MoSeL: A General, Extensible Modal Framework for Interactive Proofs in Separation Logic

Robbert Krebbers[1] Jacques-Henri Jourdan[2] Ralf Jung[3] Joseph Tassarotti[4]
Jan-Oliver Kaiser[3] Amin Timany[5] Arthur Charguéraud[6] Derek Dreyer[3]

[1]Delft University of Technology, The Netherlands
[2]LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France
[3]MPI-SWS, Germany
[4]Carnegie Mellon University, USA
[5]imec-Distrinet, KU Leuven, Belgium
[6]Inria & Université de Strasbourg, CNRS, ICube, France

September 10, 2018 @ Inria, Paris, France

Proofs in separation logic

You have a new separation logic, what do you do?

1. Prove that the logic is sound
2. Use it to reason about programs

Proofs in separation logic

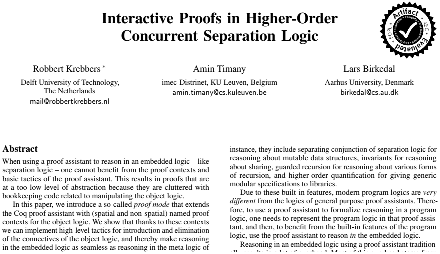
You have a new separation logic, what do you do?

1. Prove that the logic is sound in Coq
2. Use it to reason about programs

Proofs in separation logic

You have a new separation logic, what do you do?

1. Prove that the logic is sound — in Coq
2. Use it to reason about programs — using Iris Proof Mode



Interactive Proofs in Higher-Order Concurrent Separation Logic

Robbert Krebbers *
Delft University of Technology, The Netherlands
mail@robbertkrebbers.nl

Amin Timany
imec-Distrinet, KU Leuven, Belgium
amin.timany@cs.kuleuven.be

Lars Birkedal
Aarhus University, Denmark
birkedal@cs.au.dk

Abstract

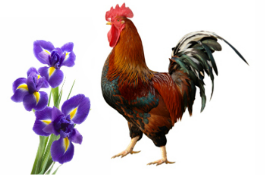
When using a proof assistant to reason in an embedded logic – like separation logic – one cannot benefit from the proof contexts and basic tactics of the proof assistant. This results in proofs that are at a too low level of abstraction because they are cluttered with bookkeeping code related to manipulating the object logic.

In this paper, we introduce a so-called *proof mode* that extends the Coq proof assistant with (spatial and non-spatial) named proof contexts for the object logic. We show that thanks to these contexts we can implement high-level tactics for introduction and elimination of the connectives of the object logic, and thereby make reasoning in the embedded logic as seamless as reasoning in the meta logic of

instance, they include separating conjunction of separation logic for reasoning about mutable data structures, invariants for reasoning about sharing, guarded recursion for reasoning about various forms of recursion, and higher-order quantification for giving generic modular specifications to libraries.

Due to these built-in features, modern program logics are *very different* from the logics of general purpose proof assistants. Therefore, to use a proof assistant to formalize reasoning in a program logic, one needs to represent the program logic in that proof assistant, and then, to benefit from the built-in features of the program logic, use the proof assistant to reason *in* the embedded logic.

Reasoning in an embedded logic using a proof assistant tradition-



Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
P ∗ (∃ a : A, Φ a ∨ Ψ a) −∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
P ∗ (∃ a : A, Φ a ∨ Ψ a) -∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
```

```
2 subgoals
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/2)
"HP" : P
"H1" : Φ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a

______________________________________(2/2)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  -
```

```
2 subgoals
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/2)
"HP" : P
"H1" : Φ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a

______________________________________(2/2)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  -
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
P ∗ Φ x ∨ P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
P ∗ Φ x ∨ P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
P ∗ Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
---------------------------------------∗
P ∗ Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
```

```
2 subgoals
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------∗
P

______________________________________(2/2)
"H1" : Φ x
--------------------------------------∗
Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    +
```

```
2 subgoals
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------∗
P

______________________________________(2/2)
"H1" : Φ x
--------------------------------------∗
Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    +
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
_______________________________________(1/1)
"HP" : P
---------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------∗
P
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
```

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

______________________________________(1/2)
"H1" : Φ x
--------------------------------------∗
Φ x

______________________________________(2/2)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    +
```

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

______________________________________(1/2)
"H1" : Φ x
--------------------------------------∗
Φ x

______________________________________(2/2)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    +
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"H1" : Φ x
--------------------------------------∗
Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"H1" : Φ x
--------------------------------------∗
Φ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
```

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

______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  -
```

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

______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  -
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
P ∗ Φ x ∨ P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
P ∗ Φ x ∨ P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
    iSplitL "HP"; iAssumption.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
    iSplitL "HP"; iAssumption.
```

```
No more subgoals.
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
    iSplitL "HP"; iAssumption.
Qed.
```

```
No more subgoals.
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
    + iAssumption.
    + iAssumption.
  - iExists x.
    iRight.
    iSplitL "HP"; iAssumption.
Qed.
```

```
example_1 is defined
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
```

Logical notations overridden in scope for separation logic

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  Unset Printing Notations.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
______________________________________(1/1)
"HP" : P
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, P ∗ Φ a ∨ P ∗ Ψ a
```

Notation for deeply embedded context

Iris Proof Mode (IPM) demo

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  Unset Printing Notations.
```

