\boxed{R}^{\mathcal{N}}$ expresses that R is maintained as an invariant on the state

Invariant opening:

$$\frac{\{\triangleright R * P\}\ e\ \{\triangleright R * Q\}_{\mathcal{E}} \qquad e \text{ atomic}}{\left\{\boxed{R}^{\mathcal{N}} * P\right\}\ e\ \left\{\boxed{R}^{\mathcal{N}} * Q\right\}_{\mathcal{E}\uplus\mathcal{N}}}$$

Invariant allocation:

$$\frac{\left\{\boxed{R}^{\mathcal{N}} * P\right\}\ e\ \{Q\}_{\mathcal{E}}}{\{\triangleright R * P\}\ e\ \{Q\}}$$

Invariant duplication: $\boxed{R}^{\mathcal{N}} \vdash \boxed{R}^{\mathcal{N}} * \boxed{R}^{\mathcal{N}}$

Technicalities: names prevent opening the same invariant twice and the later $\triangleright$ is needed for impredicativity, i.e., $\boxed{\ldots\ \boxed{R}^{\mathcal{N}_2}\ \ldots}^{\mathcal{N}_1}$

Invariants in action

Let us consider a simpler problem first:

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$$

$!\, x$

$\{n.\, \mathit{even}(n)\}$

Invariants in action

Let us consider a simpler problem first:

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$

$!\, x$

$\{n.\, \mathit{even}(n)\}$

Invariants in action

Let us consider a simpler problem first:

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
allocate $\boxed{\exists n.\, x \mapsto n * even(n)}$

$$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$$

$!\, x$

$\{n.\, even(n)\}$

Invariants in action

Let us consider a simpler problem first:

$\{\mathsf{True}\}$

$\texttt{let}\; x = \texttt{ref}(0)\; \texttt{in}$

$\{x \mapsto 0\}$

allocate $\boxed{\exists n.\, x \mapsto n * even(n)}$

$$\begin{array}{l||l}
\left\{\boxed{\exists n.\, x \mapsto n * even(n)}\right\} & \left\{\boxed{\exists n.\, x \mapsto n * even(n)}\right\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\left\{\boxed{\exists n.\, x \mapsto n * even(n)}\right\} & \left\{\boxed{\exists n.\, x \mapsto n * even(n)}\right\}
\end{array}$$

$\quad !\, x$

$\{n.\, even(n)\}$

Invariants in action

Let us consider a simpler problem first:

$$\{\text{True}\}$$

$$\texttt{let}\, x = \texttt{ref}(0)\ \texttt{in}$$

$$\{x \mapsto 0\}$$

$$\text{allocate}\ \boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}$$

$$\begin{array}{l||l}
\left\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\right\} & \left\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\right\} \\
\quad\{x \mapsto n * \mathit{even}(n)\} & \\
\quad\texttt{fetchandadd}(x, 2) & \quad\texttt{fetchandadd}(x, 2) \\
\quad\{x \mapsto n + 2 * \mathit{even}(n + 2)\} & \\
\left\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\right\} & \left\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\right\}
\end{array}$$

$$!\, x$$

$$\{n.\, \mathit{even}(n)\}$$

Invariants in action

Let us consider a simpler problem first:

$\{\mathsf{True}\}$

$\mathtt{let}\, x = \mathtt{ref}(0)\, \mathtt{in}$

$\{x \mapsto 0\}$

allocate $\boxed{\exists n.\, x \mapsto n * even(n)}$

$$\begin{array}{l||l}
\{\boxed{\exists n.\, x \mapsto n * even(n)}\} & \{\boxed{\exists n.\, x \mapsto n * even(n)}\} \\
\quad\{x \mapsto n * even(n)\} & \quad\{x \mapsto n * even(n)\} \\
\quad\mathtt{fetchandadd}(x, 2) & \quad\mathtt{fetchandadd}(x, 2) \\
\quad\{x \mapsto n + 2 * even(n + 2)\} & \quad\{x \mapsto n + 2 * even(n + 2)\} \\
\{\boxed{\exists n.\, x \mapsto n * even(n)}\} & \{\boxed{\exists n.\, x \mapsto n * even(n)}\}
\end{array}$$

