Iris: Higher-Order Concurrent Separation Logic

The Later and Persistently Modalities[1]

Amin Timany

Aarhus University, Denmark

August 13, 2026

[1] Based on slides by Lars Birkedal

Later Modality

- Recall the recursion rule:

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

- This rule involves a kind of recursive reasoning.
- We mentioned earlier that this rule is *sound* because function application involves reduction steps, *i.e.,* we will only use the recursive assumption after some reduction steps have taken place.
- Later in the course, when discussing *invariants*, we will want to have other forms of recursive reasoning, where the recursive reasoning steps are not directly tied to corresponding reduction steps.
- But soundness will still hinge on some reduction steps taking place.
- Thus to ensure soundness, we will need some way to express, in the logic, that a property is only supposed to hold *later*, after a reduction step has taken place.
- This is what the later modality $\triangleright$ achieves: intuitively, $\triangleright P$ holds if P holds after a reduction step has been taken.

Löb Rule

- Typing for $\triangleright$:

$$\frac{\Gamma \vdash P : \mathsf{Prop}}{\Gamma \vdash \triangleright P : \mathsf{Prop}}$$

- Löb Rule:

$$\frac{\textsc{Löb} \quad S \wedge \triangleright P \vdash P}{S \vdash P}$$

- Akin to a coinduction proof rule: to show P, it suffices to show P *under the assumption that* P *holds later*.

Aside: semantics of propositions

- As suggested by the above, the meaning of Iris propositions is not just a set of resources.
- In more detail, an Iris proposition P is[2] a set of pairs (k, r), with k a natural number and r a resource.
- Think of k as a step-index, a natural number which expresses for how many reduction steps we know that r is in P.
- If $(k, r) \in P$ and $m \leq k$, then also $(m, r) \in P$.
- The step-indices are used to interpret $\triangleright$:

$$\triangleright P = \{(m+1, r) \mid (m, r) \in P\} \cup \{(0, r) \mid r \in \mathcal{R}\}$$

- "later" means that the index number is smaller (there are fewer reduction steps left, after we have taken some reduction steps).
- The Löb Rule is proved sound by induction on these step-indices.

[2] Not really, but closer to being...

Laws for Later Modality

$$\textsc{Later-Mono}\ \frac{Q \vdash P}{\triangleright Q \vdash \triangleright P} \qquad \textsc{Later-weak}\ \frac{Q \vdash P}{Q \vdash \triangleright P} \qquad \textsc{Löb}\ \frac{Q \wedge \triangleright P \vdash P}{Q \vdash P}$$

Laws for Later Modality

$$\textsc{Later-conj}\ \frac{R \vdash \triangleright(P \wedge Q)}{R \vdash \triangleright P \wedge \triangleright Q} \qquad \textsc{Later-disj}\ \frac{R \vdash \triangleright(P \vee Q)}{R \vdash \triangleright P \vee \triangleright Q} \qquad \textsc{Later-all}\ \frac{Q \vdash \triangleright \forall x.\, P}{Q \vdash \forall x.\, \triangleright P} \qquad \textsc{Later-sep}\ \frac{R \vdash \triangleright P * \triangleright Q}{R \vdash \triangleright(P * Q)}$$

$$\triangleright\text{-}\exists\ \frac{\tau \text{ is inhabited} \qquad Q \vdash \triangleright \exists x : \tau.\, P}{Q \vdash \exists x : \tau.\, \triangleright P} \qquad \triangleright\text{-}\exists\ \frac{Q \vdash \exists x.\, \triangleright P}{Q \vdash \triangleright \exists x.\, P}$$

Stronger rules for Hoare triples

HT-BETA

$$\frac{S \vdash \{P\}\ e\,[v/x]\ \{u.Q\}}{S \vdash \{\triangleright P\}\ (\lambda x.e)v\ \{u.Q\}}$$

HT-REC

$$\frac{Q \vdash \{P\}\ e\,[(\mathsf{rec}\ f\ x := e)/f, v/x]\ \{\Phi\}}{Q \vdash \{\triangleright P\}\ (\mathsf{rec}\ f\ x := e)v\ \{\Phi\}}$$

HT-LOAD

$$\frac{}{S \vdash \{\triangleright \ell \hookrightarrow u\}\ !\ell\ \{v.v = u \wedge \ell \hookrightarrow u\}}$$

