Iris: A Modular Foundation for Higher-Order Concurrent Separation Logic

Derek Dreyer[1]

Max Planck Institute for Software Systems (MPI-SWS), Germany

CMMRS 2017

[1]Iris is joint work with: Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Hoang-Hai Dang, Jan-Oliver Kaiser, David Swasey, Filip Sieczkowski, Kasper Svendsen, Aaron Turon, Amin Timany, and Lars Birkedal

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 morphed from a failed book project into a serious development platform for multiple ongoing efforts in program verification:

- The Rust type system (Jung, Jourdan, Krebbers, Dreyer)
- Logical relations (Krogh-Jespersen, Svendsen, Timany, Birkedal, Krebbers)
- Termination-preserving refinement (Tassarotti, Jung, Harper)
- Weak memory concurrency (Kaiser, Dang, Dreyer, Lahav, Vafeiadis)
- Object capability patterns (Swasey, Garg, Dreyer)
- Logical atomicity (Jung, Swasey, Krogh-Jespersen, Zhang, Dreyer, Birkedal)
- Defining Iris (Krebbers, Jung, Jourdan, Bizjak, Dreyer, Birkedal)

Most of these projects are formalized in Iris in  Coq

Prologue: A brief history of Iris

Pre-Iris [2013]

Pre-Iris [2013]

- Lars Birkedal and I were asked to write a book for Foundations and Trends in PL series on reasoning techniques for higher-order imperative programs. We brought Aaron Turon on board.

Pre-Iris [2013]

- Lars Birkedal and I were asked to write a book for Foundations and Trends in PL series on reasoning techniques for higher-order imperative programs. We brought Aaron Turon on board.
- We set out to write a book on step-indexed Kripke logical relations for ML-like languages, but quickly realized this would be very painful to explain: **Proofs too low-level!**

Pre-Iris [2013]

- Lars Birkedal and I were asked to write a book for Foundations and Trends in PL series on reasoning techniques for higher-order imperative programs. We brought Aaron Turon on board.
- We set out to write a book on step-indexed Kripke logical relations for ML-like languages, but quickly realized this would be very painful to explain: **Proofs too low-level!**
- Idea: Use a **modal separation logic** (à la Appel's work) to lift the level of abstraction. **But which one?**

Pre-Iris [2013]

- Lars Birkedal and I were asked to write a book for Foundations and Trends in PL series on reasoning techniques for higher-order imperative programs. We brought Aaron Turon on board.
- We set out to write a book on step-indexed Kripke logical relations for ML-like languages, but quickly realized this would be very painful to explain: **Proofs too low-level!**
- Idea: Use a **modal separation logic** (à la Appel's work) to lift the level of abstraction. **But which one?**
- We had just developed a logic called CaReSL [ICFP'13] for verifying contextual refinement for fine-grained concurrent data structures, but it was somewhat ad-hoc and didn't feel canonical enough for a pedagogical text.

Iris 1.0 [POPL'15]

Iris 1.0 [POPL'15]

- Aaron has a bright idea – Simplify foundations of advanced concurrent separation logics (CSLs) down to 2 mechanisms:
  - **Partial commutative monoids (PCMs)**
    for representing user-defined ghost state
  - **Invariants** for tying ghost state to physical state
  - Fancier mechanisms should be derivable in terms of these two!

Iris 1.0 [POPL'15]

- Aaron has a bright idea – Simplify foundations of advanced concurrent separation logics (CSLs) down to 2 mechanisms:
  - **Partial commutative monoids (PCMs)**
    for representing user-defined ghost state
  - **Invariants** for tying ghost state to physical state
  - Fancier mechanisms should be derivable in terms of these two!

- Around the end of 2013, I gave this idea to my new student, **Ralf Jung**, to play with. He picked it up and ran with it.
  - Developed encodings of sophisticated proof rules from various advanced CSLs in terms of PCMs and invariants
  - Developed encoding of "logical atomicity", inspired by TaDA logic (da Rocha Pinto et al., ECOOP'14), but going beyond it

Iris 1.0 [POPL'15]

- Aaron has a bright idea – Simplify foundations of advanced concurrent separation logics (CSLs) down to 2 mechanisms:
  - **Partial commutative monoids (PCMs)**
    for representing user-defined ghost state
  - **Invariants** for tying ghost state to physical state
  - Fancier mechanisms should be derivable in terms of these two!

- Around the end of 2013, I gave this idea to my new student, **Ralf Jung**, to play with. He picked it up and ran with it.
  - Developed encodings of sophisticated proof rules from various advanced CSLs in terms of PCMs and invariants
  - Developed encoding of "logical atomicity", inspired by TaDA logic (da Rocha Pinto et al., ECOOP'14), but going beyond it

Iris is born!

Some obvious questions you might have about Iris 1.0

1. Are PCMs-and-invariants the "right" foundation?
   - Are there things it can't express?
   - Is it as simple and canonical as possible?

Some obvious questions you might have about Iris 1.0

1. Are PCMs-and-invariants the "right" foundation?
   - Are there things it can't express?
   - Is it as simple and canonical as possible?

2. We have developed yet another concurrent separation logic. Yawn.
   - How can we get "people" to use this thing?

Some obvious questions you might have about Iris 1.0

1. Are PCMs-and-invariants the "right" foundation?
   - Are there things it can't express? **Yes.**
   - Is it as simple and canonical as possible? **No.**

2. We have developed yet another concurrent separation logic. Yawn.
   - How can we get "people" to use this thing?

     Coq support!

Iris 2.0, Iris 3.0, and Iris proof mode

Iris 2.0, Iris 3.0, and Iris proof mode

Iris 2.0 [ICFP'16]: Making Iris more **expressive**

- Extends Iris 1.0 with "higher-order ghost state", i.e., ghost state that can store (Iris) propositions
- Generalizes PCMs to "cameras" (step-indexed PCMs)

Iris 2.0, Iris 3.0, and Iris proof mode

Iris 2.0 [ICFP'16]: Making Iris more **expressive**

- Extends Iris 1.0 with "higher-order ghost state", i.e., ghost state that can store (Iris) propositions
- Generalizes PCMs to "cameras" (step-indexed PCMs)

Iris 3.0 [ESOP'17]: Making Iris more **canonical**

- Distills essence of Iris to a bare-bones modal base logic, which knows nothing about programs
- Entire program specification layer of Iris (e.g., Hoare triples) is *encoded* on top of the base logic

Iris 2.0, Iris 3.0, and Iris proof mode

Iris 2.0 [ICFP'16]: Making Iris more **expressive**

- Extends Iris 1.0 with "higher-order ghost state", i.e., ghost state that can store (Iris) propositions
- Generalizes PCMs to "cameras" (step-indexed PCMs)

Iris 3.0 [ESOP'17]: Making Iris more **canonical**

- Distills essence of Iris to a bare-bones modal base logic, which knows nothing about programs
- Entire program specification layer of Iris (e.g., Hoare triples) is *encoded* on top of the base logic

Iris proof mode [POPL'17]: Making Iris more **usable**

- Builds tactical support for not only proving soundness of Iris in Coq, but doing **proofs of programs in Iris in Coq**
- Fundamental catalyst for all the ongoing applications of Iris!