```
1 subgoal
A : Type
P : ofe_car (uPredC (iResUR Σ))
Φ, Ψ : forall _ : A, ofe_car (uPredC (iResUR Σ))
______________________________________(1/1)
@envs_entails (uPredI (iResUR Σ))
 (@Envs (uPredI (iResUR Σ)) (@Enil (bi_car (uPredI (iResUR Σ))))
   (@Esnoc (bi_car (uPredI (iResUR Σ)))
    (@Esnoc (bi_car (uPredI (iResUR Σ))) (@Enil (bi_car (uPredI
      (iResUR Σ))))
      (INamed
       (String (Ascii.Ascii false false false true false false
      true false)
          (String
           (Ascii.Ascii false false false false true false true
      false)
            EmptyString))) P)
    (INamed
      (String (Ascii.Ascii false false false true false false
      true false)
       EmptyString))
    (@bi_exist (uPredI (iResUR Σ)) A
      (fun a : A => @bi_or (uPredI (iResUR Σ)) (Φ a) (Ψ a))))
   (xI xH))
 (@bi_exist (uPredI (iResUR Σ)) A
   (fun a : A =>
   @bi_or (uPredI (iResUR Σ)) (@bi_sep (uPredI (iResUR Σ)) P (Φ
      a))
```

The **good things** of Iris Proof Mode

- **Proofs have the look and feel of Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`

The **good things** of Iris Proof Mode

- **Proofs have the look and feel of Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for all features of Iris**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...

The **good things** of Iris Proof Mode

- **Proofs have the look and feel of Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for all features of Iris**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...
- **Integration with tactics for proving programs**
  Symbolic execution tactics for weakest preconditions

The **good things** of Iris Proof Mode

- **Proofs have the look and feel of Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for all features of Iris**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...
- **Integration with tactics for proving programs**
  Symbolic execution tactics for weakest preconditions
- **It scales to non-trivial projects**
  - Safety of Rust and its standard libraries [Jung *et. al.*, POPL'18]
  - Encapsulation of the ST monad [Timany *et. al.*, POPL'18]
  - A calculus for program refinements [Frumin *et. al.*, LICS'18]
  - Verification of object capability patterns [Swasey *et. al.*, OOPSLA'17]
  - Soundness of a logic for weak memory [Kaiser *et. al.*, ECOOP'17]

The **bad thing** of Iris Proof Mode

The implementation is tied to Iris

The **bad thing** of Iris Proof Mode

The implementation is tied to Iris

~~Iris~~ Proof Mode

The **bad thing** of Iris Proof Mode

The implementation is tied to Iris

~~Iris~~ Proof Mode

Our contribution:

MoSeL: A General, Extensible Modal Framework for Interactive Proofs in Separation Logic

ROBBERT KREBBERS, Delft University of Technology, The Netherlands
JACQUES-HENRI JOURDAN, LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France
RALF JUNG, MPI-SWS, Germany
JOSEPH TASSAROTTI, Carnegie Mellon University, USA
JAN-OLIVER KAISER, MPI-SWS, Germany
AMIN TIMANY, imec-Distrinet, KU Leuven, Belgium
ARTHUR CHARGUÉRAUD, Inria & Université de Strasbourg, CNRS, ICube, France
DEREK DREYER, MPI-SWS, Germany

A number of tools have been developed for carrying out separation-logic proofs mechanically using an interactive proof assistant. One of the most advanced such tools is the Iris Proof Mode (IPM) for Coq, which offers a rich set of tactics for making separation-logic proofs look and feel like ordinary Coq proofs. However, IPM is tied to a particular separation logic (namely, Iris), thus limiting its applicability.

Making IPM independent of Iris (1)

[. . .] we believe that our proof mode is very generic, and can be applied to a variety of different embedded logics [. . .]

[Krebbers *et. al.*, POPL'17]

Making IPM independent of Iris (2)

Doing it in a **generic fashion** turned out to be challenging:

- Iris is affine, not all separation logics are affine

$$P * Q \vdash P \tag{affine}$$

MoSeL supports general and affine separation logics, and mixtures thereof

Making IPM independent of Iris (2)

Doing it in a **generic fashion** turned out to be challenging:

- Iris is affine, not all separation logics are affine

$$P * Q \vdash P \qquad \text{(affine)}$$

  MoSeL supports general and affine separation logics, and mixtures thereof

- IPM has hard-wired support for Iris's connectives, but other logics have other bespoke connectives

  MoSeL is parametric in the connectives/modalities of the logic

Making IPM independent of Iris (2)

Doing it in a **generic fashion** turned out to be challenging:

- Iris is affine, not all separation logics are affine

  $$P * Q \vdash P \qquad \text{(affine)}$$

  MoSeL supports general and affine separation logics, and mixtures thereof
- IPM has hard-wired support for Iris's connectives, but other logics have other bespoke connectives
  MoSeL is parametric in the connectives/modalities of the logic
- Some separation logics (*e.g.* iGPS) are encoded in terms of another (*e.g.* Iris), and mix both levels of abstraction
  MoSeL's tactics allow reasoning in a mixture of logics

Making IPM independent of Iris (2)

Doing it in a **generic fashion** turned out to be challenging:

- Iris is affine, not all separation logics are affine

$$P * Q \vdash P \qquad \text{(affine)}$$

  MoSeL supports general and affine separation logics, and mixtures thereof

- IPM has hard-wired support for Iris's connectives, but other logics have other bespoke connectives
  MoSeL is parametric in the connectives/modalities of the logic

- Some separation logics (*e.g.* iGPS) are encoded in terms of another (*e.g.* Iris), and mix both levels of abstraction
  MoSeL's tactics allow reasoning in a mixture of logics

- Lots of Coq engineering to make it **actually usable**
  Backwards compatibility with IPM, performance, error messages, ...

Part #1: Basic tactics in IPM/MoSeL

Embedding separation logic entailments into Coq

Visible goal (with pretty printing):

$\vec{\mathrm{x}} \ : \ \vec{\phi}$ Variables and pure Coq hypotheses

Π Spatial separation logic hypotheses

R Separation logic goal

Embedding separation logic entailments into Coq

Visible goal (with pretty printing):

$$\begin{array}{l} \vec{x} : \vec{\phi} \quad \text{Variables and pure Coq hypotheses} \\ \text{---} \\ \Pi \quad \text{Spatial separation logic hypotheses} \\ \text{---}* \\ R \quad \text{Separation logic goal} \end{array}$$

Actual Coq goal (without pretty printing):

$$\begin{array}{l} \vec{x} : \vec{\phi} \\ \text{---} \\ \Pi \Vdash Q \end{array}$$

Where:

$$\Pi \Vdash Q \triangleq \mathop{\Huge *} \Pi \vdash Q$$

Example: the `iSplitL`/`iSplitR` tactic

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
P ∗ Φ x
```