HT-STORE

$$\frac{}{S \vdash \{\triangleright \exists u.\, \ell \hookrightarrow u\}\ \ell \leftarrow w\ \{v.v = () \wedge \ell \hookrightarrow w\}}$$

HT-MATCH

$$\frac{S \vdash \{P\}\ e_i\,[u/x_i]\ \{v.Q\}}{S \vdash \{\triangleright P\}\ \mathsf{match}\ \mathsf{inj}_i\ u\ \mathsf{with}\ \mathsf{inj}_1\ x_1 \Rightarrow e_1 \mid \mathsf{inj}_2\ x_2 \Rightarrow e_2\ \mathsf{end}\ \{v.Q\}}$$

Remark on soundness

Why are the rules HT-LOAD and HT-STORE sound ?

- Difficult to explain intuitively.
- Relies on $\hookrightarrow$ being a timeless predicate together with the definition of Hoare triples (the fact that weakest precondition is "closed wrt. the fancy update modality").

Stronger derived Hoare triples

$$\textsc{Ht-seq}\quad \frac{S \vdash \{P\}\ e_1\ \{\_.\ \triangleright Q\} \qquad S \vdash \{Q\}\ e_2\ \{u.R\}}{S \vdash \{P\}\ e_1; e_2\ \{u.R\}}$$

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

Guarded recursively defined predicates

- We extend terms of the logic with

$$t ::= \cdots \mid \mu x : \tau.\, t$$

  with the side-condition that the recursive occurrences must be *guarded*: in $\mu x.\, t$, the variable x can only appear under the later $\triangleright$ modality.

- Fixed-point property expressed by the following rule:

$$\textsc{Mu-fixed}\quad \frac{}{Q \vdash \mu x : \tau.\, t =_\tau t\,[\mu x : \tau.\, t/x]}$$

- You may think of the existence of the fixed point as an application of Banach's fixed point theorem.

Guarded recursively defined predicates

- Example: using a stream (infinite list) as model of linked list:

$$\mu\,\mathsf{isStream} : \mathit{Val} \to \mathrm{stream}\ \mathit{Val} \to \mathsf{Prop}.\ \lambda l : \mathit{Val}.\ \lambda xs : \mathrm{stream}\ \mathit{Val}.$$
$$(xs = [] \land l = \mathsf{inj}_1())\lor$$
$$(\exists x, xs'.\ xs = x : xs' \land \exists hd, l'.\ l = \mathsf{inj}_2(hd) * hd \hookrightarrow (x, l') * \triangleright(\mathsf{isStream}\ l'\,xs'))$$

- Note that xs is a stream (infinite list). Therefore we cannot define the predicate by induction on xs.
- Above, the recursion variable occurs positively.
- In Iris, Hoare triples are defined in terms of weakest-preconditions, which are defined by means of a guarded recursive definition for a positive definition to give a partial correctness interpretation.
- One can also define mixed-variance recursive predicates.

Guarded recursively defined predicates

- Mixed-variance guarded recursive predicates are useful for
  - Interpreting recursive types in a typed programming language by Iris predicates / relations (see the paper A Logical Approach to Type Soundness).
  - Defining models of untyped / unityped languages (*e.g.*, for object capabilities).
  - Specifying and reasoning about libraries that can call themself recursively, *e.g.*, an event loop library (see iCap-paper).

Example: Fixed-point combinator Θ_F

- Given a value F, the call-by-value Turing fixed-point combinator Θ_F is:

$$\Omega_F = \lambda r.F(\lambda x.rrx)$$
$$\Theta_F = \Omega_F \Omega_F$$

- For any values F and v,

$$\Theta_F v \rightsquigarrow F(\lambda x.\Theta_F x)v$$

- Thus, if $F = \lambda fx.e$ then one should think of Θ_F as rec $f\ x := e$.

Proof Rule for Θ_F

- Now we wish to derive proof rule for Θ_F, similar to the recursion rule.

$$\frac{\Gamma \mid S \land \forall v.\, \{P\}\, \Theta_F v\, \{u.Q\} \vdash \forall v.\, \{P\}\, F(\lambda x.\Theta_F x) v\, \{u.Q\}}{\Gamma \mid S \vdash \forall v.\, \{P\}\, \Theta_F v\, \{u.Q\}}\ \textsc{Ht-Turing-fp}$$

- We will use the Löb rule.
- We will also use that if P is persistent, then $\triangleright P$ is persistent, which means that it can be moved in and out of preconditions.

