

# Iris: Higher-Order Concurrent Separation Logic

## Lecture 1: Introduction and Operational Semantics of $\lambda_{\text{ref},\text{conc}}$

Lars Birkedal

Aarhus University, Denmark

April 5, 2021

# Overview

Today:

- ▶ Course Introduction
- ▶ Operational Semantics of  $\lambda_{\text{ref},\text{conc}}$

# Introduction: goals of this course

- ▶ Formal verification of programs written in realistic programming languages
  - ▶ verification can mean many things, depending on which properties we try to verify
  - ▶ the properties we focus on include full functional correctness, so properties are rich / deep
- ▶ We focus on techniques that scale to concurrent higher-order imperative programs
  - ▶ important in practise
  - ▶ hard to reason about, especially modularly

# Applications

- ▶ Verification of challenging concurrent libraries whose correctness is critical (interactively, in the Coq proof assistant)
- ▶ Foundation for semi-automated tools, such as Caper
- ▶ Framework for expressing and proving invariants captured by type systems.
  - ▶ ML types, runST, type-and-effect systems, Rust, ...

# Projects

- ▶ After this course, you can do projects related to above applications, e.g., using our Coq implementation of Iris.
- ▶ Selected Example Projects:
  - ▶ Verifying Hash Tables in Iris, M.Sc. thesis by Esben Clausen, 2017.
  - ▶ Formalizing Concurrent Stacks With Helping: A Case Study In Iris, project by Daniel Gratzer and Mathias Høier, 2017.  
[A version of this is now a chapter in the lecture notes.]
  - ▶ Modular Verification of the Ticket Lock, project by Marit Ohlenbusch, 2018.  
[A version of this is now a chapter in the lecture notes.]
  - ▶ The Array-Based Queueing Lock, project by Simon Vindum and Emil Gjørup, 2019.  
[A version of this is now a chapter in the lecture notes.]
  - ▶ The CHL Lock and Logical Relations in Iris, project by Zongyaun Liu, 2020.

- ▶ A **framework** for higher-order concurrent separation logic
- ▶ Applicable to many different programming languages (see <http://iris-project.org> for examples)
- ▶ **In this course:** we fix a particular higher-order concurrent imperative programming language, called  $\lambda_{\text{ref,conc}}$ .
- ▶ Now: syntax and operational semantics of  $\lambda_{\text{ref,conc}}$ .

# Syntax, I

	$x, y, f$	$\in$	$Var$
	$\ell$	$\in$	$Loc$
	$n$	$\in$	$\mathbb{Z}$
	$\odot$	$::=$	$+ \mid - \mid * \mid = \mid < \mid \dots$
$Val$	$v$	$::=$	$() \mid \text{true} \mid \text{false} \mid n \mid \ell \mid (v, v) \mid \text{inj}_1 v \mid \text{inj}_2 v \mid \text{rec } f(x) = e$
$Exp$	$e$	$::=$	$x \mid n \mid e \odot e \mid () \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e \text{ else } e \mid \ell$ $\mid (e, e) \mid \pi_1 e \mid \pi_2 e \mid \text{inj}_1 e \mid \text{inj}_2 e$ $\mid \text{match } e \text{ with } \text{inj}_1 x \Rightarrow e \mid \text{inj}_2 y \Rightarrow e \text{ end}$ $\mid \text{rec } f(x) = e \mid e \ e$ $\mid \text{ref}(e) \mid !e \mid e \leftarrow e \mid \text{cas}(e, e, e) \mid \text{fork } \{e\}$

## Syntax, II

$ECtx \quad E ::= - \mid E \odot e \mid v \odot E \mid \text{if } E \text{ then } e \text{ else } e \mid (E, e) \mid (v, E) \mid \pi_1 E \mid \pi_2 E$   
 $\mid \text{inj}_1 E \mid \text{inj}_2 E \mid \text{match } E \text{ with } \text{inj}_1 x \Rightarrow e \mid \text{inj}_2 y \Rightarrow e \text{ end}$   
 $\mid E e \mid v E \mid \text{ref}(E) \mid !E \mid E \leftarrow e \mid v \leftarrow E$   
 $\mid \text{cas}(E, e, e') \mid \text{cas}(v, E, e) \mid \text{cas}(v, v', E)$