Example: the `iSplitL`/`iSplitR` tactic

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iSplitL "HP".
```

```
1 subgoal
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/1)
"HP" : P
"H1" : Φ x
--------------------------------------∗
P ∗ Φ x
```

Example: the iSplitL/iSplitR tactic

```
Lemma example_1 {A} (P : iProp Σ) (Φ Ψ : A → iProp Σ) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, (P ∗ Φ a) ∨ (P ∗ Ψ a).
Proof.
  iIntros "[HP H]".
  iDestruct "H" as (x) "[H1|H2]".
 - iExists x.
   iLeft.
```

```
2 subgoals
A : Type
P : iProp Σ
Φ, Ψ : A → iProp Σ
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------∗
P

______________________________________(2/2)
"H1" : Φ x
--------------------------------------∗
Φ x
```

Example: Implementation of the `iSplitL`/`iSplitR` tactic

Tactics implemented by reflection as mere lemmas:

```
Lemma tac_sep_split Π Π₁ Π₂ lr js Q₁ Q₂ :
  envs_split lr js Π = Some (Π₁,Π₂) →
  (Π₁ ⊢ Q₁) → (Π₂ ⊢ Q₂) → Π ⊢ Q₁ ∗ Q₂.
```

$$\frac{\Pi_1 \Vdash Q_1 \qquad \Pi_2 \Vdash Q_2}{\Pi_1, \Pi_2 \Vdash Q_1 * Q_2}$$

Example: Implementation of the `iSplitL`/`iSplitR` tactic

Tactics implemented by reflection as mere lemmas:

```
Lemma tac_sep_split Π Π₁ Π₂ lr js Q₁ Q₂ :
  envs_split lr js Π = Some (Π₁,Π₂) →
  (Π₁ ⊢ Q₁) → (Π₂ ⊢ Q₂) → Π ⊢ Q₁ ∗ Q₂.
```

$$\frac{\Pi_1 \Vdash Q_1 \qquad \Pi_2 \Vdash Q_2}{\Pi_1, \Pi_2 \Vdash Q_1 * Q_2}$$

Context splitting implemented as a computable Coq function

Example: Implementation of the `iSplitL`/`iSplitR` tactic

Tactics implemented by reflection as mere lemmas:

```
Lemma tac_sep_split Π Π₁ Π₂ lr js Q₁ Q₂ :
  envs_split lr js Π = Some (Π₁,Π₂) →
  (Π₁ ⊢ Q₁) → (Π₂ ⊢ Q₂) → Π ⊢ Q₁ ∗ Q₂.
```

$$\frac{\Pi_1 \Vdash Q_1 \qquad \Pi_2 \Vdash Q_2}{\Pi_1, \Pi_2 \Vdash Q_1 * Q_2}$$

Context splitting implemented as a computable Coq function

Ltac wrappers around the reflective tactic:

```
Tactic Notation "iSplitL" constr(Hs) :=
  let Hs := words Hs in
  let Hs := eval vm_compute in (INamed <$> Hs) in
  eapply tac_sep_split with _ _ Left Hs _ _;
    [pm_reflexivity ||
      fail "iSplitL: hypotheses" Hs "not found"
    | (* goal 1 *)
    | (* goal 2 *) ].
```

Example: Implementation of the iSplitL/iSplitR tactic

Tactics implemented by reflection as mere lemmas:

```
Lemma tac_sep_split Π Π1 Π2 lr js Q1 Q2 :
  envs_split lr js Π = Some (Π1,Π2) →
  (Π1 ⊢ Q1) → (Π2 ⊢ Q2) → Π ⊢ Q1 ∗ Q2.
```

$$\frac{\Pi_1 \Vdash Q_1 \qquad \Pi_2 \Vdash Q_2}{\Pi_1, \Pi_2 \Vdash Q_1 * Q_2}$$

Context splitting implemented as a computable Coq function

Ltac wrappers around the reflective tactic:

```
Tactic Notation "iSplitL" constr(Hs) :=
  let Hs := words Hs in
  let Hs := eval vm_compute in (INamed <$> Hs) in
  eapply tac_sep_split with _ _ Left Hs _ _;
    [pm_reflexivity ||
      fail "iSplitL: hypotheses" Hs "not found"
    | (* goal 1 *)
    | (* goal 2 *) ].
```

Report sensible error to the user

Making MoSeL separation logic independent

First step: Make everything parametric in a BI logic

```
Structure bi := Bi {
  bi_car     :> Type;
  bi_pure     : Prop → bi_car;
  bi_entails : bi_car → bi_car → Prop;
  bi_forall  : ∀ A, (A → bi_car) → bi_car;
  bi_sep      : bi_car → bi_car → bi_car;
  (* other separation logic operations and axioms *)
}.
Notation "P ⊢ Q" := (bi_entails P Q).
```

Making MoSeL separation logic independent

First step: Make everything parametric in a BI logic

```
Structure bi := Bi {
  bi_car     :> Type;
  bi_pure     : Prop → bi_car;
  bi_entails : bi_car → bi_car → Prop;
  bi_forall  : ∀ A, (A → bi_car) → bi_car;
  bi_sep      : bi_car → bi_car → bi_car;
  (* other separation logic operations and axioms *)
}.
Notation "P ⊢ Q" := (bi_entails P Q).

