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

Robbert Krebbers[1]

Delft University of Technology, The Netherlands

October 9, 2018 @ Types Meeting, Aarhus, Denmark

[1]MoSeL is joint work with Jacques-Henri Jourdan, Ralf Jung, Joseph Tassarotti, Jan-Oliver Kaiser, Amin Timany, Arthur Charguéraud, and Derek Dreyer

Separation logic [O'Hearn, Reynolds, and Yang, 2001]

Propositions P, Q denote ownership of resources

Separating conjunction $P * Q$:
The resources consists of separate parts satisfying P and Q

Basic example:

$$\{x \mapsto v_1 * y \mapsto v_2\} swap(x, y) \{x \mapsto v_2 * y \mapsto v_1\}$$

the $*$ ensures that x and y are different memory locations

Why is separation logic useful?

Separation logic is very useful:

- It provides a high level of modularity
- It scales to fancy PL features like concurrency

Just in Coq, there is an ever growing collection of separation logics:

- Bedrock
- CFML
- Charge!
- CHL
- FCSL
- Iris
- VST
- ...

The challenge

When developing a new separation logic in a proof assistant, one has to:

1. Prove soundness
2. Develop tactics to carry out proofs

The challenge

When developing a new separation logic in a proof assistant, one has to:

1. Prove soundness
2. Develop tactics to carry out proofs

These steps are tedious, can we simplify them?

In prior work, we proposed solutions for both problems:

1. Proving soundness: **Iris** [POPL'15, ICFP'16, ESOP'17, JFP'18]
2. Tactics: **Iris Proof Mode** [POPL'17]

Iris [POPL'15, ICFP'16, ESOP'17, JFP'18]

A general, language-independent, framework for modeling your own domain specific higher-order separation logics

Iris [POPL'15, ICFP'16, ESOP'17, JFP'18]

A general, language-independent, framework for modeling your own domain specific higher-order separation logics

- **General:** unifies the reasoning principles in many other logics

Iris [POPL'15, ICFP'16, ESOP'17, JFP'18]

A general, language-independent, framework for modeling your own domain specific higher-order separation logics

- **General:** unifies the reasoning principles in many other logics
- **Language-independent:** parameterized by the language

Iris [POPL'15, ICFP'16, ESOP'17, JFP'18]

A general, language-independent, framework for modeling your own domain specific higher-order separation logics

- **General:** unifies the reasoning principles in many other logics
- **Language-independent:** parameterized by the language
- **Modeling logics:** can be used to model domain specific logics
  - iGPS for weak memory [ECOOP'17]
  - RustBelt's lifetime logic [POPL'18]
  - ReLoC for program refinements [LICS'18]
  - Iron for resource management [manuscript]

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iInt
  iDes
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Lemma in the Iris logic

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"H1" : P
"H2" : Ψ x
"H3" : Q
--------------------------------------∗
Q ∗ (∃ a : A, P ∗ Ψ a)
```

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________(1/1)
"H1" : P
"H2" : Ψ x
"H3" : Q
-------------------------∗
Q ∗ (∃ a : A, P ∗ Ψ a)
```

∗ means: resources should be split

Iris Proof Mode [Krebbers *et al.*, POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
```

The hypotheses for the left conjunct

```
Qed.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________(1/1)
"H1" : P
"H2" : Ψ x
"H3" : Q
______________________∗
Q ∗ (∃ a : A, P ∗ Ψ a)
```

∗ means: resources should be split

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
```

The hypotheses for the left conjunct

```
Qed.
```

```
2 subgoals
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________________________(1/2)
"H3" : Q
--------------------------------------∗
Q

______________________________________(2/2)
"H1" : P
"H2" : Ψ x
--------------------------------------∗
∃ a : A, P ∗ Ψ a
```

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Iris Proof Mode [Krebbers *et al.*,POPL'17]: Coq tactics for Iris

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q -∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  by iFrame.
Qed.
```

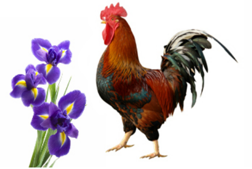
We can also solve this lemma automatically