$!\, x$

$\{n.\, even(n)\}$

Invariants in action

Let us consider a simpler problem first:

$$
\begin{array}{l}
\{\mathsf{True}\} \\
\texttt{let}\, x = \texttt{ref}(0)\ \texttt{in} \\
\{x \mapsto 0\} \\
\text{allocate } \boxed{\exists n.\, x \mapsto n * \mathit{even}(n)} \\
\left(
\begin{array}{l}
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\} \\
\quad \{x \mapsto n * \mathit{even}(n)\} \\
\quad \texttt{fetchandadd}(x, 2) \\
\quad \{x \mapsto n + 2 * \mathit{even}(n + 2)\} \\
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\}
\end{array}
\ \middle\|\
\begin{array}{l}
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\} \\
\quad \{x \mapsto n * \mathit{even}(n)\} \\
\quad \texttt{fetchandadd}(x, 2) \\
\quad \{x \mapsto n + 2 * \mathit{even}(n + 2)\} \\
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\}
\end{array}
\right. \\
\quad \{x \mapsto n * \mathit{even}(n)\} \\
\quad !\, x \\
\quad \{n.\, x \mapsto n * \mathit{even}(n)\} \\
\{n.\, \mathit{even}(n)\}
\end{array}
$$

Invariants in action

Let us consider a simpler problem first:

$$\{\mathsf{True}\}$$
$$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$$
$$\{x \mapsto 0\}$$
$$\text{allocate } \boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}$$

$$\begin{array}{l||l}
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\} & \{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\} \\
\quad \{x \mapsto n * \mathit{even}(n)\} & \quad \{x \mapsto n * \mathit{even}(n)\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{x \mapsto n + 2 * \mathit{even}(n + 2)\} & \quad \{x \mapsto n + 2 * \mathit{even}(n + 2)\} \\
\{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\} & \{\boxed{\exists n.\, x \mapsto n * \mathit{even}(n)}\}
\end{array}$$

$$\begin{array}{l}
\quad \{x \mapsto n * \mathit{even}(n)\} \\
\quad !\, x \\
\quad \{n.\, x \mapsto n * \mathit{even}(n)\} \\
\{n.\, \mathit{even}(n)\}
\end{array}$$

Problem: still cannot prove it returns 4

Ghost variables

Consider the invariant:

$$\boxed{\exists n.\, x \mapsto n * \ldots}$$

How to avoid **information loss** due to existential quantification?

Ghost variables

Consider the invariant:

$$\boxed{\exists n.\, x \mapsto n * \ldots}$$

How to avoid **information loss** due to existential quantification?

Solution: ghost variables

Ghost variables

Consider the invariant:

$$\boxed{\exists n_1, n_2.\, x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$$

How to avoid **information loss** due to existential quantification?

Solution: ghost variables

Ghost variables come in "entangled" pairs:

$$\underbrace{\gamma \hookrightarrow_{\bullet} n}_{\substack{\text{in the invariant} \\ \text{(“\textbf{authoritative}”)}}} \quad * \quad \underbrace{\gamma \hookrightarrow_{\circ} n}_{\substack{\text{in the Hoare triple} \\ \text{(“\textbf{fragment}”)}}}$$

Ghost variables

Consider the invariant:

$$\boxed{\exists n_1, n_2.\, x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$$

How to avoid **information loss** due to existential quantification?