$Heap \quad h \in Loc \xrightarrow{\text{fin}} Val$

$TPool \quad \mathcal{E} \in \mathbb{N} \xrightarrow{\text{fin}} Exp$

$Config \quad \varsigma ::= (h, \mathcal{E})$



# Syntactic Sugar

- ▶ We write  $\lambda x.e$  for the term  $\text{rec } f(x) = e$  where  $f$  is some fresh variable not appearing in  $e$ . Thus  $\lambda x.e$  is a non-recursive function with argument  $x$  and body  $e$ .
- ▶ We write  $\text{let } x = e_1 \text{ in } e_2$  for the term  $(\lambda x.e_2)e_1$ .
- ▶ We write  $e_1; e_2$  for the term  $\text{let } x = e_1 \text{ in } e_2$  where  $x$  is some fresh variable not appearing in  $e_2$ .

## Pure reduction

$$v \odot v' \overset{\text{pure}}{\rightsquigarrow} v''$$

$$\text{if } v'' = v \odot v'$$

$$\text{if true then } e_1 \text{ else } e_2 \overset{\text{pure}}{\rightsquigarrow} e_1$$

$$\text{if false then } e_1 \text{ else } e_2 \overset{\text{pure}}{\rightsquigarrow} e_2$$

$$\pi_i(v_1, v_2) \overset{\text{pure}}{\rightsquigarrow} v_i$$

$$\text{match inj}_i v \text{ with inj}_1 x_1 \Rightarrow e_1 \mid \text{inj}_2 x_2 \Rightarrow e_2 \text{ end} \overset{\text{pure}}{\rightsquigarrow} e_i[v/x_i]$$

$$(\text{rec } f(x) = e) v \overset{\text{pure}}{\rightsquigarrow} e[(\text{rec } f(x) = e)/f, v/x]$$

## Per-thread one-step reduction

$(h, e) \rightsquigarrow (h, e')$	if $e \overset{\text{pure}}{\rightsquigarrow} e'$
$(h, \text{ref}(v)) \rightsquigarrow (h[\ell \mapsto v], \ell)$	if $\ell \notin \text{dom}(h)$
$(h, !\ell) \rightsquigarrow (h, h(\ell))$	if $\ell \in \text{dom}(h)$
$(h, \ell \leftarrow v) \rightsquigarrow (h[\ell \mapsto v], ())$	if $\ell \in \text{dom}(h)$
$(h, \text{cas}(\ell, v_1, v_2)) \rightsquigarrow (h[\ell \mapsto v_2], \text{true})$	if $h(\ell) = v_1$
$(h, \text{cas}(\ell, v_1, v_2)) \rightsquigarrow (h, \text{false})$	if $h(\ell) \neq v_1$

## Configuration reduction

$$\frac{(h, e) \rightsquigarrow (h', e')}{(h, \mathcal{E}[i \mapsto E[e]]) \rightarrow (h', \mathcal{E}[i \mapsto E[e']])}$$

$$\frac{j \notin \text{dom}(\mathcal{E}) \cup \{i\}}{(h, \mathcal{E}[i \mapsto E[\text{fork } \{e\}]] \rightarrow (h, \mathcal{E}[i \mapsto E[()]] [j \mapsto e])}$$

## Example: factorial

Let  $v = \text{rec fac}(n) = \text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fac}(n - 1)$ . We wish to consider the evaluation of  $v(2)$ . For each step, think about what the evaluation context is.

$$\begin{aligned} ([], [0 \mapsto v(2)]) &\rightsquigarrow ([], [0 \mapsto \text{if } 2 = 0 \text{ then } 1 \text{ else } 2 * v(2 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto \text{if false then } 1 \text{ else } 2 * v(2 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * v(2 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * v(1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * \text{if } 1 = 0 \text{ then } 1 \text{ else } 1 * v(1 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * \text{if false then } 1 \text{ else } 1 * v(1 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1 * v(1 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1 * v(0)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1 * \text{if } 0 = 0 \text{ then } 1 \text{ else } 0 * v(0 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1 * \text{if true then } 1 \text{ else } 0 * v(0 - 1)]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1 * 1]) \\ &\rightsquigarrow ([], [0 \mapsto 2 * 1]) \\ &\rightsquigarrow ([], [0 \mapsto 2]) \end{aligned}$$

## Example: functional lists

Let  $v = \text{rec inc}(xs) = \text{match } xs \text{ with } \text{inj}_1 x_1 \Rightarrow xs \mid \text{inj}_2 x_2 \Rightarrow \text{inj}_2 (1 + \pi_1 x_2, \text{inc}(\pi_2 x_2)) \text{ end}$ .