Preview of the rules of the Iris base logic

Laws of (affine) bunched implications

$$\begin{array}{c} \mathsf{True} * P \dashv\vdash P \\ P * Q \vdash Q * P \\ (P * Q) * R \vdash P * (Q * R) \end{array} \qquad \frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * P_2 \vdash Q_1 * Q_2} \qquad \frac{P * Q \vdash R}{P \vdash Q \mathrel{-\!\!*} R} \qquad \frac{P \vdash Q \mathrel{-\!\!*} R}{P * Q \vdash R}$$

Laws for resources and validity

$$\mathsf{Own}(a) * \mathsf{Own}(b) \dashv\vdash \mathsf{Own}(a \cdot b) \qquad \mathsf{True} \vdash \mathsf{Own}(\varepsilon) \qquad \mathsf{Own}(a) \vdash \Box\, \mathsf{Own}(|a|)$$
$$\mathsf{Own}(a) \vdash \mathcal{V}(a) \qquad \mathcal{V}(a \cdot b) \vdash \mathcal{V}(a) \qquad \mathcal{V}(a) \vdash \Box\, \mathcal{V}(a)$$

Laws for the basic update modality

$$\frac{P \vdash Q}{\Rrightarrow P \vdash \Rrightarrow Q} \qquad P \vdash \Rrightarrow P \qquad \Rrightarrow \Rrightarrow P \vdash \Rrightarrow P$$

$$Q * \Rrightarrow P \vdash \Rrightarrow (Q * P) \qquad \frac{a \rightsquigarrow B}{\mathsf{Own}(a) \vdash \Rrightarrow \exists b \in B.\, \mathsf{Own}(b)}$$

Laws for the always modality

$$\frac{P \vdash Q}{\Box P \vdash \Box Q} \qquad \Box P \vdash P \qquad \begin{array}{c} \mathsf{True} \vdash \Box\, \mathsf{True} \\ \Box (P \wedge Q) \vdash \Box (P * Q) \\ \Box P \wedge Q \vdash \Box P * Q \end{array} \qquad \begin{array}{c} \Box P \vdash \Box \Box P \\ \forall x.\ \Box P \vdash \Box \forall x.\ P \\ \Box \exists x.\ P \vdash \exists x.\ \Box P \end{array}$$

Laws for the later modality

$$\frac{P \vdash Q}{\triangleright P \vdash \triangleright Q} \qquad (\triangleright P \Rightarrow P) \vdash P \qquad \begin{array}{c} \forall x.\ \triangleright P \vdash \triangleright \forall x.\ P \\ \triangleright \exists x.\ P \vdash \triangleright \mathsf{False} \vee \exists x.\ \triangleright P \end{array} \qquad \begin{array}{c} \triangleright (P * Q) \dashv\vdash \triangleright P * \triangleright Q \\ \Box \triangleright P \dashv\vdash \triangleright \Box P \end{array}$$

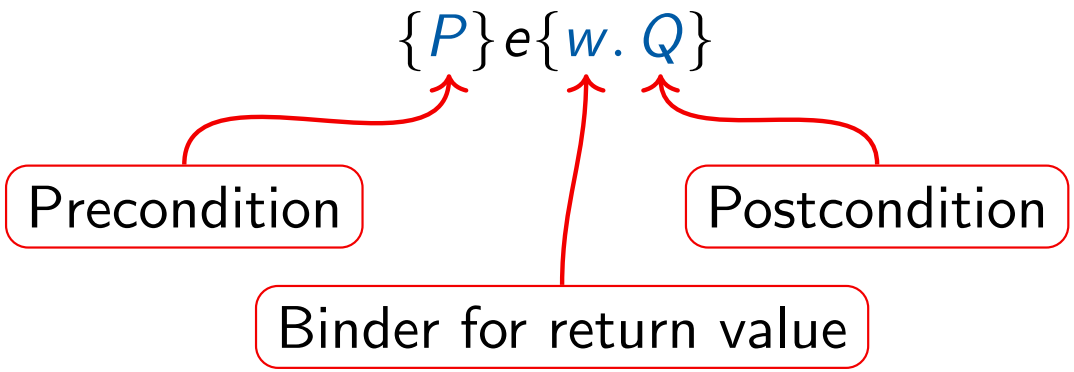
Laws for timeless assertions

$$\triangleright P \vdash \triangleright \mathsf{False} \vee (\triangleright \mathsf{False} \Rightarrow P) \qquad \triangleright \mathsf{Own}(a) \vdash \exists b.\, \mathsf{Own}(b) \wedge \triangleright (a = b)$$

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

Hoare triples

Hoare triples for partial program correctness:



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:

$$\{x \mapsto 4 * y \mapsto 6\}$$

$$x := !\, x + 2 \;\Big\|\; y := !\, y + 2$$

$$\{x \mapsto 6 * y \mapsto 8\}$$

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:

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

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:

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

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, concurrent merge sort, ...

What about shared state/racy programs?

A classic problem:

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

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

$$!\, x$$

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

What about shared state/racy programs?

A classic problem:

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

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

What about shared state/racy programs?

A classic problem:

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

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

What about shared state/racy programs?