Solution: ghost variables

Ghost variables come in “entangled” pairs:

$$\text{True} \quad \Rrightarrow\!\!\ast \quad \exists \gamma.\quad \underbrace{\gamma \hookrightarrow_{\bullet} n}_{\substack{\text{in the invariant} \\ (\text{“}\textbf{authoritative}\text{”})}} \quad * \quad \underbrace{\gamma \hookrightarrow_{\circ} n}_{\substack{\text{in the Hoare triple} \\ (\text{“}\textbf{fragment}\text{”})}}$$

Ghost variables

Consider the invariant:

$$\boxed{\exists n_1, n_2.\, x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$$

How to avoid **information loss** due to existential quantification?

Solution: ghost variables

Ghost variables come in "entangled" pairs:

$$\text{True} \quad \mathrel{\Rrightarrow\!\!\!\!\ast} \quad \exists\gamma.\ \underbrace{\gamma \hookrightarrow_\bullet n}_{\substack{\text{in the invariant} \\ \text{(“\textbf{authoritative}”)}}} \quad * \quad \underbrace{\gamma \hookrightarrow_\circ n}_{\substack{\text{in the Hoare triple} \\ \text{(“\textbf{fragment}”)}}}$$

When you own both parts you obtain that the values are equal and can update both parts:

$$\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \quad \Rightarrow \quad n = m$$
$$\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \quad \mathrel{\Rrightarrow\!\!\!\!\ast} \quad \gamma \hookrightarrow_\bullet n' * \gamma \hookrightarrow_\circ n'$$

Ghost variables in action

$\{\text{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\texttt{fetchandadd}(x, 2) \quad \Bigg\| \quad \texttt{fetchandadd}(x, 2)$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\; x = \texttt{ref}(0)\; \texttt{in}$
$\{x \mapsto 0\}$

$$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$$

$!\,x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\text{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$$\texttt{fetchandadd}(x, 2) \quad \Big\| \quad \texttt{fetchandadd}(x, 2)$$

$!\,x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$\{\gamma_1 \hookrightarrow_{\circ} 0\}$ ‖ $\{\gamma_2 \hookrightarrow_{\circ} 0\}$

$\texttt{fetchandadd}(x, 2)$ ‖ $\texttt{fetchandadd}(x, 2)$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$$\begin{array}{l||l} \{\gamma_1 \hookrightarrow_{\circ} 0\} & \{\gamma_2 \hookrightarrow_{\circ} 0\} \\ \quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\ \{\gamma_1 \hookrightarrow_{\circ} 2\} & \{\gamma_2 \hookrightarrow_{\circ} 2\} \end{array}$$

$\{\gamma_1 \hookrightarrow_{\circ} 2 * \gamma_2 \hookrightarrow_{\circ} 2\}$

$\quad !\, x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_{\circ} 0\} & \{\gamma_2 \hookrightarrow_{\circ} 0\} \\
\quad \{\gamma_1 \hookrightarrow_{\circ} 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2\} & \\
\quad \texttt{fetchandadd}(x, 2) & \texttt{fetchandadd}(x, 2) \\
\{\gamma_1 \hookrightarrow_{\circ} 2\} & \{\gamma_2 \hookrightarrow_{\circ} 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_{\circ} 2 * \gamma_2 \hookrightarrow_{\circ} 2\}$

$!\,x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\texttt{fetchandadd}(x, 2)$

$\{\gamma_1 \hookrightarrow_\circ 2\}$

$\|$

$\{\gamma_2 \hookrightarrow_\circ 0\}$

$\texttt{fetchandadd}(x, 2)$

$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad\texttt{fetchandadd}(x, 2) & \quad\texttt{fetchandadd}(x, 2) \\
\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\,x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad \texttt{fetchandadd}(x, 2)$

$\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\{\gamma_1 \hookrightarrow_\circ 2\}$

$\|$

$\{\gamma_2 \hookrightarrow_\circ 0\}$

$\texttt{fetchandadd}(x, 2)$

$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\,x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$

$\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad\texttt{fetchandadd}(x, 2)$

$\quad\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\{\gamma_1 \hookrightarrow_\circ 2\}$