We consider the evaluation of  $v(\text{inj}_2(7, \text{inj}_1()))$  ( $v$  applied to the list with one element, the value 7).

$([], [0 \mapsto v(\text{inj}_2(7, \text{inj}_1()))])$   
 $\rightsquigarrow ([], [0 \mapsto \text{match } \text{inj}_2(7, \text{inj}_1()) \text{ with } \text{inj}_1 x_1 \Rightarrow \text{inj}_2(7, \text{inj}_1()) \mid \text{inj}_2 x_2 \Rightarrow \text{inj}_2(1 + \pi_1 x_2, v(\pi_2 x_2)) \text{ end}])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(1 + \pi_1(7, \text{inj}_1()), v(\pi_2(7, \text{inj}_1())))])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(1 + 7, v(\pi_2(7, \text{inj}_1())))])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(8, v(\pi_2(7, \text{inj}_1())))])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(8, v(\text{inj}_1()))])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(8, \text{match } \text{inj}_1() \text{ with } \text{inj}_1 x_1 \Rightarrow \text{inj}_1() \mid \text{inj}_2 x_2 \Rightarrow \text{inj}_2(1 + \pi_1 x_2, v(\pi_2 x_2)) \text{ end})])$   
 $\rightsquigarrow ([], [0 \mapsto \text{inj}_2(8, \text{inj}_1())])$

## Example: references

Let  $v = \text{rec swap}(p) = \text{let } z = !(\pi_1 p) \text{ in } \pi_1 p \leftarrow !(\pi_2 p); \pi_2 p \leftarrow z.$

We consider the evaluation of  $v(\text{ref}(2), \text{ref}(3))$ .

$([], [0 \mapsto v(\text{ref}(2), \text{ref}(3))])$

$([l_1 \mapsto 2], [0 \mapsto v(l_1, \text{ref}(3))])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto v(l_1, l_{18})])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto \text{let } z = !(\pi_1(l_1, l_{18})) \text{ in } \pi_1(l_1, l_{18}) \leftarrow !(\pi_2(l_1, l_{18})); \pi_2(l_1, l_{18}) \leftarrow z])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto \text{let } z = !l_1 \text{ in } \pi_1(l_1, l_{18}) \leftarrow !(\pi_2(l_1, l_{18})); \pi_2(l_1, l_{18}) \leftarrow z])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto \text{let } z = 2 \text{ in } \pi_1(l_1, l_{18}) \leftarrow !(\pi_2(l_1, l_{18})); \pi_2(l_1, l_{18}) \leftarrow z])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto \pi_1(l_1, l_{18}) \leftarrow !(\pi_2(l_1, l_{18})); \pi_2(l_1, l_{18}) \leftarrow 2])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto l_1 \leftarrow !(\pi_2(l_1, l_{18})); \pi_2(l_1, l_{18}) \leftarrow 2])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto l_1 \leftarrow !l_{18}; \pi_2(l_1, l_{18}) \leftarrow 2])$

$([l_1 \mapsto 2, l_{18} \mapsto 3], [0 \mapsto l_1 \leftarrow 3; \pi_2(l_1, l_{18}) \leftarrow 2])$

$([l_1 \mapsto 3, l_{18} \mapsto 3], [0 \mapsto \pi_2(l_1, l_{18}) \leftarrow 2])$

$([l_1 \mapsto 3, l_{18} \mapsto 3], [0 \mapsto l_{18} \leftarrow 2])$

$([l_1 \mapsto 3, l_{18} \mapsto 2], [0 \mapsto ()])$

## Example: concurrency

Let  $e = \text{fork } \{(1 + 2) + 3\}; (4 + 5) + 6$ .

We consider the evaluation of  $e$ , and just show one possible reduction sequence (more than one possible). Notice the interleaving of reductions in the two threads.

$([], [0 \mapsto e])$   
 $([], [0 \mapsto (); (4 + 5) + 6, 1 \mapsto (1 + 2) + 3])$   
 $([], [0 \mapsto (4 + 5) + 6, 1 \mapsto (1 + 2) + 3])$   
 $([], [0 \mapsto 9 + 6, 1 \mapsto (1 + 2) + 3])$   
 $([], [0 \mapsto 9 + 6, 1 \mapsto 3 + 3])$   
 $([], [0 \mapsto 9 + 6, 1 \mapsto 6])$   
 $([], [0 \mapsto 15, 1 \mapsto 6])$