A classic problem:

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

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:

$$\frac{\{R * P\}\, e\, \{R * Q\} \qquad e \text{ atomic}}{\boxed{R} \quad \vdash \{P\}\, e\, \{Q\}}$$

Invariants

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

Invariant opening:

$$\frac{\{R * P\}\, e\, \{R * Q\} \qquad e \text{ atomic}}{\boxed{R} \quad \vdash \{P\}\, e\, \{Q\}}$$

Invariant allocation:

$$\frac{\boxed{R} \quad \vdash \{P\}\, e\, \{Q\}}{\{R * P\}\, e\, \{Q\}}$$

Invariants

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

Invariant opening:

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

Invariant allocation:

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

Technical detail: names are needed to avoid *reentrancy*, i.e., 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:

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

Invariant allocation:

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

Technical detail: names are needed to avoid *reentrancy*, i.e., opening the same invariant twice

Other technical detail: the later $\triangleright$ is needed to support impredicative invariants, i.e., $\boxed{\ldots \boxed{R}^{\mathcal{N}_2} \ldots}^{\mathcal{N}_1}$

Invariants in action

Let us consider a simpler problem first:

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

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

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

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

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

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

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

$\{\mathsf{True}\}$	$\Vert$	$\{\mathsf{True}\}$
$\texttt{fetchandadd}(x, 2)$	$\Vert$	$\texttt{fetchandadd}(x, 2)$
$\{\mathsf{True}\}$	$\Vert$	$\{\mathsf{True}\}$

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

$\{\mathsf{True}\}$	$\{\mathsf{True}\}$
$\{x \mapsto n \wedge \mathit{even}(n)\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{x \mapsto n + 2 \wedge \mathit{even}(n+2)\}$	
$\{\mathsf{True}\}$	$\{\mathsf{True}\}$

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

$$
\begin{array}{l}
\{\mathsf{True}\} \\
\quad \{x \mapsto n \wedge \mathit{even}(n)\} \\
\quad \texttt{fetchandadd}(x, 2) \\
\quad \{x \mapsto n + 2 \wedge \mathit{even}(n+2)\} \\
\{\mathsf{True}\}
\end{array}
\quad \| \quad
\begin{array}{l}
\{\mathsf{True}\} \\
\quad \{x \mapsto n \wedge \mathit{even}(n)\} \\
\quad \texttt{fetchandadd}(x, 2) \\
\quad \{x \mapsto n + 2 \wedge \mathit{even}(n+2)\} \\
\{\mathsf{True}\}
\end{array}
$$

$!\, x$

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

Invariants in action

Let us consider a simpler problem first:

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

$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad \texttt{fetchandadd}(x, 2)$
$\quad \{x \mapsto n + 2 \wedge \mathit{even}(n + 2)\}$
$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad !\, x$
$\quad \{n.\, x \mapsto n \wedge \mathit{even}(n)\}$
$\{n.\, \mathit{even}(n)\}$

$\|$

$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad \texttt{fetchandadd}(x, 2)$
$\quad \{x \mapsto n + 2 \wedge \mathit{even}(n + 2)\}$
$\{\mathsf{True}\}$

Invariants in action

Let us consider a simpler problem first:

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

$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad \texttt{fetchandadd}(x, 2)$
$\quad \{x \mapsto n + 2 \wedge \mathit{even}(n + 2)\}$
$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad !\, x$
$\quad \{n.\, x \mapsto n \wedge \mathit{even}(n)\}$
$\{n.\, \mathit{even}(n)\}$

$\|$

$\{\mathsf{True}\}$
$\quad \{x \mapsto n \wedge \mathit{even}(n)\}$
$\quad \texttt{fetchandadd}(x, 2)$
$\quad \{x \mapsto n + 2 \wedge \mathit{even}(n + 2)\}$
$\{\mathsf{True}\}$

Problem: still cannot prove it returns 4

Ghost variables

Consider the invariant:

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

How to relate the quantified value to the state of the threads?

Ghost variables

Consider the invariant:

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

How to relate the quantified value to the state of the threads?

Solution: ghost variables

Ghost variables

Consider the invariant:

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

How to relate the quantified value to the state of the threads?

Solution: ghost variables

Ghost variables are allocated in pairs:

$$\text{True} \quad \Rrightarrow\!\!\!\!\!\!\ast \quad \exists \gamma.\ \underbrace{\gamma \hookrightarrow_\bullet n}_{\text{in the invariant}} \quad * \quad \underbrace{\gamma \hookrightarrow_\circ n}_{\text{in the Hoare triple}}$$

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 relate the quantified value to the state of the threads?

Solution: ghost variables

Ghost variables are allocated in pairs:

$$\text{True} \quad \Rrightarrow \quad \exists \gamma.\ \underbrace{\gamma \hookrightarrow_\bullet n}_{\text{in the invariant}} \quad * \quad \underbrace{\gamma \hookrightarrow_\circ n}_{\text{in the Hoare triple}}$$

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 relate the quantified value to the state of the threads?

Solution: ghost variables

Ghost variables are allocated in pairs:

$$\text{True} \quad \Rrightarrow \quad \exists \gamma.\ \underbrace{\gamma \hookrightarrow_\bullet n}_{\text{in the invariant}} \quad * \quad \underbrace{\gamma \hookrightarrow_\circ n}_{\text{in the Hoare triple}}$$

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

$$\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \quad \Rightarrow \quad n = m$$

$$\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \quad \Rrightarrow \quad \gamma \hookrightarrow_\bullet n' * \gamma \hookrightarrow_\circ n'$$

Ghost variables in action

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

$!\, x$

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

Ghost variables in action

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

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

$\|$

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

$!\, x$

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

Ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$

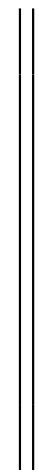
$\texttt{fetchandadd}(x, 2)$ $\quad\|\quad$ $\texttt{fetchandadd}(x, 2)$

$!\, x$

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

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



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

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

$!\, x$

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

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

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

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

$!\, x$

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

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

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

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

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

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

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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

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

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

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

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

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

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

$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1 + n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

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

$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$
$!\, x$

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

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

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

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

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

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

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

$!\, x$

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

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 $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

$\{\gamma_1 \hookrightarrow_\circ 0\}$	$\{\gamma_2 \hookrightarrow_\circ 0\}$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto n_2 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$	$\texttt{fetchandadd}(x, 2)$
$\{\gamma_1 \hookrightarrow_\circ 0 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\bullet n_2\}$	$\{\ldots\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * x \mapsto (2+n_2) * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet n_2\}$	
$\{\gamma_1 \hookrightarrow_\circ 2\}$	$\{\gamma_2 \hookrightarrow_\circ 2\}$