$\|$

$\{\gamma_2 \hookrightarrow_\circ 0\}$

$\quad\{\ldots\}$

$\quad\texttt{fetchandadd}(x, 2)$

$\quad\{\ldots\}$

$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$!\,x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\texttt{fetchandadd}(x, 2)$

$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\{\gamma_1 \hookrightarrow_\circ 2\}$

$\|$

$\{\gamma_2 \hookrightarrow_\circ 0\}$

$\{\dots\}$

$\texttt{fetchandadd}(x, 2)$

$\{\dots\}$

$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$!\, x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * \gamma_2 \hookrightarrow_{\circ} 0\}$

$\{\gamma_1 \hookrightarrow_{\circ} 0\}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2\}$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\bullet} n_2\}$

$\texttt{fetchandadd}(x, 2)$

$\{\gamma_1 \hookrightarrow_{\circ} 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_{\bullet} 0 * \gamma_2 \hookrightarrow_{\bullet} n_2\}$

$\{\gamma_1 \hookrightarrow_{\circ} 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_{\bullet} 2 * \gamma_2 \hookrightarrow_{\bullet} n_2\}$

$\{\gamma_1 \hookrightarrow_{\circ} 2\}$

$\|$

$\{\gamma_2 \hookrightarrow_{\circ} 0\}$

$\{\ldots\}$

$\texttt{fetchandadd}(x, 2)$

$\{\ldots\}$

$\{\gamma_2 \hookrightarrow_{\circ} 2\}$

$\{\gamma_1 \hookrightarrow_{\circ} 2 * \gamma_2 \hookrightarrow_{\circ} 2\}$

$\{\gamma_1 \hookrightarrow_{\circ} 2 * \gamma_2 \hookrightarrow_{\circ} 2 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_{\bullet} n_1 * \gamma_2 \hookrightarrow_{\bullet} n_2\}$

$!\, x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\ x = \texttt{ref}(0)\ \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$

$\quad !\,x$

$\{n.\ n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$

$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$

$\{x \mapsto 0\}$

$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$

$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$

$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * {\color{red} x \mapsto 4} * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$

$\quad !\, x$

$\{n.\, n = 4\}$

Ghost variables in action

$$\{\mathsf{True}\}$$

$$\texttt{let}\; x = \texttt{ref}(0)\; \texttt{in}$$

$$\{x \mapsto 0\}$$

$$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$$

$$\text{allocate } \boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$$

$$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$$

$$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$$

$$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$$

$$\quad !\, x$$

$$\quad \{n.\, n = 4 * \gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$$

$$\{n.\, n = 4\}$$

Ghost variables in action

$$\{\mathsf{True}\}$$

$$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$$

$$\{x \mapsto 0\}$$

$$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$$

$$\text{allocate } \boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$$

$$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$$

$$\begin{array}{l||l}
\{\gamma_1 \hookrightarrow_\circ 0\} & \{\gamma_2 \hookrightarrow_\circ 0\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \texttt{fetchandadd}(x, 2) & \quad \texttt{fetchandadd}(x, 2) \\
\quad \{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\} & \quad \{\ldots\} \\
\quad \{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2 + n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\} & \\
\{\gamma_1 \hookrightarrow_\circ 2\} & \{\gamma_2 \hookrightarrow_\circ 2\}
\end{array}$$

$$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$$

$$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$$

$$\quad \{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$$

$$\quad !\, x$$

$$\quad \{n.\, n = 4 * \gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$$

$$\{n.\, n = 4\}$$

The Iris story, Part 2:

Modeling ghost state via “PCMs”

Mechanisms for concurrent reasoning

We have seen so far:

- Invariants $\boxed{R}^{\mathcal{N}}$
- Ghost variables $\gamma \hookrightarrow_{\bullet} n$ and $\gamma \hookrightarrow_{\circ} n$

You may also have heard of:

- Fractional permissions $a \mapsto_{\pi} v$
- State-transition systems and monotonic state

How can we make sure we have all the mechanisms we will need?