```
No more subgoals.
```

The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

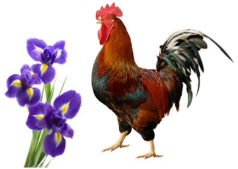
- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`



The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

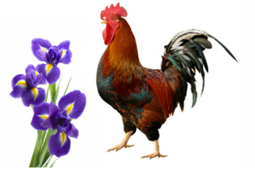
- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for advanced features of separation logic**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...



The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

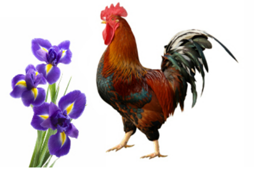
- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for advanced features of separation logic**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...
- **Integration with tactics for proving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



The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for advanced features of separation logic**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...
- **Integration with tactics for proving programs**
  Symbolic execution tactics for weakest preconditions
- **Usable in practice**
  Used for any project involving Iris today



The **bad thing** about Iris Proof Mode

The implementation is tied to Iris

~~Iris~~ Proof Mode

Making Iris Proof Mode independent of Iris

It sounds easy **[Krebbers et al., POPL'17]**:

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

Making Iris Proof Mode independent of Iris

It sounds easy [Krebbers et al., POPL'17]:

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

But doing it generally will be be more challenging

Problem #1: Iris propositions are affine

In Iris you may "forget" about resources:

$$\{\ell_1 \mapsto v_1 * \ell_2 \mapsto v_2\}\, \ell_2 := !\, \ell_1\, \{\ell_2 \mapsto v_1\}$$

Problem #1: Iris propositions are affine

In Iris you may "forget" about resources:

$$\{\ell_1 \mapsto v_1 * \ell_2 \mapsto v_2\}\, \ell_2 := !\, \ell_1\, \{\ell_2 \mapsto v_1\}$$

Due to the **affinity axiom** $P * Q \vdash Q$, which is hard-wired into many tactics:

`iClear`

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

`iAssumption`

$$\Pi, P \Vdash P$$

Problem #1: Iris propositions are affine

In Iris you may "forget" about resources:

$$\{\ell_1 \mapsto v_1 * \ell_2 \mapsto v_2\}\; \ell_2 := !\,\ell_1 \;\{\ell_2 \mapsto v_1\}$$

Due to the **affinity axiom** $P * Q \vdash Q$, which is hard-wired into many tactics:

`iClear`

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

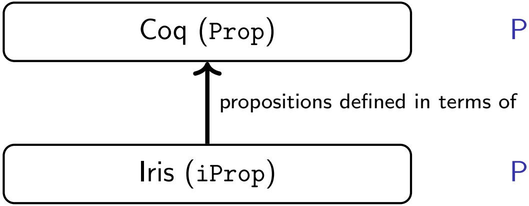
`iAssumption`

$$\Pi, P \Vdash P$$

Not having the affinity axiom is useful: precise accounting of resources (recall Aleš's talk)

Challenge: How to disentangle the affinity axiom from the Iris tactics?

Problem #2: No tactical support for derived logics



Proof using standard Coq tactics

Proof using Iris tactics

Problem #2: No tactical support for derived logics

Coq (`Prop`) — Proof using standard Coq tactics

↑ propositions defined in terms of

Iris (`iProp`) — Proof using Iris tactics

↑ propositions defined in terms of, $\mathrm{iGpsProp} \triangleq \mathrm{View} \xrightarrow{\text{mon}} \mathrm{iProp}$

Derived logic (*e.g.* `iGpsProp`)

Problem #2: No tactical support for derived logics

Coq (`Prop`) — Proof using standard Coq tactics

↑ propositions defined in terms of

Iris (`iProp`) — Proof using Iris tactics

↑ propositions defined in terms of, $\text{iGpsProp} \triangleq \text{View} \xrightarrow{\text{mon}} \text{iProp}$

Derived logic (*e.g.* `iGpsProp`) — Proof using **???**

Problem #2: No tactical support for derived logics

Coq (`Prop`) — Proof using standard Coq tactics

↑ propositions defined in terms of

Iris (`iProp`) — Proof using Iris tactics

↑ propositions defined in terms of, iGpsProp $\triangleq$ View $\xrightarrow{\text{mon}}$ iProp

Derived logic (*e.g.* `iGpsProp`) — Proof using **???**

Challenge: How to reason in logics defined in terms of another

Contributions

MoSeL: A General, Extensible Modal Framework for Interactive Proofs in Separation Logic in Coq [Krebbers *et al.*,ICFP'18]

Contributions:

- MoSeL is parameterized by a general abstraction of separation logic
- MoSeL supports general and affine separation logics, and combinations thereof
- MoSeL supports reasoning in derived separation logics
- MoSeL can be fine-tuned for each logic using type classes

[2] Not in the ICFP'18 paper

Contributions

MoSeL: A General, Extensible Modal Framework for Interactive Proofs in Separation Logic in Coq [Krebbers *et al.*,ICFP'18]

Contributions:

- MoSeL is parameterized by a general abstraction of separation logic
- MoSeL supports general and affine separation logics, and combinations thereof
- MoSeL supports reasoning in derived separation logics
- MoSeL can be fine-tuned for each logic using type classes

MoSeL is usable in practice: we used it on 6 very different existing separation logics

CFML CHL Fairis iGPS Iris Iron[2]

[2] Not in the ICFP'18 paper

Implementation of tactics in IPM/MoSeL

Separation logic entailments in Coq

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Separation logic entailments in Coq

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________________________(1/1)
"H1" : P
"H2" : Ψ x
"H3" : Q
--------------------------------------∗
Q ∗ (∃ a : A, P ∗ Ψ a)
```