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$
$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$
$!\, x$
$\{n.\, n = 4 \wedge \gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$
$\{n.\, n = 4\}$

Ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\{x \mapsto 0 * \gamma_1 \hookrightarrow_\bullet 0 * \gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\bullet 0 * \gamma_2 \hookrightarrow_\circ 0\}$
allocate $\boxed{\exists n_1, n_2.\, x \mapsto n_1 + n_2 * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2}$
$\{\gamma_1 \hookrightarrow_\circ 0 * \gamma_2 \hookrightarrow_\circ 0\}$

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

$\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2\}$
$\quad\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto (n_1+n_2) * \gamma_1 \hookrightarrow_\bullet n_1 * \gamma_2 \hookrightarrow_\bullet n_2\}$
$\quad\{\gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$
$\quad !\, x$
$\quad\{n.\, n = 4 \wedge \gamma_1 \hookrightarrow_\circ 2 * \gamma_2 \hookrightarrow_\circ 2 * x \mapsto 4 * \gamma_1 \hookrightarrow_\bullet 2 * \gamma_2 \hookrightarrow_\bullet 2\}$
$\{n.\, n = 4\}$

Ghost variables with fractional permissions [Boyland]

What if we have n threads? Using n different ghost variables, results in different proofs for each thread. *That is not modular.*

Better way: ghost variables with a *fractional permission* $(0, 1]_{\mathbb{Q}}$:

$$\gamma \xhookrightarrow{\pi_1+\pi_2}_{\circ} (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \xhookrightarrow{\pi_1}_{\circ} n_1 * \gamma \xhookrightarrow{\pi_2}_{\circ} n_2$$

Ghost variables with fractional permissions [Boyland]

What if we have n threads? Using n different ghost variables, results in different proofs for each thread. *That is not modular.*

Better way: ghost variables with a *fractional permission* $(0, 1]_{\mathbb{Q}}$:

$$\gamma \overset{\pi_1+\pi_2}{\hookrightarrow}_\circ (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \overset{\pi_1}{\hookrightarrow}_\circ n_1 * \gamma \overset{\pi_2}{\hookrightarrow}_\circ n_2$$

You only get the equality when you have *full ownership* ($\pi = 1$):

$$\gamma \hookrightarrow_\bullet n * \gamma \overset{1}{\hookrightarrow}_\circ m \quad \Rightarrow \quad n = m$$

Ghost variables with fractional permissions [Boyland]

What if we have n threads? Using n different ghost variables, results in different proofs for each thread. *That is not modular.*

Better way: ghost variables with a *fractional permission* $(0, 1]_{\mathbb{Q}}$:

$$\gamma \overset{\pi_1+\pi_2}{\hookrightarrow}_\circ (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \overset{\pi_1}{\hookrightarrow}_\circ n_1 * \gamma \overset{\pi_2}{\hookrightarrow}_\circ n_2$$

You only get the equality when you have *full ownership* $(\pi = 1)$:

$$\gamma \hookrightarrow_\bullet n * \gamma \overset{1}{\hookrightarrow}_\circ m \quad \Rightarrow \quad n = m$$

Updating is possible with *partial ownership* $(0 < \pi \leq 1)$:

$$\gamma \hookrightarrow_\bullet n * \gamma \overset{\pi}{\hookrightarrow}_\circ m \quad \Rrightarrow \quad \gamma \hookrightarrow_\bullet (n + i) * \gamma \overset{\pi}{\hookrightarrow}_\circ (m + i)$$

Ghost variables with fractional permissions [Boyland]

What if we have n threads? Using n different ghost variables, results in different proofs for each thread. *That is not modular.*

Better way: ghost variables with a *fractional permission* $(0, 1]_{\mathbb{Q}}$:

$$\gamma \overset{\pi_1+\pi_2}{\hookrightarrow}_\circ (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \overset{\pi_1}{\hookrightarrow}_\circ n_1 * \gamma \overset{\pi_2}{\hookrightarrow}_\circ n_2$$

You only get the equality when you have *full ownership* $(\pi = 1)$:

$$\gamma \hookrightarrow_\bullet n * \gamma \overset{1}{\hookrightarrow}_\circ m \quad \Rightarrow \quad n = m$$

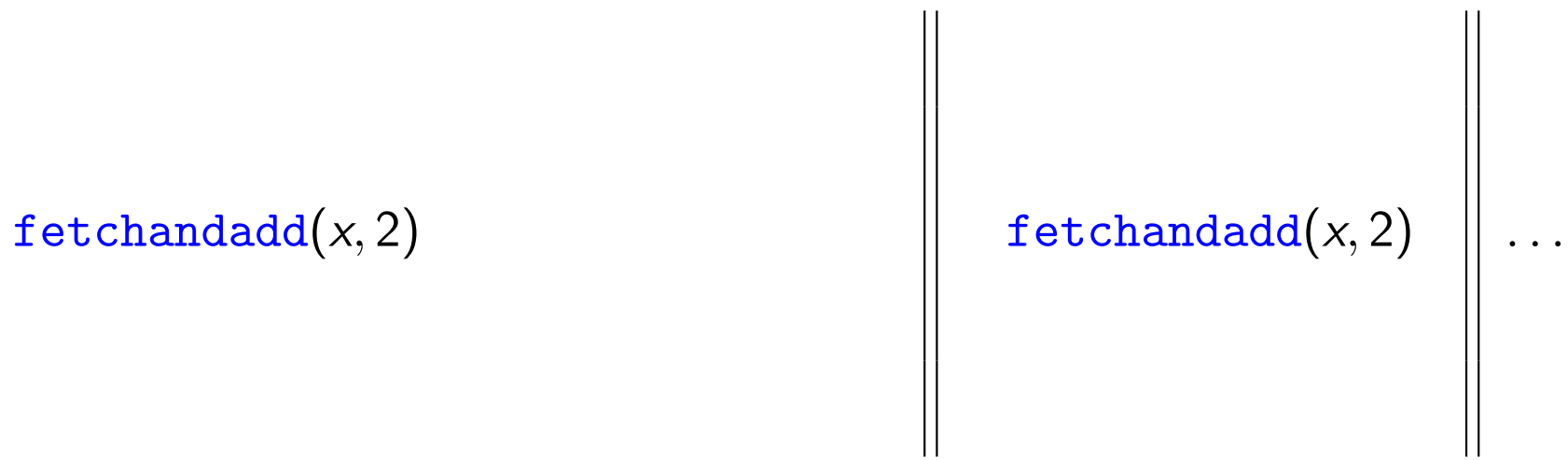
Updating is possible with *partial ownership* $(0 < \pi \leq 1)$:

$$\gamma \hookrightarrow_\bullet n * \gamma \overset{\pi}{\hookrightarrow}_\circ m \quad \Rrightarrow \quad \gamma \hookrightarrow_\bullet (n + i) * \gamma \overset{\pi}{\hookrightarrow}_\circ (m + i)$$