Mechanisms for concurrent reasoning

We have seen so far:

- Invariants $\boxed{R}^{\mathcal{N}}$
- Ghost variables $\gamma \hookrightarrow_{\bullet} n$ and $\gamma \hookrightarrow_{\circ} n$

You may also have heard of:

- Fractional permissions $a \mapsto_{\pi} v$
- State-transition systems and monotonic state

How can we make sure we have all the mechanisms we will need?
The Iris story: these mechanisms can be **encoded** using a simple mechanism of *ghost resource ownership*

Resource algebras (RAs): A generalization of PCMs

Resource algebra (RA) with carrier M:

- Composition $(\cdot) : M \to M \to M$
- Validity predicate $\mathcal{V} \subseteq M$

Satisfying:

$$a \cdot b = b \cdot a \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c \qquad (a \cdot b) \in \mathcal{V} \Rightarrow a \in \mathcal{V}$$

Resource algebras (RAs): A generalization of PCMs

Resource algebra (RA) with carrier M:

- Composition $(\cdot) : M \to M \to M$
- Validity predicate $\mathcal{V} \subseteq M$

Satisfying:

$$a \cdot b = b \cdot a \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c \qquad (a \cdot b) \in \mathcal{V} \Rightarrow a \in \mathcal{V}$$

Iris provides $\boxed{a : M}^{\gamma}$ expressing ownership of an element a of resource algebra M (with name γ)

Ghost variables revisited

Resource algebra for ghost variables:

$$M \triangleq \bullet\, n \mid \circ\, n \mid \bullet\!\!\circ\, n \mid \bot$$

$$\mathcal{V} \triangleq \{a \neq \bot \mid a \in M\}$$

$$\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n \triangleq \begin{cases} \bullet\!\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases}$$

$$\text{other combinations} \triangleq \bot$$

Ghost variables revisited

Resource algebra for ghost variables:

$$
\begin{aligned}
M &\triangleq \bullet\, n \mid \circ\, n \mid \bullet\!\circ\, n \mid \bot \\
\mathcal{V} &\triangleq \{a \neq \bot \mid a \in M\} \\
\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n &\triangleq \begin{cases} \bullet\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases} \\
\text{other combinations} &\triangleq \bot
\end{aligned}
$$

And define:

$$\gamma \hookrightarrow_{\bullet} n \triangleq \boxed{\bullet\, n}^{\gamma} \qquad\qquad \gamma \hookrightarrow_{\circ} n \triangleq \boxed{\circ\, n}^{\gamma}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow \exists \gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\ast\; \exists\gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\ast\; \exists\gamma.\, \boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, n}^{\gamma}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\!\ast\ \exists\gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\!\ast\ \exists\gamma.\, \boxed{\bullet\!\circ\, n}^{\gamma} \Rrightarrow\!\!\!\ast\ \exists\gamma.\, \boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, n}^{\gamma}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\ast \exists\gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\ast \exists\gamma.\, \boxed{\bullet\!\circ\, n}^{\gamma} \Rrightarrow\!\!\ast \exists\gamma.\, \boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, n}^{\gamma}$$

Remember:

$$\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n \triangleq \begin{cases} \bullet\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{\bullet\!\!\circ\, n}^{\gamma} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, n}^{\gamma}$$

$$\boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, m}^{\gamma} \Rightarrow n = m$$

Remember:

$$\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n \triangleq \begin{cases} \bullet\!\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} * \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{\bullet\!\!\circ\, n}^{\gamma} \Rrightarrow\!\!\!\!\ast\; \exists \gamma.\, \boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, n}^{\gamma}$$

$$\boxed{\bullet\, n}^{\gamma} * \boxed{\circ\, m}^{\gamma} \Rightarrow (\bullet\, n \cdot \circ\, m) \in \mathcal{V} \Rightarrow n = m$$

