Iris: A Modular Foundation for Higher-Order Concurrent Separation Logic[1]

Jacques-Henri Jourdan[2] **Robbert Krebbers**[3]

[2]CNRS, LRI, Université Paris-Sud, France

[3]Delft University of Technology, The Netherlands

January 8, 2018 @ POPL Tutorials, Los Angeles

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

Preparation for this tutorial

- Download the tutorial lecture material
  `http://iris-project.org/tutorial`
- Follow README to install Iris **3.1**

Iris Proof Mode (IPM)

Many recent program logics come with mechanized soundness proofs, but how to reason in these logics?

Goal of IPM: reasoning in Iris in the same style as reasoning in Coq

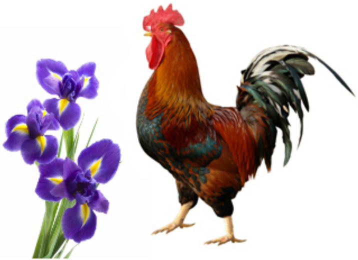
Iris Proof Mode (IPM)

Many recent program logics come with mechanized soundness proofs, but how to reason in these logics?

Goal of IPM: reasoning in Iris in the same style as reasoning in Coq

Features of IPM:

- Extends Coq with (spatial and non-spatial) named proof contexts for Iris
- Tactics for introduction and elimination of all connectives of Iris
- Entirely implemented using reflection, type classes and Ltac (no OCaml plugin needed)



Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
P ∗ (∃ a : A, Ψ a) ∗ R −∗ ∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
P ∗ (∃ a : A, Ψ a) ∗ R −∗ ∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
"HP" : P
"HΨ" : ∃ a : A, Ψ a
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
"HP" : P
"HΨ" : ∃ a : A, Ψ a
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HΨ" : Ψ x
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HΨ" : Ψ x
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HΨ" : Ψ x
"HR" : R
--------------------------------------∗
Ψ x ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HΨ" : Ψ x
"HR" : R
--------------------------------------∗
Ψ x ∗ P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
```

```
2 subgoals
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/2)
"HΨ" : Ψ x
--------------------------------------∗
Ψ x

______________________________________(2/2)
"HP" : P
"HR" : R
--------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  -
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HΨ" : Ψ x
--------------------------------------∗
Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HΨ" : Ψ x
--------------------------------------∗
Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
```

```
This subproof is complete, but there are
some unfocused goals:

______________________________________(1/1)
"HP" : P
"HR" : R
--------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
  -
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HR" : R
--------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
  - iAssumption.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"HP" : P
"HR" : R
--------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
  - iAssumption.
```

```
No more subgoals.
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDestruct "HΨ" as (x) "HΨ".
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
  - iAssumption.
Qed.
```

```
No more subgoals.
```

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  iDest
  iExists x.
  iSplitL "HΨ".
  - iAssumption.
  - iAssumption.
Qed.
```

Logical notations overridden in scope for Iris

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
"HP" : P
"HΨ" : ∃ a : A, Ψ a
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Notation for deeply embedded context

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  Unset Printing Notations.
```

```
1 subgoal
M : ucmraT
A : Type
P, R : iProp
Ψ : A → iProp
______________________________________(1/1)
"HP" : P
"HΨ" : ∃ a : A, Ψ a
"HR" : R
--------------------------------------∗
∃ a : A, Ψ a ∗ P
```

Notation for deeply embedded context

Iris Proof Mode (IPM) demo

```
Lemma and_exist_sep {A} P R (Ψ: A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ R −∗ ∃ a, Ψ a ∗ P.
Proof.
  iIntros "[HP [HΨ HR]]".
  Unset Printing Notations.
```

```
1 subgoal
M : ucmraT
A : Type@{Top.105}
P, R : uPred M
Ψ : forall _ : A, uPred M
______________________________________(1/1)
@uPred_entails M
 (@of_envs M
   (@Envs M (@Enil (uPred M))
    (@Esnoc (uPred M)
      (@Esnoc (uPred M)
       (@Esnoc (uPred M) (@Enil (uPred M))
         (String
          (Ascii false false false true false
     false true
              false)
          (String
            (Ascii false false false false true
      false true
               false) EmptyString)) P)
       (String
```

Motivation

Why should we care about interactive proofs? Why not automate everything?

Infeasible to automate everything, for example:

- 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 IPM

Coq demo