Separation logic entailments in Coq

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  Set Printing All.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
________________________(1/1)
"H1" : P
"H2" : Ψ x
```

Notation for deeply embedded context

```
--------------------------------∗
Q ∗ (∃ a : A, P ∗ Ψ a)
```

Separation logic entailments in Coq

```
Lemma test {A} (P Q : iProp) (Ψ : A → iProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  Set Printing All.
```

```
1 subgoal
A : Type
P, Q : iProp
Ψ : A → iProp
x : A
______________________(1/1)
envs_entails (Envs Enil
 (Esnoc (Esnoc (Esnoc Enil
   (String (Ascii false false
    false true false false true
    false)
   (String (Ascii true false
    false false true true false
    false)
   EmptyString)) P)
   ...
```

How to embed separation logic entailments into Coq?

Visible goal (with pretty printing):

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

———————————————————————————

Π Spatial separation logic hypotheses

———————————————————————————∗

R Separation logic goal

How to embed separation logic entailments into Coq?

Visible goal (with pretty printing):

$$\begin{array}{ll}
\vec{x} : \vec{\phi} & \text{Variables and pure Coq hypotheses} \\
\hline
\Pi & \text{Spatial separation logic hypotheses} \\
\hline
R & \text{Separation logic goal}
\end{array}$$

Actual Coq goal (without pretty printing):

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

Where:

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

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.
```

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

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

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 *) ].
```

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.
```

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

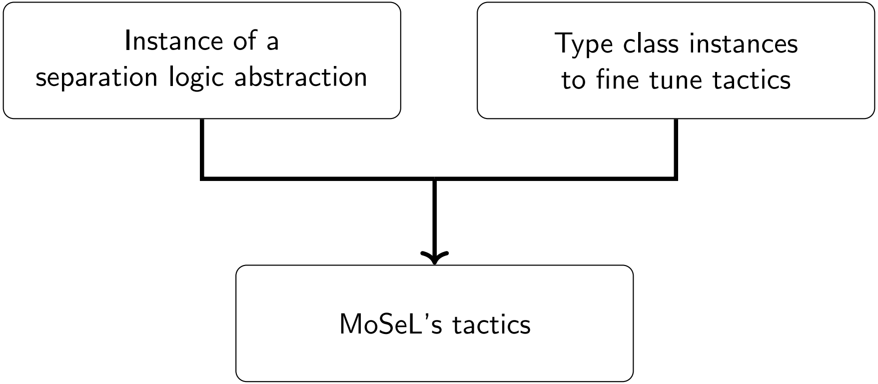
Context splitting implemented as a computable Coq function

Ltac wrappers around the reflective tactic:

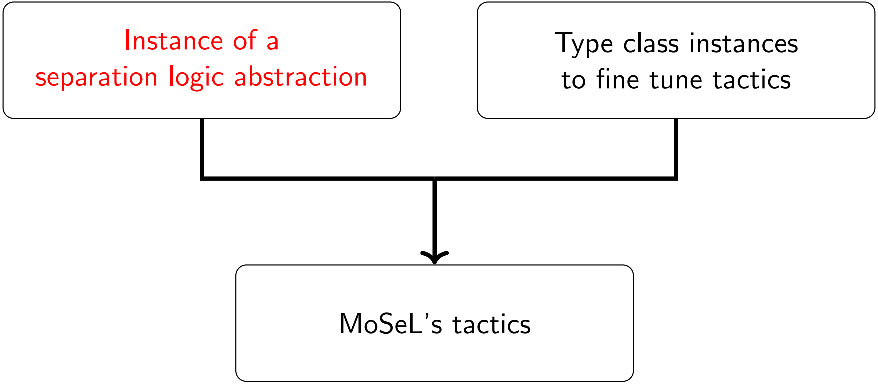
```
Tactic Notation "iSplitL" constr(Hs) :=
  let Hs := words Hs in
  let Hs := eval vm_compute in (INamed <$> Hs) in
  eapply tac_sep_split with _ _ Left Hs _ _;
    [pm_reflexivity ||
      fail "iSplitL: hypotheses" Hs "not found"
    | (* goal 1 *)
    | (* goal 2 *) ].
```

Report sensible error to the user

Making MoSeL separation logic independent



Making MoSeL separation logic independent



How to abstract over separation logics?

BI logics: An abstract interface for separation logics

A **Bunched Implications (BI) logic** [O'Hearn&Pym,99] is a preorder $(Prop, \vdash)$ with:

- Operations $\mathsf{True}, \mathsf{False}, \wedge, \vee, \Rightarrow, \forall, \exists$ satisfying the axioms of intuitionistic logic
- Operations $\mathsf{emp}, *, -\!\!*$ satisfying:

$$\begin{aligned} \mathsf{emp} * P &\dashv\vdash P \\ P * Q &\vdash Q * P \\ (P * Q) * R &\vdash P * (Q * R) \end{aligned} \qquad \frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * P_2 \vdash Q_1 * Q_2} \qquad \frac{P * Q \vdash R}{\overline{P \vdash Q -\!\!* R}}$$

BI logics: An abstract interface for separation logics

A **Bunched Implications (BI) logic** [O'Hearn&Pym,99] is a preorder $(Prop, \vdash)$ with:

- Operations $\mathsf{True}, \mathsf{False}, \wedge, \vee, \Rightarrow, \forall, \exists$ satisfying the axioms of intuitionistic logic
- Operations $\mathsf{emp}, *, \mathrel{-\!\!*}$ satisfying:

$$\begin{array}{rcl} \mathsf{emp} * P & \dashv\vdash & P \\ P * Q & \vdash & Q * P \\ (P * Q) * R & \vdash & P * (Q * R) \end{array} \qquad \frac{P_1 \vdash Q_1 \qquad P_2 \vdash Q_2}{P_1 * P_2 \vdash Q_1 * Q_2} \qquad \frac{P * Q \vdash R}{\overline{\overline{P \vdash Q \mathrel{-\!\!*} R}}}$$

```
Structure bi := Bi {
  bi_car     :> Type;
  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 operators and axioms *)
}.
```

Proofs in MoSeL

Proofs in a specific logic:

```
Lemma test {A} (P Q : iGpsProp) (Ψ : A → iGpsProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Proofs for all logics:

```
Lemma test {PROP : bi} {A} (P Q : PROP) (Ψ : A → PROP) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
    iFrame.
Qed.
```

Proofs in MoSeL

Proofs in a specific logic:

```
Lemma test {A} (P Q : iGpsProp) (Ψ : A → iGpsProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
    iAssumption.
  – iExists x.
    iFrame.
Qed.
```

Lemma for another logic than Iris

Proofs for all logics:

```
Lemma test {PROP : bi} {A} (P Q : PROP) (Ψ : A → PROP) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  – iAssumption.
  – iExists x.
    iFrame.
Qed.
```

Proofs in MoSeL

Proofs in a specific logic:

```
Lemma test {A} (P Q : iGpsProp) (Ψ : A → iGpsProp) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
    iAssumption.
  – iExists x.
    iFrame.
Qed.
```

Lemma for another logic than Iris

Proofs for all logics:

```
Lemma test {PROP : bi} {A} (P Q : PROP) (Ψ : A → PROP) :
  P ∗ (∃ a, Ψ a) ∗ Q −∗ Q ∗ ∃ a, P ∗ Ψ a.
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  – iAssumption.
  – iExists x.
    iFrame.
```

Lemma universally quantified in the BI logic

Addressing challenge #1:

Disentangling the affinity axiom

$$P * Q \vdash Q$$

A poor man's solution

Make two versions of the tactics

1. For affine logics (like Iris and iGPS)
2. For non-affine logics (like CFML and CHL)

A poor man's solution

Make two versions of the tactics

1. For affine logics (like Iris and iGPS)
2. For non-affine logics (like CFML and CHL)

Problems:

- Duplicate work/maintenance
- Some logics mix affine and non-affine propositions, for example:

GC locations (affine)	**Non-GC locations** (not affine)
$\ell \mapsto_{\mathrm{gc}} v$	$\ell \mapsto v$

(Another example in [Tassarotti *et al.*, ESOP'17])

Key idea

- **Don't**: classify whether the whole logic is affine
- **Do:** classify whether individual propositions are affine

Classifying whether propositions are affine

Affine propositions:

$$\mathrm{affine}(P) \triangleq P \vdash \mathrm{emp}$$

(propositions that can be "thrown away")

The new tactics:

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

$$\texttt{iAssumption}\ \frac{\mathrm{affine}(\Pi)}{\Pi, Q \Vdash Q}$$

Classifying whether propositions are affine in Coq

A new type class:

```
Class Affine {PROP : bi} (Q : PROP) := affine : Q ⊢ emp.
```

Instances:

- Tell MoSeL that specific connectives are affine:

  ```
  Instance mapsto_gc_affine l v : Affine (l ↦gc v).
  ```

- Capture that affine propositions are closed under most connectives:

  ```
  Instance sep_affine {PROP : bi} (P Q : bi) :
    Affine P → Affine Q → Affine (P ∗ Q).
  ```

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

What about modalities?

The affine modality

The **affine modality**:

$$
\begin{aligned}
\langle \mathit{affine} \rangle\, P \quad &\triangleq \quad P \wedge \mathsf{emp} \\
&\approx \quad \text{“}P\text{ holds using just affine resources”}
\end{aligned}
$$

This modality is useful for reasoning about affinity

The affine modality

The **affine modality**:

$$\begin{aligned} \langle \mathit{affine} \rangle\, P &\triangleq P \wedge \mathsf{emp} \\ &\approx \text{“}P\text{ holds using just affine resources”} \end{aligned}$$

This modality is useful for reasoning about affinity

- Can be used to turn any proposition into an affine version, *e.g.* A wand that can be dropped $\langle \mathit{affine} \rangle\, (P \mathrel{-\!\!*} Q)$

The affine modality

The **affine modality**:

$$\begin{aligned} \langle \mathit{affine} \rangle\, P &\triangleq P \wedge \mathsf{emp} \\ &\approx \text{“}P\text{ holds using just affine resources”} \end{aligned}$$

This modality is useful for reasoning about affinity

- Can be used to turn any proposition into an affine version, *e.g.*
  A wand that can be dropped $\langle \mathit{affine} \rangle\, (P \mathrel{-\!\!*} Q)$
- Commutes with most operators, *e.g.*
  $\langle \mathit{affine} \rangle\, (P \vee Q) \dashv\vdash \langle \mathit{affine} \rangle\, P \vee \langle \mathit{affine} \rangle\, Q$

The affine modality

The **affine modality**:

$$\langle \mathit{affine} \rangle \, P \quad \triangleq \quad P \wedge \mathsf{emp}$$
$$\approx \quad \text{“}P \text{ holds using just affine resources”}$$

This modality is useful for reasoning about affinity

- Can be used to turn any proposition into an affine version, *e.g.* A wand that can be dropped $\langle \mathit{affine} \rangle \, (P \mathrel{-\!\!*} Q)$
- Commutes with most operators, *e.g.* $\langle \mathit{affine} \rangle \, (P \vee Q) \dashv\vdash \langle \mathit{affine} \rangle \, P \vee \langle \mathit{affine} \rangle \, Q$
- Gives rise to an alternative classification of affine propositions $\mathsf{affine}(P) \quad \text{iff} \quad P \vdash \langle \mathit{affine} \rangle \, P$

The idea of carving out classes of propositions and defining their corresponding modalities is widely applicable:

- Persistent propositions $\boxdot$
- Intuitionistic propositions $\square$
- Absorbing propositions $\langle \mathit{absorb} \rangle$
- Timeless propositions (in step-indexed logics) $\triangleright, \diamond$
- Objective propositions (in iGPS) $\langle \mathit{obj} \rangle, \langle \mathit{subj} \rangle$
- Normal propositions (in CFML) $\langle \mathit{normal} \rangle$
- ...

The paper shows how to modularly deal with such classes and use them in general tactics

The idea of carving out classes of propositions and defining their corresponding modalities is widely applicable:

- Persistent propositions $\boxdot$
- Intuitionistic propositions $\square$
- Absorbing propositions $\langle absorb \rangle$
- Timeless propositions (in step-indexed logics) $\triangleright, \diamond$
- Objective propositions (in iGPS) $\langle obj \rangle, \langle subj \rangle$
- Normal propositions (in CFML) $\langle normal \rangle$
- ...

The paper shows how to modularly deal with such classes and use them in general tactics, we discuss two

Persistent and intuitionistic propositions

	Not droppable	**Droppable**
Not duplicable	Any	Affine
Duplicable	Persistent	Intuitionistic = (affine & persistent)

Persistent and intuitionistic propositions

	Not droppable	**Droppable**
Not duplicable	Any	Affine
Duplicable	Persistent	Intuitionistic = (affine & persistent)

Intuitionistic propositions:

- Widely used in practice, *e.g.* for reasoning about persistent (and garbage collected) data structures
- MoSeL has special support for them

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
    (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) -∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
  iIntros "[H1 #H2]".
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
  - by iFrame.
Qed.
```