Remember:

$$\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n \triangleq \begin{cases} \bullet\!\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases}$$

$$\mathcal{V} \triangleq \{a \neq \bot \mid a \in M\}$$

Ghost resource laws

Iris provides general laws for ghost resources:

$$a \in \mathcal{V} \Rrightarrow\!\!\ast\ \exists\gamma.\, \boxed{a}^{\gamma} \qquad \boxed{a \cdot b}^{\gamma} \Leftrightarrow \boxed{a}^{\gamma} \ast \boxed{b}^{\gamma} \qquad \boxed{a}^{\gamma} \Rightarrow \mathcal{V}(a)$$

The ghost variable laws follow from these:

$$\text{True} \Rrightarrow\!\!\ast\ \exists\gamma.\, \boxed{\bullet\!\!\circ\, n}^{\gamma} \Rrightarrow\!\!\ast\ \exists\gamma.\, \boxed{\bullet\, n}^{\gamma} \ast \boxed{\circ\, n}^{\gamma}$$

$$\boxed{\bullet\, n}^{\gamma} \ast \boxed{\circ\, m}^{\gamma} \Rightarrow (\bullet\, n \cdot \circ\, m) \in \mathcal{V} \Rightarrow n = m$$

Remember:

$$\bullet\, n \cdot \circ\, n' = \circ\, n' \cdot \bullet\, n \triangleq \begin{cases} \bullet\!\!\circ\, n & \text{if } n = n' \\ \bot & \text{otherwise} \end{cases}$$

$$\mathcal{V} \triangleq \{a \neq \bot \mid a \in M\}$$

Updating resources

Resources can be *updated* using frame-preserving updates:

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \Rrightarrow\!\!\ast \boxed{b}^{\gamma}} \qquad a \rightsquigarrow b \triangleq \forall a_{\mathrm{f}}.\, a \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow b \cdot a_{\mathrm{f}} \in \mathcal{V}$$

Updating resources

Resources can be *updated* using frame-preserving updates:

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \Rrightarrow\!\!\ast\ \boxed{b}^{\gamma}} \qquad a \rightsquigarrow b \triangleq \forall a_{\mathrm{f}}.\, a \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow b \cdot a_{\mathrm{f}} \in \mathcal{V}$$

Key idea: a resource can be updated if the update does not invalidate the resources of concurrently-running threads

$$\begin{array}{ccccccccc}
\text{Thread 1} & & \text{Thread 2} & & \ldots & & \text{Thread n} & \\
a & \cdot & a_2 & \cdot & \ldots & \cdot & a_n & \in \mathcal{V} \\
\rotatebox{-90}{$\rightsquigarrow$} & & & & & & & \\
b & \cdot & a_2 & \cdot & \ldots & \cdot & a_n & \in \mathcal{V}
\end{array}$$

Updating resources

Resources can be *updated* using frame-preserving updates:

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \Rrightarrow\!\!\ast \boxed{b}^{\gamma}} \qquad a \rightsquigarrow b \triangleq \forall a_{\mathrm{f}}.\, a \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow b \cdot a_{\mathrm{f}} \in \mathcal{V}$$

For ghost variables:

$$\frac{\bullet\!\circ\, n \rightsquigarrow \bullet\!\circ\, n'}{\gamma \hookrightarrow_{\bullet} n * \gamma \hookrightarrow_{\circ} n \Rrightarrow\!\!\ast \gamma \hookrightarrow_{\bullet} n' * \gamma \hookrightarrow_{\circ} n'}$$

Updating resources

Resources can be *updated* using frame-preserving updates:

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \Rrightarrow\!\!\ast\ \boxed{b}^{\gamma}} \qquad a \rightsquigarrow b \triangleq \forall a_{\mathrm{f}}.\, a \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow b \cdot a_{\mathrm{f}} \in \mathcal{V}$$