Keeps the invariant that all $\gamma \overset{\pi_i}{\hookrightarrow}_\circ n_i$ sum up to $\gamma \hookrightarrow_\bullet \sum n_i$

Fractional ghost variables in action

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

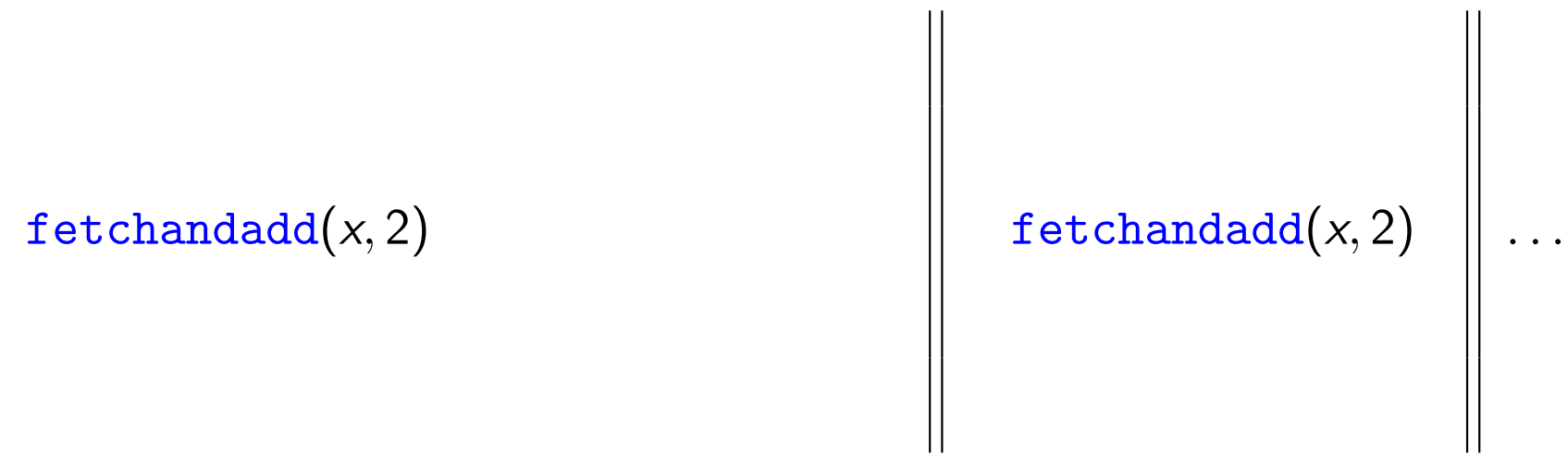


$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

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



$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$

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

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

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

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$ $\quad\|\quad$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$ $\quad\|\quad \ldots$

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

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$ $\quad\|\quad$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$ $\quad\|$

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0 * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\ldots$

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0 * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2 * x \mapsto (2+n) * \gamma_1 \hookrightarrow_\bullet (2+n)\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\parallel$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\parallel \ldots$

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$

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

$\{x \mapsto 0\}$

$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$

allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0 * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$

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

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2 * x \mapsto (2+n) * \gamma_1 \hookrightarrow_\bullet (2+n)\right\}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\Big\|$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$

$\{\ldots\}$

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

$\{\ldots\}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$

$\Big\| \ldots$

$!\, x$

$\{n.\, n = 2k\}$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0 * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2 * x \mapsto (2+n) * \gamma_1 \hookrightarrow_\bullet (2+n)\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$
$\left\{\gamma \overset{1}{\hookrightarrow}_\circ 2k * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$!\, x$
$\{n.\, n = 2k\}$

$\|$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$ $\|$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$ $\|$ $\ldots$
$\|$ $\{\ldots\}$ $\|$
$\|$ $\texttt{fetchandadd}(x, 2)$ $\|$
$\|$ $\{\ldots\}$ $\|$
$\|$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$ $\|$

Fractional ghost variables in action

$\{\mathsf{True}\}$
$\texttt{let}\, x = \texttt{ref}(0)\, \texttt{in}$
$\{x \mapsto 0\}$
$\left\{x \mapsto 0 * \gamma \hookrightarrow_\bullet 0 * \gamma \overset{1}{\hookrightarrow}_\circ 0\right\}$
allocate $\boxed{\exists n.\, x \mapsto n * \gamma \hookrightarrow_\bullet n}$

$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0 * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$\texttt{fetchandadd}(x, 2)$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2 * x \mapsto (2+n) * \gamma_1 \hookrightarrow_\bullet (2+n)\right\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$
$\left\{\gamma \overset{1}{\hookrightarrow}_\circ 2k * x \mapsto n * \gamma \hookrightarrow_\bullet n\right\}$
$!\, x$
$\left\{n.\, n = 2k \wedge \gamma \overset{1}{\hookrightarrow}_\circ 2k * x \mapsto 2k * \gamma \hookrightarrow_\bullet 2k\right\}$
$\{n.\, n = 2k\}$

$\|$ $\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 0\right\}$
$\{\ldots\}$
$\texttt{fetchandadd}(x, 2)$
$\{\ldots\}$
$\left\{\gamma \overset{1/k}{\hookrightarrow}_\circ 2\right\}$ $\|$ $\ldots$

The Iris story, Part 2:
Modeling ghost state via "PCMs"