Record envs (PROP : bi) :=
  Envs { env_spatial : env PROP;    (* the spatial context Π *)
         env_counter : positive     (* a counter for fresh name generation *) }.
Definition envs_entails {PROP} (Δ : envs PROP) (Q : PROP) : Prop :=
  ⌜ envs_wf Δ ⌝ ∧ [∗] env_spatial Δ ⊢ Q.
```

Making MoSeL separation logic independent

First step: Make everything parametric in a BI logic

```
Structure bi := Bi {
  bi_car     :> Type;
  bi_pure     : Prop → bi_car;
  bi_entails : bi_car → bi_car → Prop;
  bi_forall  : ∀ A, (A → bi_car) → bi_car;
  bi_sep      : bi_car → bi_car → bi_car;
  (* other separation logic operations and axioms *)
}.
Notation "P ⊢ Q" := (bi_entails P Q).

Record envs (PROP : bi) :=
  Envs { env_spatial : env PROP;   (* the spatial context Π *)
         env_counter : positive    (* a counter for fresh name generation *) }.
Definition envs_entails {PROP} (Δ : envs PROP) (Q : PROP) : Prop :=
  ⌜ envs_wf Δ ⌝ ∧ [∗] env_spatial Δ ⊢ Q.
```

Useful fact: primitive records provide a significant performance boost

Part #2: Affine versus general BI logics

$$\cancel{\mathbf{P} * \mathbf{Q} \vdash \mathbf{P}}$$

Affinety in IPM tactics

Problem: many IPM tactics relied on affinety of Iris

$$P * Q \vdash P \qquad \text{(affine)}$$

For example:

`iClear`

$$\frac{\Pi \Vdash Q}{\Pi, P \Vdash Q}$$

`iAssumption`

$$\Pi, P \Vdash P$$

Affinety in IPM tactics

Problem: many IPM tactics relied on affinety of Iris

$$P * Q \vdash P \qquad \text{(affine)}$$

For example:

`iClear`

$$\frac{\Pi \Vdash Q}{\Pi, P \Vdash Q}$$

`iAssumption`

$$\Pi, P \Vdash P$$

Many logics (*e.g.* CFML and CHL) are **not** affine, MoSeL should support them

What to do with these tactics?

We cannot remove these tactics:

- That destroys backwards compatibility with IPM

What to do with these tactics?

We cannot remove these tactics:

- That destroys backwards compatibility with IPM

We cannot include these tactics just for affine logics:

- Some logics use a mixture of affine and linear resources
  For example: Fairis [Tassarotti et. al., ESOP'17]

What to do with these tactics?

We cannot remove these tactics:

- That destroys backwards compatibility with IPM

We cannot include these tactics just for affine logics:

- Some logics use a mixture of affine and linear resources
  For example: Fairis [Tassarotti et. al., ESOP'17]

Better solution: add *precise* side-conditions to these tactics

Affine and absorbing propositions

Two classes of propositions:

$$\text{affine}(P) \triangleq P \vdash \text{emp} \qquad \text{(propositions that can be “thrown away”)}$$

$$\text{absorbing}(Q) \triangleq Q * \text{True} \vdash Q \qquad \text{(propositions that can “suck up others”)}$$

The new tactics:

$$\texttt{iClear}\ \frac{\Pi \Vdash Q \qquad \text{affine}(P) \text{ or } \text{absorbing}(Q)}{\Pi, P \Vdash Q}$$

$$\texttt{iAssumption}\ \frac{\text{affine}(\Pi) \text{ or } \text{absorbing}(Q)}{\Pi, Q \Vdash Q}$$

Affine and absorbing propositions

Two classes of propositions:

$$\mathsf{affine}(P) \triangleq P \vdash \mathsf{emp} \qquad \text{(propositions that can be “thrown away”)}$$

$$\mathsf{absorbing}(Q) \triangleq Q * \mathsf{True} \vdash Q \qquad \text{(propositions that can “suck up others”)}$$

The new tactics:

`iClear`
$$\frac{\Pi \Vdash Q \qquad \mathsf{affine}(P) \textit{ or } \mathsf{absorbing}(Q)}{\Pi, P \Vdash Q}$$

`iAssumption`
$$\frac{\mathsf{affine}(\Pi) \textit{ or } \mathsf{absorbing}(Q)}{\Pi, Q \Vdash Q}$$

Key features:

- Full backwards compatibility with Iris: all Iris propositions are affine and absorbing because $\mathsf{emp} \triangleq \mathsf{True}$ in Iris
- Provides support for logics with both linear and affine resources

Affine and absorbing propositions in Coq

Type classes:

```
Class Affine {PROP : bi} (Q : PROP) := affine : Q ⊢ emp.
Class Absorbing {PROP : bi} (P : PROP) := absorbing : <absorb> P ⊢ P.
(* where <absorb> P := P ∗ True *)
```

Instances:

- To capture that both classes are closed under most connectives
- To allow logics to tell MoSeL that their bespoke connectives are affine/absorbing

Tactics are parameterized by said type classes:

```
Lemma tac_clear Δ Δ' i p P Q :
  envs_lookup_delete true i Δ = Some (p,P,Δ') →
  (if p then TCTrue else TCOr (Affine P) (Absorbing Q)) →
  envs_entails Δ' Q →
  envs_entails Δ Q.
```

Part #3: Intuitionistic propositions

$$\square\,\mathbf{P} \vdash \square\,\mathbf{P} * \square\,\mathbf{P}$$

Classes of separation logic propositions in MoSeL

Kind	**# of times it should be used**
Arbitrary proposition	1 times
Affine proposition	0-1 times

Classes of separation logic propositions in MoSeL

Kind	**# of times it should be used**
Arbitrary proposition	1 times
Affine proposition	0-1 times
Persistent proposition	1-n times

Classes of separation logic propositions in MoSeL

Kind	**# of times it should be used**
Arbitrary proposition	1 times
Affine proposition	0-1 times
Persistent proposition	1-n times
Intuitionistic proposition	0-n times (= affine & persistent)

Classes of separation logic propositions in MoSeL

Kind	**# of times it should be used**
Arbitrary proposition	1 times
Affine proposition	0-1 times
Persistent proposition	1-n times
Intuitionistic proposition	0-n times (= affine & persistent)

Persistent/intuitionistic propositions are common (especially in Iris derivatives)
$\Rightarrow$ MoSeL needs special support for them

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
______________________________________(1/1)
P ∗ (∃ a : A, Φ a ∨ Ψ a) −∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
______________________________________(1/1)
P ∗ (∃ a : A, Φ a ∨ Ψ a) -∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H" : ∃ a : A, Φ a ∨ Ψ a
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
```

```
2 subgoals
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a

______________________________________(2/2)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  -
```

```
2 subgoals
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a

______________________________________(2/2)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  -
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
Φ x ∨ P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
Φ x ∨ P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
Φ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H1" : Φ x
--------------------------------------∗
Φ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
```

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

______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  -
```

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

______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  -
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
∃ a : A, Φ a ∨ P ∗ P ∗ Ψ a
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
Φ x ∨ P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
Φ x ∨ P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
```

```
2 subgoals
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/2)
"HP" : P
--------------------------------------□
P

______________________________________(2/2)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    +
```

```
2 subgoals
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
_______________________________________(1/2)
"HP" : P
---------------------------------------□
P

_______________________________________(2/2)
"HP" : P
---------------------------------------□
"H2" : Ψ x
---------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    +
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
P
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) −∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
P
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
```

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

______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    +
```

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

______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    +
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    + iSplitR; iAssumption.
```

```
1 subgoal
PROP : bi
A : Type
P : PROP
Persistent0 : Persistent P
Affine0 : Affine P
Φ, Ψ : A → PROP
x : A
______________________________________(1/1)
"HP" : P
--------------------------------------□
"H2" : Ψ x
--------------------------------------∗
P ∗ Ψ x
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    + iSplitR; iAssumption.
```

```
No more subgoals.
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    + iSplitR; iAssumption.
Qed.
```

```
No more subgoals.
```

Intuitionistic propositions in action

```
Lemma example_3 {PROP : bi} {A} (P : PROP)
    `{!Persistent P, !Affine P} (Φ Ψ : A → PROP) :
  P ∗ (∃ a, Φ a ∨ Ψ a) -∗ ∃ a, Φ a ∨ (P ∗ P ∗ Ψ a).
Proof.
  iIntros "[#HP H]".
  iDestruct "H" as (x) "[H1|H2]".
  - iExists x.
    iLeft.
    iAssumption.
  - iExists x.
    iRight.
    iSplitR.
    + iAssumption.
    + iSplitR; iAssumption.
Qed.
```

```
example_3 is defined
```

Intuitionistic propositions from the user's point of view

Visible goal in MoSeL:

$\vec{x} : \vec{\phi}$ Variables and pure Coq hypotheses

Γ Intuitionistic separation logic hypotheses

---□

Π Spatial separation logic hypotheses

---∗

R Separation logic goal

We thus need to extend the form of the entailment relation:

$$\Gamma; \Pi \Vdash Q$$

Intuitionistic propositions from the user's point of view

Visible goal in MoSeL:

$\vec{x} : \vec{\phi}$ Variables and pure Coq hypotheses

Γ Intuitionistic separation logic hypotheses

---$\Box$

Π Spatial separation logic hypotheses

---$*$

R Separation logic goal

We thus need to extend the form of the entailment relation:

$$\Gamma; \Pi \Vdash Q$$

Requirements:

- The context Γ should be duplicable (by tactics like `iSplitL`)

Intuitionistic propositions from the user's point of view

Visible goal in MoSeL:

$\vec{x} : \vec{\phi}$ Variables and pure Coq hypotheses

$\overline{\hspace{20em}}$

Γ Intuitionistic separation logic hypotheses

$\overline{\hspace{20em}}\square$

Π Spatial separation logic hypotheses

$\overline{\hspace{20em}}*$

R Separation logic goal

We thus need to extend the form of the entailment relation:

$$\Gamma; \Pi \Vdash Q$$

Requirements:

- The context Γ should be duplicable (by tactics like `iSplitL`)
- The context Γ should be droppable (by tactics like `iAssumption`)

Intuitionistic propositions from the user's point of view

Visible goal in MoSeL:

$\vec{x} : \vec{\phi}$ Variables and pure Coq hypotheses

\---------------------------------------

Γ Intuitionistic separation logic hypotheses

\---------------------------------------$\square$

Π Spatial separation logic hypotheses

\---------------------------------------$*$

R Separation logic goal

We thus need to extend the form of the entailment relation:

$$\Gamma; \Pi \Vdash Q$$

Requirements:

- The context Γ should be duplicable (by tactics like `iSplitL`)
- The context Γ should be droppable (by tactics like `iAssumption`)
- The context Γ should be closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, $\ulcorner \_ \urcorner$, ...