Proof

- We proceed by the Löb rule and hence we assume

$$\triangleright \forall v.\, \{P\}\, \Theta_F\, v\, \{u.Q\} \qquad (*)$$

- and we are to show

$$\forall v.\, \{P\}\, \Theta_F\, v\, \{u.Q\}.$$

- Let v be a value.
- By Later-weak and the rule of consequence SFTS

$$\{\triangleright P\}\, \Theta_F\, v\, \{u.Q\}.$$

Proof

- Since Hoare triples are persistent, we can move our assumption $(*)$ into the precondition, and thus SFTS:

$$\{\triangleright(\forall v.\, \{P\}\, \Theta_F v\, \{u.Q\} \wedge P)\}\, \Theta_F v\, \{u.Q\}$$

- By the bind rule and the stronger rule HT-BETA introduced above SFTS

$$\{\forall v.\, \{P\}\, \Theta_F v\, \{u.Q\} \wedge P\}\, F(\lambda x.\Theta_F x) v\, \{u.Q\}$$

- We again use persistence and move the triple $\forall v.\, \{P\}\, \Theta_F v\, \{u.Q\}$ into the context and then SFTS

$$\{P\}\, F(\lambda x.\Theta_F x) v\, \{u.Q\}$$

- But this is exactly the premise of the rule HT-TURING-FP, and thus the proof is concluded.

Persistently Modality $\square$

- Earlier: explained that Hoare triples and equality predicates are *persistent* and hence may be moved in-and-out of preconditions.
- It is possible to give a systematic definition and treatment of persistent predicates, see the chapter in the lecture notes.
- Informally, persistent predicates are those predicates that do not assert exclusive ownership over resources.
- Hence they only express "knowledge", not exclusive ownership.
- Hence persistent predicates P are duplicable: $P \dashv\vdash P * P$.
- Duplicable predicates are important because we can always make a copy of them to give away to other threads.
- (Why not just a "duplicable" modality? Not so easy... See Bizjak and Birkedal: On Models of Higher-Order Separation Logic for an abstract detailed study.)

Persistently Modality $\square$

- Typing for $\square$:

$$\frac{\Gamma \vdash P : \mathsf{Prop}}{\Gamma \vdash \square P : \mathsf{Prop}}$$

- Definition: we call a proposition P *persistent* if it satisfies $P \vdash \square P$.

Persistently Modality $\square$

- Typing for $\square$:

$$\frac{\Gamma \vdash P : \text{Prop}}{\Gamma \vdash \square P : \text{Prop}}$$

- Definition: we call a proposition P *persistent* if it satisfies $P \vdash \square P$.
- Persistently modality aka Always modality

Intuitive Semantics of $\Box$ Modality

- Intuitive semantics:

$$\Box P = \{r \in \mathcal{R} \mid \exists s, r'. s \in P \land s = s \cdot s \land r = s \cdot r'\}$$

- You might think of this as "$\Box P$ is the upwards-closure of the set of duplicable resources in P" (recall that Iris propositions are upwards-closed wrt. resources, hence the upwards-closure).
- Example: If $\mathcal{R} = \mathit{Heap}$, then
  - the only duplicable resource is the empty heap,
  - hence, $\Box P$ is either the the set of all heaps (True), if the empty heap is in P, or the empty set (False), if the empty heap is not in P.

Laws for $\Box$

The following laws are immediate from the intuitive reading of $\Box P$:

$$\textsc{Persistently-mono}\ \frac{P \vdash Q}{\Box P \vdash \Box Q} \qquad \textsc{Persistently-E}\ \frac{}{\Box P \vdash P} \qquad \textsc{Persistently-idemp}\ \frac{}{\Box P \vdash \Box \Box P}$$

Using these we can derive:

$$\textsc{Persistently-intro}\ \frac{\Box P \vdash Q}{\Box P \vdash \Box Q}$$

Laws for $\Box$

The following laws are immediate from the intuitive reading of $\Box P$:

$$\textsc{Persistently-mono}\ \frac{P \vdash Q}{\Box P \vdash \Box Q} \qquad \textsc{Persistently-E}\ \frac{}{\Box P \vdash P} \qquad \textsc{Persistently-idemp}\ \frac{}{\Box P \vdash \Box \Box P}$$

Using these we can derive:

$$\textsc{Persistently-intro}\ \frac{\Box P \vdash Q}{\Box P \vdash \Box Q}$$

- Assume $\Box P \vdash Q$. By $\textsc{Persistently-mono}$, $\Box \Box P \vdash \Box Q$, and by $\textsc{Persistently-E}$, $\Box P \vdash \Box \Box P$, so done by transitivity.

