

# Iris: Higher-Order Concurrent Separation Logic

## Lecture 7: Later Modality

Lars Birkedal

Aarhus University, Denmark

November 14, 2017

# Overview

Earlier:

- ▶ Operational Semantics of  $\lambda_{\text{ref,conc}}$ 
  - ▶  $e, (h, e) \rightsquigarrow (h, e')$ , and  $(h, \mathcal{E}) \rightarrow (h', \mathcal{E}')$
- ▶ Basic Logic of Resources
  - ▶  $l \hookrightarrow v, P * Q, P \multimap Q, \Gamma \mid P \vdash Q$
- ▶ Basic Separation Logic
  - ▶  $\{P\} e \{v.Q\} : \text{Prop, isList } l \text{ xs, ADTs, foldr}$

Today:

- ▶ Later Modality:  $\triangleright$
- ▶ Necessary for working with *invariants* (defined later in the course)
- ▶ Key Points:
  - ▶ **Löb rule:**  $(\triangleright P \Rightarrow P) \Rightarrow P$
  - ▶ **Guarded recursively defined predicates:**  $\mu r. P$

## Later Modality

- ▶ Recall the recursion rule:

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

- ▶ 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.**

# Plan for today

## Today

- ▶ Rules for reasoning about  $\triangleright$ , including strengthening of earlier Hoare rules.
- ▶ Example
  - ▶ Specification and proof of a fixed point combinator.
  - ▶ Proof relies on  $\triangleright$ .
  - ▶ The example is perhaps somewhat contrived — chosen to illustrate expressiveness without being too long.

## Later on

- ▶ the rules we describe today will be important later on, especially when reasoning about invariants.

# Löb Rule

- ▶ Typing for  $\triangleright$ :

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

- ▶ Löb Rule:

$$\frac{\text{LÖB} \quad S \wedge \triangleright P \vdash P}{S \vdash P}$$

- ▶ 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 proposition is not just a set of resources.
- ▶ In more detail, an Iris proposition  $P$  is<sup>1</sup> 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-indexes 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-indexes.

---

<sup>1</sup>Not really, but closer to being. . .

## Laws for Later Modality

$$\frac{\text{LATER-MONO} \quad Q \vdash P}{\triangleright Q \vdash \triangleright P}$$

$$\frac{\text{LATER-WEAK} \quad Q \vdash P}{Q \vdash \triangleright P}$$

$$\frac{\text{LÖB} \quad Q \wedge \triangleright P \vdash P}{Q \vdash P}$$

## Laws for Later Modality

$$\frac{\text{LATER-CONJ} \quad R \vdash \triangleright(P \wedge Q)}{\underline{\underline{R \vdash \triangleright P \wedge \triangleright Q}}}$$

$$\frac{\text{LATER-DISJ} \quad R \vdash \triangleright(P \vee Q)}{\underline{\underline{R \vdash \triangleright P \vee \triangleright Q}}}$$

$$\frac{\text{LATER-ALL} \quad Q \vdash \triangleright \forall x. P}{\underline{\underline{Q \vdash \forall x. \triangleright P}}}$$

$$\frac{\text{LATER-SEP} \quad R \vdash \triangleright P * \triangleright Q}{\underline{\underline{R \vdash \triangleright(P * Q)}}}$$

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

$$\frac{\triangleright\text{-}\exists \quad Q \vdash \exists x. \triangleright P}{Q \vdash \triangleright \exists x. P}$$



## 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

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

HT-LOAD

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

HT-STORE

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

HT-MATCH

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

## 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

HT-LET

$$\frac{S \vdash \{P\} e_1 \{x. \triangleright Q\} \quad S \vdash \forall v. \{Q[v/x]\} e_2 [v/x] \{u.R\}}{S \vdash \{P\} \text{let } x = e_1 \text{ in } e_2 \{u.R\}}$$

HT-LET-DET

$$\frac{S \vdash \{P\} e_1 \{x. \triangleright (x = v) \wedge \triangleright Q\} \quad S \vdash \{Q[v/x]\} e_2 [v/x] \{u.R\}}{S \vdash \{P\} \text{let } x = e_1 \text{ in } e_2 \{u.R\}}$$

HT-SEQ

$$\frac{S \vdash \{P\} e_1 \{-.\triangleright Q\} \quad S \vdash \{R\} e_2 \{u.R\}}{S \vdash \{P\} e_1; e_2 \{u.R\}} .$$

HT-IF

$$\frac{S \vdash \{P * v = \text{true}\} e_2 \{u.Q\} \quad S \vdash \{P * v = \text{false}\} e_3 \{u.Q\}}{S \vdash \{\triangleright P\} \text{if } v \text{ then } e_2 \text{ else } e_3 \{u.Q\}}$$

# Guarded recursively defined predicates



- ▶ We extend terms of the logic with

$$t ::= \dots \mid \mu x : \tau. t$$

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:

MU-FIXED

$$\frac{}{Q \vdash \mu x : \tau. t =_{\tau} t [\mu x : \tau. t/x]}$$

## Guarded recursively defined predicates



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

$\mu \text{isStream} : Val \rightarrow \text{stream } Val \rightarrow \text{Prop. } \lambda l : Val. \lambda xs : \text{stream } Val.$

$(xs = [] \wedge l = \text{inj}_1()) \vee$

$(\exists x, xs'. xs = x : xs' \wedge \exists hd, l'. l = \text{inj}_2(hd) * hd \hookrightarrow (x, l') * \triangleright(\text{isStream } l' xs))$

- ▶ 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.



- ▶ Mixed-variance guarded recursive predicates are useful for
  - ▶ Interpreting recursive types in a typed programming language by Iris predicates / relations (see ipm-paper).
  - ▶ Defining models of untyped / untyped languages (e.g., for object capabilities).
  - ▶ Specifying and reasoning about libraries that can call themselves recursively, e.g., an event loop library (see iCap-paper).
    - ▶ **M.Sc. Project idea:** Formalize event loop library in Iris in Coq, if ambitious, consider library for asynchronous IO.

## Example: Fixed-point combinator $\Theta_F$

- ▶ Given a value  $F$ , the call-by-value Turing fixed-point combinator  $\Theta_F$  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  $\Theta_F$  as  $\text{rec } f(x) = e$ .

## Proof Rule for $\Theta_F$

- ▶ Now we wish to derive proof rule for  $\Theta_F$ , similar to the recursion rule.

$$\frac{\text{HT-TURING-FP} \quad \Gamma \mid S \wedge \forall v. \{P\} \Theta_{Fv} \{u.Q\} \vdash \forall v. \{P\} F(\lambda x. \Theta_F x) v \{u.Q\}}{\Gamma \mid S \vdash \forall v. \{P\} \Theta_{Fv} \{u.Q\}}$$

- ▶ 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

$$\triangleright \forall v. \{P\} \Theta_{FV} \{u.Q\} \quad (*)$$

- ▶ and we are to show

$$\forall v. \{P\} \Theta_{FV} \{u.Q\}.$$

- ▶ Let  $v$  be a value.
- ▶ By LATER-WEAK and the rule of consequence SFTS

$$\{\triangleright P\} \Theta_{FV} \{u.Q\}.$$

## Proof

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

$$\{\triangleright(\forall v. \{P\} \Theta_{FV} \{u.Q\} \wedge P)\} \Theta_{FV} \{u.Q\}$$

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

$$\{\forall v. \{P\} \Theta_{FV} \{u.Q\} \wedge P\} F(\lambda x. \Theta_{FX})v \{u.Q\}$$

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

$$\{P\} F(\lambda x. \Theta_{FX})v \{u.Q\}$$

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