Iris: Higher-Order Concurrent Separation Logic

Basic Logic of Resources[1]

Amin Timany

Aarhus University, Denmark

August 13, 2026

[1] Based on slides by Lars Birkedal

Syntax: Types

$$\tau ::= T \mid \mathbb{Z} \mid \mathit{Val} \mid \mathit{Exp} \mid \mathsf{Prop} \mid 1 \mid \tau + \tau \mid \tau \times \tau \mid \tau \to \tau$$

where

- T stands for additional base types which we will add later
- Val and Exp are types of values and expressions in $\lambda_{\mathrm{ref,conc}}$
- Prop is the type of Iris propositions.

Syntax: Terms

$$
\begin{aligned}
t, P ::= {} & x \mid n \mid v \mid e \mid F(t_1, \ldots, t_n) \mid \\
& () \mid (t, t) \mid \pi_i\ t \mid \lambda x : \tau.\ t \mid t(t) \mid \mathsf{inl}\ t \mid \mathsf{inr}\ t \mid \mathsf{case}(t, x.t, y.t) \mid \\
& \mathsf{False} \mid \mathsf{True} \mid t =_\tau t \mid P \Rightarrow P \mid P \wedge P \mid P \vee P \mid \exists x : \tau.\ P \mid \forall x : \tau.\ P \mid \\
& P * P \mid P \mathrel{-\!\!*} P \mid \\
& \square\, P \mid \triangleright P \mid \\
& \{P\}\ t\ \{P\} \mid \\
& t \hookrightarrow t
\end{aligned}
$$

where

- x are variables
- n are integers
- v and e range over values of the language, *i.e.*, they are primitive terms of types *Val* and *Exp*
- F ranges over the function symbols in the signature $\mathcal{S}$.

Well-typed Terms ($\Gamma \vdash_{\mathcal{S}} t : \tau$)

- Typing relation:

$$\Gamma \vdash_{\mathcal{S}} t : \tau$$

  defined inductively by inference rules

- Selected rules:

$$\frac{\Gamma \vdash P : \mathsf{Prop} \qquad \Gamma \vdash Q : \mathsf{Prop}}{\Gamma \vdash P * Q : \mathsf{Prop}}$$

$$\frac{\Gamma \vdash P : \mathsf{Prop} \qquad \Gamma \vdash Q : \mathsf{Prop}}{\Gamma \vdash P \mathrel{-\!\!*} Q : \mathsf{Prop}}$$

Entailment ($\Gamma \mid P \vdash Q$)

- Entailment relation

$$\Gamma \mid P \vdash Q$$

  for $\Gamma \vdash P : \mathrm{Prop}$ and $\Gamma \vdash Q : \mathrm{Prop}$.
- The relation is defined by induction, using standard rules from intuitionistic higher-order logic extended with new rules for the new connectives.
- To understand the entailment rules for the new connectives, we need to have an intuitive understanding of the semantics of the logical connectives.
- Note: in this course, we do not present a formal semantics of the logic and formally prove the logic sound (for that, see "Iris from the Ground Up: A Modular Foundation for Higher-Order Concurrent Separation Logic" on iris-project.org).

Intuition for Iris Propositions

- Intuition: A proposition P describes a set of resources.
- Write $\mathcal{R}$ for the set of resources, and write r_1, r_2, etc., for elements in $\mathcal{R}$.
- We assume that
  - there is an empty resource
  - there is a way to compose (or combine) resources r_1 and r_2, denoted $r_1 \cdot r_2$
  - the composition is defined for resources that are suitably disjoint, denoted $r_1 \# r_2$.
- For now, it suffices to think about the example of $\mathcal{R} = \mathit{Heap}$.

Intuition for Iris Propositions

- We define $\mathit{Heap} = \mathit{Loc} \xrightarrow{\text{fin}} \mathit{Val}$, the set of partial functions from locations to values
- The empty resource is the empty heap, denoted $[]$.
- Two heaps h_1 and h_2 are disjoint, denoted $h_1 \# h_2$, if their domains do not overlap (*i.e.*, $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$).
- The composition of two disjoint heaps h_1 and h_2 is the heap $h = h_1 \cdot h_2$ defined by

$$h(x) = \begin{cases} h_1(x) & \text{if } x \in \mathrm{dom}(h_1) \\ h_2(x) & \text{if } x \in \mathrm{dom}(h_2) \end{cases}$$

Intuition for Iris Propositions

- We said: "A proposition P *describes* a set of resources."
- Also say: "P *is* a set of resources."
- Also say: "P *denotes* a set of resources."
- $P \in \mathcal{P}(\mathcal{R})$.
- When r is a resource described by P, we also say that r *satisfies* P, or that r *is in* P.
- The intuition for $P \vdash Q$ is then that all resources in P are also in Q (*i.e.*, $\forall r \in \mathcal{R}.\, r \in P \Rightarrow r \in Q$).
- NB: this interpretation validates all the rules of IHOL we saw last time.
  - Check this in detail for some of the rules!