Laws for $\Box$

$\Box$ commutes with many of the ordinary logical connectives (note: not $\Rightarrow$):

$$\begin{aligned}
\text{True} &\dashv\vdash \Box\,\text{True} \\
\Box(P \wedge Q) &\dashv\vdash \Box P \wedge \Box Q \\
\Box(P \vee Q) &\dashv\vdash \Box P \vee \Box Q \\
\Box \triangleright P &\dashv\vdash \triangleright \Box P \\
\forall x.\ \Box P &\dashv\vdash \Box \forall x.\ P \\
\Box \exists x.\ P &\dashv\vdash \exists x.\ \Box P
\end{aligned}$$

- These facts are not supposed to be intuitively obvious; they rely on the precise semantics, which we do not cover in this course.

Laws for $\square$

$$\textsc{Persistently-sep}\quad \frac{S \vdash \square P \wedge Q}{S \vdash \square P * Q}$$

Derivable rule:

$$\textsc{Persistently-sep-derived}\quad \frac{S \vdash \square (P \wedge Q)}{S \vdash \square (P * Q)}$$

Laws for $\square$

We have two kinds of primitive persistent propositions.

$$t =_\tau t' \dashv\vdash \square(t =_\tau t') \qquad\qquad \{P\}\, e\, \{\Phi\} \dashv\vdash \square\, \{P\}\, e\, \{\Phi\}$$

Finally, we have the following rule generalizing the in-out rules (HT-HT and HT-EQ) we saw earlier:

$$\frac{\textsc{Ht-persistently}\quad \square\, Q \land S \vdash \{P\}\, e\, \{v.\, R\}}{S \vdash \{P \land \square\, Q\}\, e\, \{v.\, R\}}$$

Derived Rules

1. $\Box \Box P \vdash \Box P$
2. $\Box(P \Rightarrow Q) \vdash \Box P \Rightarrow \Box Q$
3. $P \Rightarrow Q \vdash P \mathrel{-\!\!*} Q$
4. $\Box(P \mathrel{-\!\!*} Q) \vdash \Box(P \Rightarrow Q)$
5. $\Box(P \mathrel{-\!\!*} Q) \vdash \Box P \mathrel{-\!\!*} \Box Q$
6. $(P \mathrel{-\!\!*} (Q * \Box R)) * P \vdash (P \mathrel{-\!\!*} (Q * \Box R)) * P * \Box R$

Summary: what you should know about $\Box$

- You just have to know what is included in the slides above.
- Key points:
  - Modality for capturing predicates that only express knowledge (no ownership over resources).
  - Persistence implies duplicable.
  - Persistent predicates are closed under most logical connectives (but not $\Rightarrow$) and under a persistent modality $*$ and $\wedge$ coincide.
  - Equality, Hoare triples, and Invariants (to be introduced later) are all persistent.
  - Persistent predicates can move in and out of preconditions in Hoare triples.
- Suggestion: skim the chapter in the lecture notes.

Iris: Higher-Order Concurrent Separation Logic

Concurrency Intro and Invariants[3]

Amin Timany

Aarhus University, Denmark

August 13, 2026

[3] Based on slides by Lars Birkedal

Thread-Local Reasoning

A Key Point of Concurrent Separation Logic:

- We do not reason about possible interleavings of threads (too many to reason about in a scalable way). See Hans Boehm: You Don't Know Jack About Shared Variables or Memory Models. CACM Vol. 55 No. 2, Pages 48-54.
- We reason about each thread in isolation – thread-local reasoning.
- Important for modular reasoning!

Thread-Local Reasoning

A Key Point of Concurrent Separation Logic:

- We do not reason about possible interleavings of threads (too many to reason about in a scalable way). See Hans Boehm: You Don't Know Jack About Shared Variables or Memory Models. CACM Vol. 55 No. 2, Pages 48-54.
- We reason about each thread in isolation – thread-local reasoning.
- Important for modular reasoning!
- How ?