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
    (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) −∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
```

Intuitionistic modality

```
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
  - by iFrame.
Qed.
```

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
    (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) -∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
  iIntros "[H1 #H2]".
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
  - by iFrame.
Qed.
```

```
1 subgoal
PROP : bi
A : Type
P, Q : PROP
Ψ : A → PROP
________________________(1/1)
"H2" : ∃ a : A, Ψ a
________________________□
"H1" : P
________________________∗
∃ a : A, Ψ a ∗ (P ∗ Ψ a)
```

Moves hypothesis to **intuitionistic** context

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
    (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) -∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
  iIntros "[H1 #H2]".
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
  - by iFrame.
Qed.
```

```
1 subgoal
PROP : bi
A : Type
P, Q : PROP
Ψ : A → PROP
x : A
______________________________________(1/1)
"H2" : Ψ x
--------------------------------------□
"H1" : P
--------------------------------------∗
∃ a : A, Ψ a ∗ (P ∗ Ψ a)
```

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
   (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) -∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
  iIntros "[H1 #H2]".
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
```

Do not need to split intuitionistic context

```
1 subgoal
PROP : bi
A : Type
P, Q : PROP
Ψ : A → PROP
x : A
______________________(1/1)
"H2" : Ψ x
______________________□
"H1" : P
______________________∗
x ∗ (P ∗ Ψ x)
```

Intuitionistic propositions in MoSeL

```
Lemma test {PROP : bi} {A}
    (P Q : PROP) (Ψ : A → PROP) :
  P ∗ □ (∃ a, Ψ a) -∗ ∃ a, Ψ a ∗ (P ∗ Ψ a).
Proof.
  iIntros "[H1 #H2]".
  iDestruct "H2" as (x) "H2".
  iExists x.
  iSplitL "H2".
  - iAssumption.
  - by iFrame.
Qed.
```

```
2 subgoals
PROP : bi
A : Type
P, Q : PROP
Ψ : A → PROP
x : A
______________________________________(1/2)
"H2" : Ψ x
--------------------------------------□
Ψ x

______________________________________(2/2)
"H2" : Ψ x
--------------------------------------□
"H1" : P
--------------------------------------∗
P ∗ Ψ x
```

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll} \Gamma & \text{Intuitionistic separation logic hypotheses} \\ \hline\hline \Pi & \text{Spatial separation logic hypotheses} \\ \hline R & \text{Separation logic goal} \end{array}$$