Describing Resources in the Logic

- Primitive: the points-to predicate $x \hookrightarrow v$.
- It is a formula, *i.e.*, a term of type Prop

$$\frac{\Gamma \vdash \ell : \mathit{Val} \qquad \Gamma \vdash v : \mathit{Val}}{\Gamma \vdash \ell \hookrightarrow v : \mathrm{Prop}}$$

- It describes the set of heap fragments that map location x to value v

$$x \hookrightarrow v = \{h \mid x \in \mathrm{dom}(h) \wedge h(x) = v\}$$

- Ownership reading: if I assert $\ell \hookrightarrow v$, then I express that I have the ownership of ℓ and hence I may modify what ℓ pointsto, without invalidating invariants of other parts of the program.

Intuition for $*$ and $-\!\!*$

- $P * Q = \{r \mid \exists r_1, r_2. r = r_1 \cdot r_2 \wedge r_1 \in P \wedge r_2 \in Q\}$
- For example, $x \hookrightarrow u * y \hookrightarrow v$ describes the set of heaps with two *disjoint* locations x and y, the first stores u and the second v.
- Note: $x \hookrightarrow v * x \hookrightarrow u \vdash \text{False}$.
- $P -\!\!* Q = \{r \mid \forall r_1. r_1 \# r \wedge r_1 \in P \Rightarrow r \cdot r_1 \in Q\}$
- For example, the proposition

  $$x \hookrightarrow u -\!\!* (x \hookrightarrow u * y \hookrightarrow v)$$

  describes those heap fragments that map y to v, because when we combine it with a heap fragment mapping x to u, then we get a heap fragment mapping x to u and y to v.

Weakening Rule

Weakening rule:

$$\frac{\ast\text{-WEAK}}{P_1 \ast P_2 \vdash P_1}$$

- Thus Iris is an affine separation logic.
- Example:

$$x \hookrightarrow u \ast y \hookrightarrow v \vdash x \hookrightarrow u$$

  - Suppose $h \in (x \hookrightarrow u \ast y \hookrightarrow v)$.
  - Then $h(x) = u$ and $h(y) = v$.
  - Therefore $h \in (x \hookrightarrow u)$.
  - Generally, if $h \in P$ and $h' \geq h$, then also $h' \in P$.

Weakening Rule

In a bit more detail:

- Intuitively, the fact that this rule is sound means that propositions are interpreted by upwards closed sets of resources:
  - We say that $r_1 \geq r_2$ iff $r_1 = r_2 \cdot r_3$, for some r_3.
  - Suppose $r_1 \in P_1$ and that $r \geq r_1$. Then there is r_2 such that $r = r_1 \cdot r_2$.
  - Let P_2 be $\{r_2\}$.
  - Then $r_1 \cdot r_2 \in P_1 * P_2$.
  - By the weakening rule, we then also have that $r = r_1 \cdot r_2 \in P_1$.
  - Hence P_1 is upwards closed.
- The above is not a formal proof, hence the stress on "intuitively".

Associativity and Commutativity of $*$

Basic structural rules:

$$\frac{}{P_1 * (P_2 * P_3) \dashv\vdash (P_1 * P_2) * P_3}\ *\text{-}\mathrm{ASSOC} \qquad \frac{}{P_1 * P_2 \dashv\vdash P_2 * P_1}\ *\text{-}\mathrm{COMM}$$

Sound because composition of resources, $\cdot$, is commutative and associative.

Separating Conjunction Introduction

$$\frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * P_2 \vdash Q_1 * Q_2}\ *\mathrm{I}$$

- To show a separating conjuction $Q_1 * Q_2$, we need to split the assumption and decide which resources to use to prove Q_1 and which ones to use to prove Q_2.
- Example: $P \vdash P * P$ is not provable in general

Magic wand introduction and elimination

$$\frac{R * P \vdash Q}{R \vdash P \mathrel{-\!\!*} Q}\ {-\!\!*}\mathrm{I} \qquad \frac{R_1 \vdash P \mathrel{-\!\!*} Q \qquad R_2 \vdash P}{R_1 * R_2 \vdash Q}\ {-\!\!*}\mathrm{E}$$

- Introduction rule intuitively sound because
  - Suppose $r \in R$. TS $r \in P \mathrel{-\!\!*} Q$.
  - Thus let $r_1 \in P$ and suppose $r_1 \# r$. TS $r \cdot r_1 \in Q$.
  - We have $r \cdot r_1 \in R * P$.
  - Hence, by antecedent, $r \cdot r_1 \in Q$, as required.
- Elimination rule intuitively sound because
  - ...