Thread-Local Reasoning

A Key Point of Concurrent Separation Logic:

- We do not reason about possible interleavings of threads (too many to reason about in a scalable way). See Hans Boehm: You Don't Know Jack About Shared Variables or Memory Models. CACM Vol. 55 No. 2, Pages 48-54.
- We reason about each thread in isolation – thread-local reasoning.
- Important for modular reasoning!
- How ?
  - We
    - either ensure that there are no interesting interleavings among threads (disjoint concurrency),
    - or we abstract over how threads may interfere with each other, so that it is still possible to reason thread-locally.
  - Hence Hoare triples over individual expressions continue to be the basic entity of program proofs (rather than some kind of Hoare triple over thread pools).

Disjoint Concurrency Rule

$$\textsc{Ht-par}\ \frac{S \vdash \{P_1\}\ e_1\ \{v.Q_1\} \qquad S \vdash \{P_2\}\ e_2\ \{v.Q_2\}}{S \vdash \{P_1 * P_2\}\ e_1 \mid\mid e_2\ \{v.\exists v_1 v_2.\ v = (v_1, v_2) * Q_1[v_1/v] * Q_2[v_2/v]\}}$$

- The rule states that we can run e_1 and e_2 in parallel, if they have *disjoint* footprints and that in this case we can verify the two components separately.
- Thus this rule is sometimes also referred to as the *disjoint concurrency rule*.

Disjoint Concurrency Example

- Let e_i be $\ell_i \leftarrow !\,\ell_i + 1$, for $i \in \{1, 2\}$. Then we can use HT-PAR to show:

$$\{\ell_1 \hookrightarrow n * \ell_2 \hookrightarrow m\}\,(e_1 \,||\, e_2); !\,\ell_1 + !\,\ell_2\,\{v.v = n + m + 2\}$$

Disjoint Concurrency Example

- Let e_i be $\ell_i \leftarrow !\,\ell_i + 1$, for $i \in \{1, 2\}$. Then we can use HT-PAR to show:

$$\{\ell_1 \hookrightarrow n * \ell_2 \hookrightarrow m\}\,(e_1 \,||\, e_2); !\,\ell_1 + !\,\ell_2\,\{v.v = n + m + 2\}$$

- More realistic example: merge sort.

Non-disjoint Concurrency Example

- The HT-PAR rule does not suffice to verify a concurrent program which modifies a shared location.
- For instance, we cannot use it to prove

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e); !\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Why?

Non-disjoint Concurrency Example

- The HT-PAR rule does not suffice to verify a concurrent program which modifies a shared location.
- For instance, we cannot use it to prove

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e); !\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Why?
  - We cannot split the $\ell \hookrightarrow n$ predicate to give to the two subcomputations.

Non-disjoint Concurrency Example

- The HT-PAR rule does not suffice to verify a concurrent program which modifies a shared location.
- For instance, we cannot use it to prove

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e); !\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Why?
  - We cannot split the $\ell \hookrightarrow n$ predicate to give to the two subcomputations.
- We need the ability to *share* predicate $\ell \hookrightarrow n$ among the two threads running in parallel.
- That is what *invariants* enable.

Non-disjoint Concurrency Example

- The HT-PAR rule does not suffice to verify a concurrent program which modifies a shared location.
- For instance, we cannot use it to prove

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e);\,!\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Why?
  - We cannot split the $\ell \hookrightarrow n$ predicate to give to the two subcomputations.
- We need the ability to *share* predicate $\ell \hookrightarrow n$ among the two threads running in parallel.
- That is what *invariants* enable.
- Is this even the best spec we can show ?

Non-disjoint Concurrency Example

- The HT-PAR rule does not suffice to verify a concurrent program which modifies a shared location.
- For instance, we cannot use it to prove

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e); !\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Why?
  - We cannot split the $\ell \hookrightarrow n$ predicate to give to the two subcomputations.
- We need the ability to *share* predicate $\ell \hookrightarrow n$ among the two threads running in parallel.
- That is what *invariants* enable.
- Is this even the best spec we can show ?
  - The best we can hope to prove is:

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e); !\,\ell\,\{v.v = n + 1 \vee v = n + 2\}$$

  but that is considerably harder, so won't do that for now.

Invariants

- Add a type of invariant names InvName to the logic.
- Add new term $\boxed{P}^{\iota}$, to be read as "invariant P *named* ι".
- Typing rule:

$$\frac{\Gamma \vdash P : \mathsf{Prop} \qquad \Gamma \vdash \iota : \mathsf{InvName}}{\Gamma \vdash \boxed{P}^{\iota} : \mathsf{Prop}}$$

- Note that there are *no restrictions on* P. In particular, we are also allowed to form *nested invariants*, *e.g.*, terms of the form $\boxed{\boxed{P}^{\iota}}^{\iota'}$.