For ghost variables:

$$\frac{\bullet\!\circ\, n \rightsquigarrow \bullet\!\circ\, n'}{\gamma \hookrightarrow_{\bullet} n * \gamma \hookrightarrow_{\circ} n \Rrightarrow\!\!\ast\ \gamma \hookrightarrow_{\bullet} n' * \gamma \hookrightarrow_{\circ} n'}$$

$$\bullet\!\circ\, n \rightsquigarrow \bullet\!\circ\, n' = \forall a_{\mathrm{f}}.\, \bullet\!\circ\, n \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow \bullet\!\circ\, n' \cdot a_{\mathrm{f}} \in \mathcal{V}$$

Updating resources

Resources can be *updated* using frame-preserving updates:

$$\frac{a \rightsquigarrow b}{\boxed{a}^{\gamma} \mathrel{\Rrightarrow\!\!\!\ast} \boxed{b}^{\gamma}} \qquad a \rightsquigarrow b \triangleq \forall a_{\mathrm{f}}.\, a \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow b \cdot a_{\mathrm{f}} \in \mathcal{V}$$

For ghost variables:

$$\frac{\bullet\!\circ\, n \rightsquigarrow \bullet\!\circ\, n'}{\gamma \hookrightarrow_{\bullet} n * \gamma \hookrightarrow_{\circ} n \mathrel{\Rrightarrow\!\!\!\ast} \gamma \hookrightarrow_{\bullet} n' * \gamma \hookrightarrow_{\circ} n'}$$

$$\bullet\!\circ\, n \rightsquigarrow \bullet\!\circ\, n' = \forall a_{\mathrm{f}}.\, \bullet\!\circ\, n \cdot a_{\mathrm{f}} \in \mathcal{V} \Rightarrow \bullet\!\circ\, n' \cdot a_{\mathrm{f}} \in \mathcal{V}$$

$\bullet\!\circ\, n \cdot a_{\mathrm{f}} = \bot$, so the premise holds vacuously.

Generalizing to a library of RA combinators

Iris comes with a library of useful RA combinators

- $\textsc{Auth}(M)$: Generalizes the $\bullet$, $\circ$, $\bullet\!\circ$ construction over an arbitrary RA M – we call it the "authoritative" RA.
- $\textsc{Excl}(X)$: The "exclusive" RA, whose valid elements are the elements of X, and where composition is always undefined.
- $\textsc{Frac}$: The RA for fractions in $(0, 1]$ with addition.
- The expected RA liftings of products, sums, etc.

Using these combinators, we can easily construct the necessary models of many desired forms of ghost state:

- Ghost variables from this talk: $\textsc{Auth}\ (\textsc{Excl}\ \textsc{Nat})$
- Fractional ghost variables: $\textsc{Auth}\ (\textsc{Frac} \times \textsc{Nat}_{+})$

Iris in Coq:
The Interactive Proof Mode (IPM)

Iris Proof Mode (IPM)

Many recent program logics come with mechanized soundness proofs, but how to reason in these logics?

Goal of IPM: reasoning in Iris in the same style as reasoning in Coq

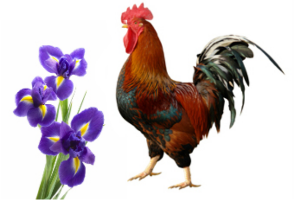
Iris Proof Mode (IPM)

Many recent program logics come with mechanized soundness proofs, but how to reason in these logics?

Goal of IPM: reasoning in Iris in the same style as reasoning in Coq

Features of IPM:

- Extends Coq with spatial contexts for Iris
- Tactics for introduction and elimination of all connectives of Iris
- Implemented entirely using reflection, type classes and Ltac (no OCaml plugin needed)



What's next?

- Exercises for this tutorial, in Coq
- Iris from the Ground Up
- Iris Lecture Notes