How to model persistent/intuitionistic propositions

- We defined affine/absorbing propositions in terms of the BI connectives

$$\mathsf{affine}(P) \triangleq P \vdash \mathsf{emp} \qquad \text{(propositions that can be "thrown away")}$$

$$\mathsf{absorbing}(Q) \triangleq Q * \mathsf{True} \vdash Q \qquad \text{(propositions that can "suck up others")}$$

How to model persistent/intuitionistic propositions

- We defined affine/absorbing propositions in terms of the BI connectives

$$\text{affine}(P) \triangleq P \vdash \text{emp} \qquad \text{(propositions that can be “thrown away”)}$$

$$\text{absorbing}(Q) \triangleq Q * \text{True} \vdash Q \qquad \text{(propositions that can “suck up others”)}$$

- For persistent propositions in Iris this is impossible [Bizjak&Birkedal, MFPS'17]

How to model persistent/intuitionistic propositions

- We defined affine/absorbing propositions in terms of the BI connectives

$$\mathsf{affine}(P) \triangleq P \vdash \mathsf{emp} \qquad \text{(propositions that can be "thrown away")}$$

$$\mathsf{absorbing}(Q) \triangleq Q * \mathsf{True} \vdash Q \qquad \text{(propositions that can "suck up others")}$$

- For persistent propositions in Iris this is impossible [Bizjak&Birkedal, MFPS'17]
- We extend the signature of BIs with a **persistence modality** $\boxdot$ and define:

$$\mathsf{persistent}(P) \triangleq P \vdash \boxdot P \qquad \mathsf{intuitionistic}(P) \triangleq P \vdash \langle affine \rangle \boxdot P$$

  where $\langle affine \rangle\, Q \triangleq Q \wedge \mathsf{emp}$
- Think of $\boxdot P$ as "P holds for resources that can be duplicated"

How to model persistent/intuitionistic propositions

- We defined affine/absorbing propositions in terms of the BI connectives

$$\mathrm{affine}(P) \triangleq P \vdash \mathrm{emp} \qquad \text{(propositions that can be "thrown away")}$$

$$\mathrm{absorbing}(Q) \triangleq Q * \mathrm{True} \vdash Q \qquad \text{(propositions that can "suck up others")}$$

- For persistent propositions in Iris this is impossible [Bizjak&Birkedal, MFPS'17]
- We extend the signature of BIs with a **persistence modality** $\boxdot$ and define:

$$\mathrm{persistent}(P) \triangleq P \vdash \boxdot P \qquad \mathrm{intuitionistic}(P) \triangleq P \vdash \langle \mathit{affine} \rangle \boxdot P$$

  where $\langle \mathit{affine} \rangle\, Q \triangleq Q \wedge \mathrm{emp}$
- Think of $\boxdot P$ as "P holds for resources that can be duplicated"
- This gives rise to what we call a **MoBI**: BI with $\boxdot$

How to model persistent/intuitionistic propositions

- We defined affine/absorbing propositions in terms of the BI connectives

$$\text{affine}(P) \triangleq P \vdash \text{emp} \qquad \text{(propositions that can be "thrown away")}$$

$$\text{absorbing}(Q) \triangleq Q * \text{True} \vdash Q \qquad \text{(propositions that can "suck up others")}$$

- For persistent propositions in Iris this is impossible [Bizjak&Birkedal, MFPS'17]
- We extend the signature of BIs with a **persistence modality** $\boxdot$ and define:

$$\text{persistent}(P) \triangleq P \vdash \boxdot P \qquad \text{intuitionistic}(P) \triangleq P \vdash \langle \mathit{affine} \rangle \boxdot P$$

where $\langle \mathit{affine} \rangle\, Q \triangleq Q \wedge \text{emp}$
- Think of $\boxdot P$ as "P holds for resources that can be duplicated"
- This gives rise to what we call a **MoBI**: BI with $\boxdot$

Question to answer: what are the laws for MoBIs?

Primitive laws of the persistence modality

- $\boxdot\, P \vdash \boxdot\, Q$, provided $P \vdash Q$
  We can introduce $\boxdot$
- $\boxdot\, P \vdash \boxdot(\boxdot\, P)$

Primitive laws of the persistence modality

- $\boxdot\, P \vdash \boxdot\, Q$, provided $P \vdash Q$
  We can introduce $\boxdot$
- $\boxdot\, P \vdash \boxdot(\boxdot\, P)$
- $\mathsf{emp} \vdash \boxdot\, \mathsf{emp}$
  emp is persistent
- $(\forall x.\ \boxdot\, P) \vdash \boxdot\, (\forall x.\ P)$ and $\boxdot\, (\exists x.\ P) \vdash (\exists x.\ \boxdot\, P)$
  Closed under $\forall$, $\exists$, $\wedge$, $\vee$, True, False (reverse directions are admissible)

Primitive laws of the persistence modality

- $\boxdot\, P \vdash \boxdot\, Q$, provided $P \vdash Q$
  We can introduce $\boxdot$
- $\boxdot\, P \vdash \boxdot(\boxdot\, P)$
- $\mathsf{emp} \vdash \boxdot\, \mathsf{emp}$
  emp is persistent
- $(\forall x.\ \boxdot\, P) \vdash \boxdot\,(\forall x.\ P)$ and $\boxdot\,(\exists x.\ P) \vdash (\exists x.\ \boxdot\, P)$
  Closed under $\forall$, $\exists$, $\wedge$, $\vee$, True, False (reverse directions are admissible)
- $(\boxdot\, P) * Q \vdash \boxdot\, P$
  Persistent propositions are absorbing (remember, they can be used **1**-*n* times)