Invariant Names on Hoare Triples

- There will be rules allowing us to temporarily *open* invariants, and, conceptually, get local ownership over the resources described by the invariant, so that we may operate on those resources.
- Of course, it does not make sense to get local ownership of some resource twice (if we "$*$ on" a resource $\ell \hookrightarrow -$ twice, then we get false).
- Hence we need to ensure that we do not open invariants more than once.
- Hence we index Hoare triples with infinite set of invariant names $\mathcal{E}$:

$$S \vdash \{P\}\ e\ \{v.Q\}_{\mathcal{E}}$$

- This set identifies the invariants we are allowed to use.
- If there is no annotation on the Hoare triple then $\mathcal{E} = \mathsf{InvName}$, the set of all invariant names. With this convention all the previous rules are still valid.

Invariant Names on Hoare Triples

- Just one new rule for relating Hoare triples with different sets of invariant names:

$$\textsc{Ht-mask-weaken}\ \frac{S \vdash \{P\}\, e\, \{v.Q\}_{\mathcal{E}_1} \qquad \mathcal{E}_1 \subseteq \mathcal{E}_2}{S \vdash \{P\}\, e\, \{v.Q\}_{\mathcal{E}_2}}$$

- Intuitively sound: if we can show the triple while being allowed to open $\mathcal{E}_1$ invariants, then we can, of course, also show the triple if we are allowed to open more invariants.

Rules for Invariants: Persistence and Allocation

- A key point of invariants is that they can be shared. Hence invariants are persistent:

$$\textsc{Inv-persistent}\quad \frac{}{\boxed{P}^{\iota} \vdash \Box \boxed{P}^{\iota}}$$

- Invariant allocation rule:

$$\textsc{Ht-inv-alloc}\quad \frac{\mathcal{E} \text{ infinite} \qquad S \land \exists \iota \in \mathcal{E}.\, \boxed{P}^{\iota} \vdash \{Q\}\, e\, \{v.R\}_{\mathcal{E}}}{S \vdash \{\triangleright P * Q\}\, e\, \{v.R\}_{\mathcal{E}}}$$

Rules for Invariants: Invariant Opening Rule

- The invariant opening rule

$$\frac{\textsc{Ht-inv-open} \quad e \text{ is an atomic expression} \qquad S \land \boxed{P}^{\iota} \vdash \{\triangleright P * Q\}\, e\, \{v. \triangleright P * R\}_{\mathcal{E}}}{S \land \boxed{P}^{\iota} \vdash \{Q\}\, e\, \{v. R\}_{\mathcal{E} \uplus \{\iota\}}}$$

  is the only way to get access to the resources governed by an invariant.

Rules for Invariants: Invariant Opening Rule

- The invariant opening rule

$$\frac{\textsc{Ht-inv-open}\quad e \text{ is an atomic expression} \qquad S \land \boxed{P}^{\iota} \vdash \{\triangleright P * Q\}\, e\, \{v. \triangleright P * R\}_{\mathcal{E}}}{S \land \boxed{P}^{\iota} \vdash \{Q\}\, e\, \{v. R\}_{\mathcal{E} \uplus \{\iota\}}}$$

  is the only way to get access to the resources governed by an invariant.

- Thus if we know an invariant $\boxed{P}^{\iota}$ exists, we can *temporarily*, for one atomic step, get access to the resources.
  - An expression is *atomic* if it reduces to a value in *one* reduction step.

Rules for Invariants: Invariant Opening Rule

- The invariant opening rule

$$\frac{\textsc{Ht-inv-open}\quad e \text{ is an atomic expression} \qquad S \land \boxed{P}^{\iota} \vdash \{\triangleright P * Q\}\, e\, \{v. \triangleright P * R\}_{\mathcal{E}}}{S \land \boxed{P}^{\iota} \vdash \{Q\}\, e\, \{v. R\}_{\mathcal{E} \uplus \{\iota\}}}$$

  is the only way to get access to the resources governed by an invariant.
- Thus if we know an invariant $\boxed{P}^{\iota}$ exists, we can *temporarily*, for one atomic step, get access to the resources.
  - An expression is *atomic* if it reduces to a value in *one* reduction step.
- This rule is the reason we need to annotate the Hoare triples with sets of invariant names $\mathcal{E}$.