Iris: Higher-Order Concurrent Separation Logic

Basic Separation Logic: Hoare Triples[2]

Amin Timany

Aarhus University, Denmark

August 13, 2026

[2] Based on slides by Lars Birkedal

Hoare Triples

$$\frac{\Gamma \vdash P : \mathsf{Prop} \qquad \Gamma \vdash e : \mathit{Exp} \qquad \Gamma \vdash \Phi : \mathit{Val} \to \mathsf{Prop}}{\Gamma \vdash \{P\}\, e\, \{\Phi\} : \mathsf{Prop}}$$

Intuition

- $\{P\}\, e\, \{\Phi\}$ holds if, when we run the program e in a heap h satisfying P, then the computation does not get stuck and, moreover, if it terminates with a value v and a heap h', then h' satisfies $\Phi(v)$.
- Φ has two purposes: describes the value v (*e.g.*, $v = 3$) and the resources after execution (*e.g.*, $x \hookrightarrow 15$).
- Note that Φ is a function – we often write Φ as $v.\, Q$ instead of $\lambda v.\, Q$.

Examples

$$\{l \hookrightarrow 5\}\; l \leftarrow !\, l + 1\; \{v.v = () \wedge l \hookrightarrow 6\}$$

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2\}\; \mathsf{swap}\; \ell_1\, \ell_2\; \{v.v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1\}.$$

Hoare Triples

More intuition

- Precondition P describes the resources necessary to run e safely (recall "does not get stuck" in the intuitive reading above).
- In our operational semantics, memory errors, *e.g.*, trying to dereference a location that has not been allocated, are modelled by the computation getting stuck.
- So if e satisfies a Hoare triple, then its computation will not lead to any memory errors.
- Precondition P describes the resources needed for e to run safely: we sometimes say that P includes the *footprint* of e.
- (Later on, not all resources needed to execute e will need to be in the precondition — resources shared among different threads will be in invariants, and only resources owned by e's thread will in the precondition.)

Frame Rule

$$\frac{S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P * R\}\, e\, \{v.Q * R\}}\ \textsc{Ht-frame}$$

- Intuitively sound because of the footprint reading of triples
- Note that the frame R is maintained unchanged from precondition to postcondition.
- We do not have to explicitly say that e does not modify other resources not in its precondition!
- Very important!
- We will use this rule all the time.

Frame Rule

Example

- Consider the specification for swap:

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2\}\, \textsf{swap}\ \ell_1\, \ell_2\, \{v. v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1\}.$$

- What if we want to apply this function somewhere, where we have more resources around ? For instance $\ell_3 \hookrightarrow 3$. Then we use the frame rule, with frame $R = \ell_3 \hookrightarrow 3$, to derive

$$\{\ell_1 \hookrightarrow v_1 * \ell_2 \hookrightarrow v_2 * \ell_3 \hookrightarrow 3\}\, \textsf{swap}\ \ell_1\, \ell_2\, \{v. v = () \wedge \ell_1 \hookrightarrow v_2 * \ell_2 \hookrightarrow v_1 * \ell_3 \hookrightarrow 3\}.$$

Value Rule

$$\dfrac{\textsc{Ht-ret} \qquad w \text{ is a value}}{S \vdash \{\mathsf{True}\}\, w\, \{v.v = w\}}$$

Bind Rule

To verify larger expressions we use the $\textsc{Ht-bind}$ rule:

$$\frac{\begin{array}{l}\textsc{Ht-bind}\\ E \text{ is an eval. context} \qquad S \vdash \{P\}\, e\, \{v.\, Q\} \qquad S \vdash \forall v.\, \{Q\}\, E[v]\, \{w.\, R\}\end{array}}{S \vdash \{P\}\, E[e]\, \{w.\, R\}}$$

- Exercise: Use $\textsc{Ht-bind}$ to show $\{\mathrm{True}\}\, 3 + 4 + 5\, \{v.v = 12\}$.