Primitive laws of the persistence modality

- $\boxdot P \vdash \boxdot Q$, provided $P \vdash Q$
  We can introduce $\boxdot$
- $\boxdot P \vdash \boxdot(\boxdot P)$
- $\mathsf{emp} \vdash \boxdot \mathsf{emp}$
  emp is persistent
- $(\forall x.\ \boxdot P) \vdash \boxdot(\forall x.\ P)$ and $\boxdot(\exists x.\ P) \vdash (\exists x.\ \boxdot P)$
  Closed under $\forall$, $\exists$, $\wedge$, $\vee$, True, False (reverse directions are admissible)
- $(\boxdot P) * Q \vdash \boxdot P$
  Persistent propositions are absorbing (remember, they can be used **1**-n times)
- $(\boxdot P) \wedge Q \vdash P * Q$
  From this we get the elimination rules $(\boxdot P) \vdash P * (\boxdot P)$ and $(\boxdot P) \wedge \mathsf{emp} \vdash P$

Derived laws of the intuitionistic modality

Let $\Box\, P \triangleq \langle \mathit{affine} \rangle \boxdot P$

- The usual laws for monotonicity/idempotence/commuting with BI connectives

Derived laws of the intuitionistic modality

Let $\Box\, P \triangleq \langle \mathit{affine} \rangle \boxdot\, P$

- The usual laws for monotonicity/idempotence/commuting with BI connectives
- $\Box\, P \vdash \mathsf{emp}$
  Intuitionistic propositions are affine

Derived laws of the intuitionistic modality

Let $\Box P \triangleq \langle affine \rangle \boxdot P$

- The usual laws for monotonicity/idempotence/commuting with BI connectives
- $\Box P \vdash \mathsf{emp}$
  Intuitionistic propositions are affine
- $\Box P \vdash P$
  Elimination of the $\Box$ modality
- $\Box P \dashv\vdash \Box P * \Box P$
  Intuitionistic propositions are duplicable

Derived laws of the intuitionistic modality

Let $\square\, P \triangleq \langle \mathit{affine} \rangle \boxdot P$

- The usual laws for monotonicity/idempotence/commuting with BI connectives
- $\square\, P \vdash \mathsf{emp}$
  Intuitionistic propositions are affine
- $\square\, P \vdash P$
  Elimination of the $\square$ modality
- $\square\, P \dashv\vdash \square\, P * \square\, P$
  Intuitionistic propositions are duplicable
- $\square\, P * \square\, P \dashv\vdash \square\, P \land \square\, P$
  $\land$ and $*$ coincide for intuitionistic propositions

Why do the laws of the persistence modality make sense?

- They satisfy the requirements from MoSeL's UI point of view
- They are backwards compatible with Iris's laws
- They are compatible with traditional classical/intuitionstic separation logic
- They are compatible with Fairis [Tassarotti *et. al.*, ESOP'17], which features mixed affine/linear resources
- We have developed a model based on **ordered resource algebras** where the laws of $\boxdot$ correspond to canonical properties of the order relation
  This model generalizes classical/intuitionistic separation logic, Iris, and Fairis

The entailment relation and some tactics

The entailment relation:

$$\Gamma; \Pi \Vdash Q \quad \triangleq \quad \square\left(\bigwedge \Gamma\right) * \mathop{\Huge *} \Pi \vdash Q$$

where $\square\, P \triangleq \langle affine \rangle \boxdot P$

The entailment relation and some tactics

The entailment relation:

$$\Gamma; \Pi \Vdash Q \quad \triangleq \quad \Box\left(\bigwedge \Gamma\right) * \mathop{\Huge *} \Pi \vdash Q$$

where $\Box\, P \triangleq \langle \mathit{affine} \rangle \boxdot P$

Some tactics:

$$\texttt{iSplitL/iSplitR}\ \frac{\Gamma; \Pi_1 \Vdash Q_1 \qquad \Gamma; \Pi_2 \Vdash Q_2}{\Gamma; \Pi_1, \Pi_2 \Vdash Q_1 * Q_2}$$

$$\texttt{iIntros-\#}\ \frac{\Gamma, P; \Pi \Vdash Q \qquad \text{intuitionistic}(P)}{\Gamma; \Pi, P \Vdash Q}$$

Part #4: Extensibility of MoSeL

Making MoSeL tactics modular using type classes (1)

We want `iDestruct "H" as "[H1 H2]"` (for example) to:

- turn `H : P ∗ Q` into `H1 : P` and `H2 : Q`
- turn `H : ▷(P ∗ Q)` into `H2 : ▷ P` and `H2 : ▷ Q`
- turn $\texttt{H} : \texttt{l} \mapsto \texttt{v}$ into $\texttt{H1} : \texttt{l} \xmapsto{1/2} \texttt{v}$ and $\texttt{H2} : \texttt{l} \xmapsto{1/2} \texttt{v}$

Making MoSeL tactics modular using type classes (1)

We want `iDestruct "H" as "[H1 H2]"` (for example) to:

- turn `H : P ∗ Q` into `H1 : P` and `H2 : Q`
- turn `H : ▷(P ∗ Q)` into `H2 : ▷ P` and `H2 : ▷ Q`
- turn `H :` $l \mapsto v$ into `H1 :` $l \xmapsto{1/2} v$ and `H2 :` $l \xmapsto{1/2} v$