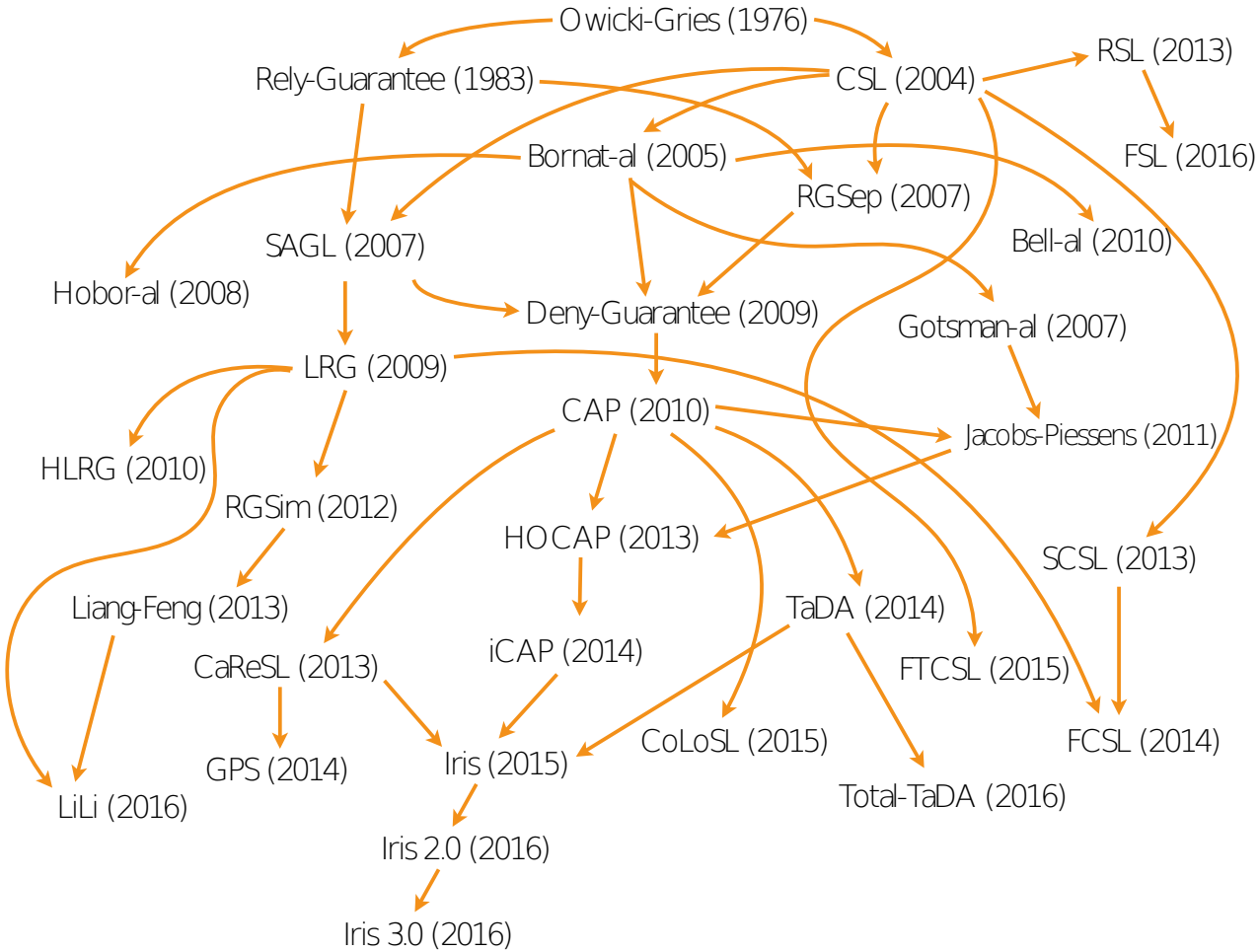
Mechanisms for concurrent reasoning

We have seen so far:

- Invariants $\boxed{R}^{\mathcal{N}}$
- Ghost variables $\gamma \hookrightarrow_\bullet n$ and $\gamma \hookrightarrow_\circ n$
- Fractional ghost variables $\gamma \hookrightarrow_\bullet n$ and $\gamma \overset{\pi}{\hookrightarrow}_\circ n$

Where do these mechanisms come from?

There are many CSLs with more powerful mechanisms...



Picture by Ilya Sergey

. . . and very complicated **primitive** rules

$$\frac{\begin{array}{c}\Gamma,\Delta \mid \Phi \vdash \mathsf{stable}(\mathsf{P}) \qquad \Gamma,\Delta \mid \Phi \vdash \forall y.\ \mathsf{stable}(\mathsf{Q}(y)) \\ \Gamma,\Delta \mid \Phi \vdash n \in C \qquad \Gamma,\Delta \mid \Phi \vdash \forall x \in X.\ (x, f(x)) \in \overline{T(A)} \vee f(x) = x \\ \Gamma \mid \Phi \vdash \forall x \in X.\ (\Delta).\langle \mathsf{P} * \circledast_{\alpha\in A}[\alpha]^{\mathsf{n}}_{g(\alpha)} * \triangleright I(x)\rangle\ c\ \langle \mathsf{Q}(x) * \triangleright I(f(x))\rangle^{C\setminus\{n\}}\end{array}}{\begin{array}{c}\Gamma \mid \Phi \vdash (\Delta).\ \langle \mathsf{P} * \circledast_{\alpha\in A}[\alpha]^{\mathsf{n}}_{g(\alpha)} * \mathsf{region}(X,T,I,n)\rangle \\ c \\ \langle \exists x.\ \mathsf{Q}(x) * \mathsf{region}(\{f(x)\},T,I,n)\rangle^{C}\end{array}}\ \textsc{Atomic}$$

$$\frac{\mathcal{C} \vdash \forall b \overset{\text{rely}}{\sqsupseteq}_\pi b_0.\ (\!|\pi[\![b]\!] * P|\!)\ i \Rightarrow_1 a\ (\!|x.\ \exists b' \overset{\text{guar}}{\sqsupseteq}_\pi b.\ \pi[\![b']\!] * Q|\!)}{\mathcal{C} \vdash \left\{\boxed{b_0}^n_\pi * \triangleright P\right\}\ i \Rightarrow a\ \left\{x.\ \exists b'.\ \boxed{b'}^n_\pi * Q\right\}}\ \textsc{UpdIsl}$$