Persistent Propositions

- Intuition: persistent propositions are propositions that do not rely on resources, *i.e.*, either they hold for all resources or none.

$$P \wedge Q \vdash P * Q \qquad \text{if } P \text{ is persistent.}$$

- Persistent propositions may be moved in and out of preconditions:

$$\textsc{Ht-Eq}\ \frac{S \wedge t =_\tau t' \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P \wedge t =_\tau t'\}\, e\, \{v.Q\}} \qquad \textsc{Ht-Ht}\ \frac{S \wedge \{P_1\}\, e_1\, \{v.Q_1\} \vdash \{P_2\}\, e_2\, \{v.Q_2\}}{S \vdash \{P_2 \wedge \{P_1\}\, e_1\, \{v.Q_1\}\}\, e_2\, \{v.Q_2\}}$$

- For now it suffices to know that persistence is preserved by $\forall$ and $\wedge$ — we will see a general treatment later.

Example of HT-HT

$$\frac{\{x \hookrightarrow 5\}\, \mathsf{inc}\, x\, \{v. v = () \wedge x \hookrightarrow 6\} \vdash \{x \hookrightarrow 5\}\, \mathsf{inc}\, x\, \{v. v = () \wedge x \hookrightarrow 6\}}{\{x \hookrightarrow 5 \wedge \{x \hookrightarrow 5\}\, \mathsf{inc}\, x\, \{v. v = () \wedge x \hookrightarrow 6\}\}\, \mathsf{inc}\, x\, \{v. v = () \wedge x \hookrightarrow 6\}}$$

Consequence Rule

$$\textsc{Ht-csq}\ \frac{S \text{ persistent} \qquad S \vdash P \Rightarrow P' \qquad S \vdash \{P'\}\, e\, \{v.\, Q'\} \qquad S \vdash \forall u.\, Q'[u/v] \Rightarrow Q[u/v]}{S \vdash \{P\}\, e\, \{v.\, Q\}}$$

Remark: S is usually a conjunction of equalities and universally quantified Hoare triples, so is usually persistent.

Load Rule

$$\frac{\textsc{Ht-load}}{S \vdash \{\ell \hookrightarrow u\} \; !\,\ell \; \{v. v = u \land \ell \hookrightarrow u\}}$$

- Intuitively sound because ...

Alloc Rule

$$\frac{\textsc{Ht-alloc}}{S \vdash \{\mathsf{True}\}\ \mathsf{ref}\,(u)\ \{v.\exists \ell.\ v = \ell \wedge \ell \hookrightarrow u\}}$$

- Intuitively sound because ...

Store Rule

$$\frac{\textsc{Ht-store}}{S \vdash \{\ell \hookrightarrow -\}\, \ell \leftarrow w \,\{v.v = () \wedge \ell \hookrightarrow w\}}$$

- $\ell \hookrightarrow -$ shorthand for $\exists u.\, \ell \hookrightarrow u$
- Intuitively sound because ...

The Rule for Conditionals

$$\textsc{Ht-If}\quad \frac{\{P * v = \mathsf{true}\}\; e_2\; \{u.Q\} \qquad \{P * v = \mathsf{false}\}\; e_3\; \{u.Q\}}{\{P\}\ \mathsf{if}\ v\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3\ \{u.Q\}}$$

Recursion Rule

$$\frac{\Gamma, f : \mathit{Val} \mid S \wedge \forall y.\, \forall v.\, \{P\}\, f\ v\, \{u.Q\} \vdash \forall y.\, \forall v.\, \{P\}\, e[v/x]\, \{u.Q\}}{\Gamma \mid S \vdash \forall y.\, \forall v.\, \{P\}\, (\mathsf{rec}\, f\ \ x := e) v\, \{u.Q\}}\ \textsc{Ht-Rec}$$

- Here y is a "logical" variable, which may be used in P and Q to relate pre and postconditions. Example:
  - $\forall y : \mathbb{N}. \forall x. \{x = y\}\, \mathsf{double}\, x\, \{v.\ v =_{\mathit{Val}} 2 \times y\}$
- When reasoning about the body, we get to assume that f satisfies the triple we are about to prove.
- Intuitively sound by induction on reduction steps.

Factorial

Implementation

- $\mathsf{rec}\ \mathsf{fac}\ \ n := \mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n * \mathsf{fac}(n-1)$

Specification

- $\forall n : \mathbb{N}. \{\mathsf{True}\}\ \mathsf{fac}\ n\ \{v.\ v =_{Val} n!\}$

Proof

- Use the recursion rule!