We follow the IPM approach to use type classes for that:

```
Class IntoSep {PROP : bi} (P Q₁ Q₂ : PROP) := into_sep : P ⊢ Q₁ ∗ Q₂.
Instance into_sep_sep P Q : IntoSep (P ∗ Q) P Q.
Instance into_sep_later P Q₁ Q₂ : IntoSep P Q₁ Q₂ → IntoSep (▷ P) (▷ Q₁) (▷ Q₂).
Instance into_sep_mapsto l q v : IntoSep (l ↦{q} v) (l ↦{q/2} v) (l ↦{q/2} v).

Lemma tac_and_destruct Δ Δ' i p j₁ j₂ P P₁ P₂ Q :
  envs_lookup i Δ = Some (p, P) →
  (if p then IntoAnd true P P₁ P₂ else IntoSep P P₁ P₂) →
  envs_simple_replace i p (Esnoc (Esnoc Enil j₁ P₁) j₂ P₂) Δ = Some Δ' →
  envs_entails Δ' Q → envs_entails Δ Q.
```

Making MoSeL tactics modular using type classes (2)

- We made every tactic MoSeL parametric by a type class
- Generalized these type classes to support general BIs
- Since the type classes are parametric in the choice of the BI PROP:

```
Class IntoSep {PROP : bi} (P Q₁ Q₂ : PROP) := into_sep : P ⊢ Q₁ ∗ Q₂.
```

We now also support connectives that involve multiple BIs:

```
Global Instance into_sep_embed `{BiEmbed PROP PROP'} P Q₁ Q₂ :
  IntoSep P Q₁ Q₂ → IntoSep ⎡P⎤ ⎡Q₁⎤ ⎡Q₂⎤.
```

Many modalities

Many logics come with bespoke modalities that need custom introduction and elimination tactics, for example:

$$\frac{\Gamma; \emptyset \Vdash Q \qquad \mathrm{affine}(\Pi)}{\Gamma; \Pi \Vdash \Box\, Q}\ \Box\text{-}\textsc{Intro}$$

$$\frac{\Gamma; \emptyset \Vdash Q}{\Gamma; \Pi \Vdash \boxdot\, Q}\ \boxdot\text{-}\textsc{Intro}$$

$$\frac{\Gamma; \emptyset \Vdash Q \qquad \mathrm{affine}(\Pi)}{\Gamma; \Pi \Vdash \langle \mathit{affine} \rangle\, Q}\ \langle \mathit{affine} \rangle\text{-}\textsc{Intro}$$

$$\frac{\Gamma'; \Pi' \Vdash Q \qquad \Gamma \vdash \triangleright \Gamma' \qquad \Pi \vdash \triangleright \Pi'}{\Gamma; \Pi \Vdash \triangleright Q}\ \triangleright\text{-}\textsc{Intro}$$

Generic tactics for modalities

MoSeL comes with generic support for introduction and elimination of modalities

Generic tactics for modalities

MoSeL comes with generic support for introduction and elimination of modalities

For introduction:

- One has to choose the action on both contexts that should be performed
- That's done by declaring a type class instance
- As part of which one has to prove that the required laws hold

Generic tactics for modalities in Coq

```
Inductive modality_action (PROP₁ : bi) : bi → Type :=
  | MIEnvIsEmpty {PROP₂ : bi} : modality_action PROP₁ PROP₂
  | MIEnvForall (C : PROP₁ → Prop) : modality_action PROP₁ PROP₁
  | MIEnvTransform {PROP₂ : bi} (C : PROP₂ → PROP₁ → Prop) : modality_action PRO
    P₁ PROP₂
  | MIEnvClear {PROP₂} : modality_action PROP₁ PROP₂
  | MIEnvId : modality_action PROP₁ PROP₁.

Record modality (PROP₁ PROP₂ : bi) := Modality {
  modality_car :> PROP₁ → PROP₂;
  modality_intuitionistic_action : modality_action PROP₁ PROP₂;
  modality_spatial_action : modality_action PROP₁ PROP₂;
  (* The modality laws, which depend on the fields modality_intuitionistic_action
     and modality_spatial_action *) }.

Class FromModal {PROP₁ PROP₂ : bi} (M : modality PROP₁ PROP₂)
    (P : PROP₂) (Q : PROP₁) := from_modal : M Q ⊢ P.
Instance from_modal_affinely P : FromModal modality_affinely (<affine> P) P.
```

Part #5: Conclusions

What's more in the paper?

- Instantiations of MoSeL using 6 very different logics
  Iris, Fairis, iGPS, CFML, CHL, our ordered RA model
- Semi-automated tactics using MoSeL for CFML and CHL
  To support read-only permissions in CFML
- Reasoning in mixed logics (iGPS and Iris)
- A generic model for MoBIs based on *ordered resource algebras*

MoSeL: A General, Extensible Modal Framework for Interactive Proofs in Separation Logic

ROBBERT KREBBERS, Delft University of Technology, The Netherlands
JACQUES-HENRI JOURDAN, LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France
RALF JUNG, MPI-SWS, Germany
JOSEPH TASSAROTTI, Carnegie Mellon University, USA
JAN-OLIVER KAISER, MPI-SWS, Germany
AMIN TIMANY, imec-Distrinet, KU Leuven, Belgium
ARTHUR CHARGUÉRAUD, Inria & Université de Strasbourg, CNRS, ICube, France
DEREK DREYER, MPI-SWS, Germany

A number of tools have been developed for carrying out separation-logic proofs mechanically using an interactive proof assistant. One of the most advanced such tools is the Iris Proof Mode (IPM) for Coq, which offers a rich set of tactics for making separation-logic proofs look and feel like ordinary Coq proofs. However, IPM is tied to a particular separation logic (namely, Iris), thus limiting its applicability.

Thank you!

Download MoSeL at `http://iris-project.org/`