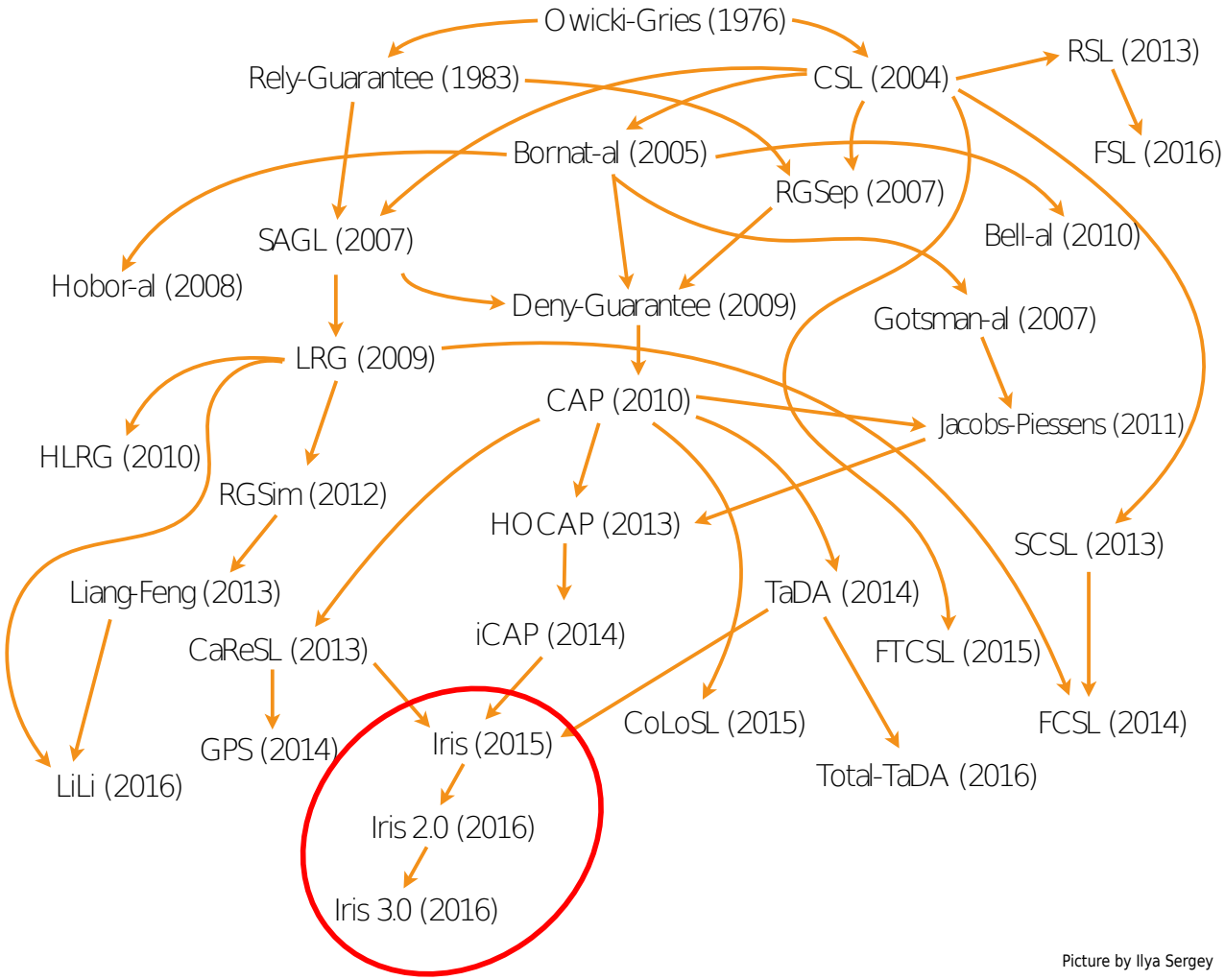
$$\frac{\begin{array}{c}\text{Use atomic rule} \\ a \notin \mathcal{A} \quad \forall x \in X.\ (x, f(x)) \in \mathcal{T}_{\mathbf{t}}(\mathrm{G})^* \\ \lambda; \mathcal{A} \vdash \mathbb{W}x \in X.\ \langle p_p \mid I(\mathbf{t}^\lambda_a(x)) * p(x) * [\mathrm{G}]_a\rangle\ \mathbb{C}\ \ \exists y \in Y.\ \langle q_p(x,y) \mid I(\mathbf{t}^\lambda_a(f(x))) * q(x,y)\rangle\end{array}}{\lambda+1; \mathcal{A} \vdash \mathbb{W}x \in X.\ \langle p_p \mid \mathbf{t}^\lambda_a(x) * p(x) * [\mathrm{G}]_a\rangle\ \mathbb{C}\ \ \exists y \in Y.\ \langle q_p(x,y) \mid \mathbf{t}^\lambda_a(f(x)) * q(x,y)\rangle}$$

$$\frac{\begin{array}{c}\Gamma \mid \Phi \vdash x \in X \qquad \Gamma \mid \Phi \vdash \forall \alpha \in \mathsf{Action}.\ \forall x \in \mathsf{SId} \times \mathsf{SId}.\ up(T(\alpha)(x)) \\ \Gamma \mid \Phi \vdash A \text{ and } B \text{ are finite} \qquad \Gamma \mid \Phi \vdash C \text{ is infinite} \\ \Gamma \mid \Phi \vdash \forall n \in C.\ \mathsf{P} * \circledast_{\alpha\in A}[\alpha]^n_1 \Rightarrow \triangleright I(n)(x) \\ \Gamma \mid \Phi \vdash \forall n \in C.\ \forall s.\ \mathsf{stable}(I(n)(s)) \qquad \Gamma \mid \Phi \vdash A \cap B = \emptyset\end{array}}{\Gamma \mid \Phi \vdash \mathsf{P} \sqsubseteq^C \exists n \in C.\ \mathsf{region}(X,T,I(n),n) * \circledast_{\alpha\in B}[\alpha]^n_1}\ \textsc{VAlloc}$$

$$\frac{\begin{array}{c}\text{Update region rule} \\ \lambda; \mathcal{A} \vdash \mathbb{W}x \in X.\ \left\langle p_p \,\middle|\, I(\mathbf{t}^\lambda_a(x)) * p(x)\right\rangle\ \mathbb{C}\ \ \exists y \in Y.\ \left\langle q_p(x,y) \,\middle|\, \begin{array}{c} I(\mathbf{t}^\lambda_a(Q(x))) * q_1(x,y) \\ \vee\ I(\mathbf{t}^\lambda_a(x)) * q_2(x,y)\end{array}\right\rangle\end{array}}{\begin{array}{l}\mathbb{W}x \in X.\ \langle p_p \mid \mathbf{t}^\lambda_a(x) * p(x) * a \Mapsto \blacklozenge\rangle \\ \mathbb{C} \\ \exists y \in Y.\ \left\langle q_p(x,y) \,\middle|\, \begin{array}{c}\exists z \in Q(x).\ \mathbf{t}^\lambda_a(z) * q_1(x,y) * a \Mapsto (x,z) \\ \vee\ \mathbf{t}^\lambda_a(x) * q_2(x,y) * a \Mapsto \blacklozenge\end{array}\right\rangle\end{array}}$$

$$\lambda{+}1; a : x{\in}X \rightsquigarrow Q(x), \mathcal{A} \vdash$$

The Iris story



The Iris story: many of these mechanisms can be **encoded** using a simple mechanism of *resource ownership*