Regarding ▷ in the Invariant Opening Rule

- Note: we only get access to the resources *later* (▷).
- This is essential, logic would be inconsistent otherwise; proof not covered in this course, see `https://iris-project.org/pdfs/2016-icfp-iris2-final.pdf`
- There is a wide class of *timeless* propositions for which it does not matter.
- Timeless propositions include most ordinary propositions, but not those involving a later modality, an update modality, or a general predicate variable.
- Many concrete examples will thus not need the general rule with later above.
- We therefore did consider leaving it out of this course.
- But it is a key feature of Iris that invariants can contain general predicates (not just timeless ones), in particular predicate variables.
- This is important for giving modular specs, see, e.g., the specification for a lock.
- And for other advanced applications: models of type systems.

Stronger Frame Rule

- Stronger frame rule which allows to remove $\triangleright$ from frame:

$$\frac{\textsc{Ht-frame-atomic}\quad e \text{ is an atomic expression} \qquad S \vdash \{P\}\, e\, \{v.Q\}}{S \vdash \{P * \triangleright R\}\, e\, \{v.Q * R\}}$$

- (We will see an example application of this rule later.)

Remark: Footprint Reading of Hoare Triples

- Earlier "minimal footpring" reading must be refined now.
- Given triple $\{P\}\, e\, \{v.Q\}$, the resources required for running e can
  - either be in the precondition P,
  - or be governed by one or more invariants.
- For example, may prove triples of the form $\{\text{True}\}\, e\, \{v.Q\}$, for some Q, where e accesses shared state governed by an invariant.

Example

- Recall the example we cannot prove with disjoint concurrency rule:

$$\{\ell \hookrightarrow n\}\,(e \mid\mid e);\,!\,\ell\,\{v.v \geq n\}$$

  where e is the program $\ell \leftarrow !\,\ell + 1$.
- Let's prove it now!
- We start by allocating invariant

$$I = \exists m.\, m \geq n \wedge \ell \hookrightarrow m$$

  using HT-INV-ALLOC rule. This is possible by rule of consequence, since $\ell \hookrightarrow n$ implies I and hence $\triangleright I$.

Example proof

- Thus we have to prove

$$\boxed{I}^{\iota} \vdash \{\mathsf{True}\}\,(e \mid\mid e);\,!\,\ell\,\{v.v \geq n\} \tag{1}$$

  for some ι.

- Using the derived sequencing rule HT-SEQ SFTS the following two triples

$$\boxed{I}^{\iota} \vdash \{\mathsf{True}\}\,(e \mid\mid e)\,\{\_.\mathsf{True}\}.$$
$$\boxed{I}^{\iota} \vdash \{\mathsf{True}\}\;!\,\ell\,\{v.v \geq n\}.$$

- We show the first one; during the proof of that we will need to show the second triple as well.
- Using HT-PAR, SFTS

$$\boxed{I}^{\iota} \vdash \{\mathsf{True}\}\,e\,\{\_.\mathsf{True}\}$$

  (Note that we cannot open the invariant now since the expression e is not atomic.)

Example proof

- Using the bind rule we first show

$$\boxed{I}^{\iota} \vdash \{\mathsf{True}\}\ !\,\ell\,\{v.v \geq n\}.$$

- Note that this is exactly the second premise of the sequencing rule mentioned above.
- By invariant opening rule HT-INV-OPEN SFTS

$$\{\triangleright I\}\ !\,\ell\,\{v.v \geq n \wedge \triangleright I\}_{\mathsf{InvName}\setminus\{\iota\}}.$$

- Using rule HT-FRAME-ATOMIC together with HT-LOAD and structural rules we have

$$\{\triangleright I\}\ !\,\ell\,\{v.v = m \wedge m \geq n \wedge \ell \hookrightarrow m\}_{\mathsf{InvName}\setminus\{\iota\}}.$$

From this we easily derive the needed triple.

Example proof

- To show the second premise of the bind rule, SFTS

$$\boxed{I}^{\iota} \vdash \forall m.\, \{m \geq n\}\, \ell \leftarrow (m+1)\, \{\_.\mathsf{True}\}.$$

- To show this we again use the invariant opening rule and HT-FRAME-ATOMIC (exercise!).