Generalizing ownership

All forms of ownership have common properties:

- Ownership of different threads can be composed
  For example:

$$\gamma \xhookrightarrow{\pi_1+\pi_2}_{\circ} (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \xhookrightarrow{\pi_1}_{\circ} n_1 * \gamma \xhookrightarrow{\pi_2}_{\circ} n_2$$

Generalizing ownership

All forms of ownership have common properties:

- Ownership of different threads can be composed
  For example:

$$\gamma \overset{\pi_1+\pi_2}{\hookrightarrow}_\circ (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \overset{\pi_1}{\hookrightarrow}_\circ n_1 * \gamma \overset{\pi_2}{\hookrightarrow}_\circ n_2$$

- Composition of ownership is associative and commutative
  Mirroring that parallel composition and separating conjunction is associative and commutative

Generalizing ownership

All forms of ownership have common properties:

- Ownership of different threads can be composed
  For example:

$$\gamma \xhookrightarrow{\pi_1+\pi_2}_{\circ} (n_1 + n_2) \quad \Leftrightarrow \quad \gamma \xhookrightarrow{\pi_1}_{\circ} n_1 * \gamma \xhookrightarrow{\pi_2}_{\circ} n_2$$

- Composition of ownership is associative and commutative
  Mirroring that parallel composition and separating conjunction is associative and commutative
- Combinations of ownership that do not make sense are ruled out
  For example:

$$\gamma \hookrightarrow_{\bullet} 5 * \gamma \xhookrightarrow{1/2}_{\circ} 3 * \gamma \xhookrightarrow{1/2}_{\circ} 4 \quad \Rightarrow \quad \text{False}$$

  (because $5 \neq 3 + 4$)

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 has ghost variables $\boxed{a : M}^{\gamma}$ for each resource algebra M

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

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

Ghost variables revisited

Resource algebra for ghost variables:

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

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

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

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

And define:

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

Ghost variables revisited

Resource algebra for ghost variables:

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

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

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

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

And define:

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

The ghost variable rules follow directly from the general rules:

$$\text{True} \Rrightarrow \exists \gamma.\, \gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ n$$

Ghost variables revisited

Resource algebra for ghost variables:

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

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

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

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

And define:

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

The ghost variable rules follow directly from the general rules:

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

Ghost variables revisited

Resource algebra for ghost variables:

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

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

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

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

And define:

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

The ghost variable rules follow directly from the general rules:

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

$$\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \Rightarrow n = m$$

Ghost variables revisited

Resource algebra for ghost variables:

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

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

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

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

And define:

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

The ghost variable rules follow directly from the general rules:

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

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

Updating resources

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

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

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

Thread 1		Thread 2		...		Thread n	
a_1	$\cdot$	a_2	$\cdot$	$\ldots$	$\cdot$	a_n	$\in \mathcal{V}$
$\rightsquigarrow$							
b_1	$\cdot$	a_2	$\cdot$	$\ldots$	$\cdot$	a_n	$\in \mathcal{V}$

Updating resources

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

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

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

Thread 1		Thread 2		...		Thread n	
a_1	$\cdot$	a_2	$\cdot$	$\ldots$	$\cdot$	a_n	$\in \mathcal{V}$
$\rightsquigarrow$							
b_1	$\cdot$	a_2	$\cdot$	$\ldots$	$\cdot$	a_n	$\in \mathcal{V}$

The rule $\gamma \hookrightarrow_\bullet n * \gamma \hookrightarrow_\circ m \Rrightarrow \gamma \hookrightarrow_\bullet n' * \gamma \hookrightarrow_\circ n'$ follows directly

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.
- $\textsc{Frac}$: The RA for fractions in $(0, 1]$ with addition.
- $\textsc{Excl}(X)$: The "exclusive" RA, whose valid elements are the elements of X, and where composition is always undefined.
- 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: $\textsc{Auth}\,(\textsc{Frac} \times \textsc{Nat}_+)$

Many things I haven't covered

Modal basis of Iris: $\Box$, $\triangleright$, $\Rightarrow\!\!\!\!\mid$

- Persistent modality $\Box P$: Says P holds forever, i.e., only relying on duplicable resources, such as invariants
- Later modality $\triangleright P$: Says P holds one step-index later (lower); needed to model impredicative invariants
- Update modality $\Rightarrow\!\!\!\!\mid P$: Says P holds after some frame-preserving update to ghost state

Higher-order ghost state, e.g., named propositions $\gamma \mapsto P$

- $\gamma \mapsto P * \gamma \mapsto Q \Rightarrow \triangleright(P = Q)$
- Sounds arcane, but turns out to be surprisingly useful!
- Achieved by equipping RAs with a step-indexing structure

Encoding of Iris program logic (including invariants)
within the modal base logic (with higher-order ghost state)

Conclusion

The Iris methodology for concurrent reasoning:

- Divide up logical ownership of shared physical state using appropriately chosen **ghost state predicates and axioms**
- Tie ghost state assertions to physical state using **invariants**
- Build model of ghost state predicates by choosing an appropriate (step-indexed) **"PCM"**
- Verify ghost state axioms as instances of a few basic laws like **frame-preserving update**

Conclusion

The Iris methodology for concurrent reasoning:

- Divide up logical ownership of shared physical state using appropriately chosen **ghost state predicates and axioms**
- Tie ghost state assertions to physical state using **invariants**
- Build model of ghost state predicates by choosing an appropriate (step-indexed) **"PCM"**
- Verify ghost state axioms as instances of a few basic laws like **frame-preserving update**

Yes, but is it "canonical"?

I'll let you be the judge!