Γ Intuitionistic separation logic hypotheses
——————————————————— $\square$
Π Spatial separation logic hypotheses
———————————————————$*$
R Separation logic goal

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll} \Gamma & \text{Intuitionistic separation logic hypotheses} \\ \hline \Pi & \text{Spatial separation logic hypotheses} \\ \hline R & \text{Separation logic goal} \end{array}$$

Actual Coq goal (without pretty printing):

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

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll} \Gamma & \text{Intuitionistic separation logic hypotheses} \\ \hline \Pi & \text{Spatial separation logic hypotheses} \\ \hline R & \text{Separation logic goal} \end{array}$$

Actual Coq goal (without pretty printing):

$$\Gamma; \Pi \Vdash Q \quad \triangleq \quad \Box\left(\bigwedge \Gamma\right) * \left(\bigast \Pi\right) \vdash Q$$

Properties of the **intuitionistic context** Γ:

- Hypotheses can be duplicated (by tactics like `iSplitL`)

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll}
\Gamma & \text{Intuitionistic separation logic hypotheses} \\
\hline
\Pi & \text{Spatial separation logic hypotheses} \\
\hline
R & \text{Separation logic goal}
\end{array}$$

Actual Coq goal (without pretty printing):

$$\Gamma; \Pi \Vdash Q \quad \triangleq \quad \Box\left(\bigwedge \Gamma\right) * \left(\bigast \Pi\right) \vdash Q$$

Properties of the **intuitionistic context** Γ:

- Hypotheses can be duplicated (by tactics like `iSplitL`)
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll} \Gamma & \text{Intuitionistic separation logic hypotheses} \\ \hline \Pi & \text{Spatial separation logic hypotheses} \\ \hline R & \text{Separation logic goal} \end{array}$$

Actual Coq goal (without pretty printing):

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

Properties of the **intuitionistic context** Γ:

- Hypotheses can be duplicated (by tactics like `iSplitL`)
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
- Hypotheses can be moved to the spatial context Π (by tactics like `iAssumption`)

Formal treatment of the intuitionistic context

Visible Coq goal (with pretty printing):

$$\begin{array}{ll} \Gamma & \text{Intuitionistic separation logic hypotheses} \\ \hline \multicolumn{2}{r}{\square} \\ \Pi & \text{Spatial separation logic hypotheses} \\ \hline \multicolumn{2}{r}{*} \\ R & \text{Separation logic goal} \end{array}$$

Actual Coq goal (without pretty printing):

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

Properties of the **intuitionistic context** Γ:

- Hypotheses can be duplicated (by tactics like `iSplitL`)
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
- Hypotheses can be moved to the spatial context Π (by tactics like `iAssumption`)
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, ...

Where does the $\Box$ modality come from?

The properties of $\square$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, $\ldots$

The properties of $\Box$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
  $\Box P \dashv\vdash \Box P * \Box P$
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, $\ldots$

The properties of $\Box$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
  $\Box P \dashv\vdash \Box P * \Box P$
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
  $\Box P \vdash \mathsf{emp}$
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)

- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, ...

The properties of $\square$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
  $\square\, P \dashv\vdash \square\, P * \square\, P$
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
  $\square\, P \vdash \mathsf{emp}$
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)
  $\square\, P \vdash P$
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, $\ldots$

The properties of $\square$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
  $\square\, P \dashv\vdash \square\, P * \square\, P$
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
  $\square\, P \vdash \mathsf{emp}$
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)
  $\square\, P \vdash P$
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, ...
  $\square(\exists x.\, P) \vdash (\exists x.\, \square\, P)$ and $\square(\forall x.\, P) \vdash (\forall x.\, \square\, P)$

The properties of $\square$ that are needed

- Hypotheses can be duplicated (by tactics like `iSplitL`)
  $\square\, P \dashv\vdash \square\, P * \square\, P$
- Hypotheses can be dropped (by tactics like `iClear` and `iAssumption`)
  $\square\, P \vdash \mathsf{emp}$
- Hypotheses can be moved to the spatial context (by tactics like `iAssumption`)
  $\square\, P \vdash P$
- Closed under elimination of $\vee$, $\wedge$, $\exists$, $\forall$, ...
  $\square(\exists x.\, P) \vdash (\exists x.\, \square\, P)$ and $\square(\forall x.\, P) \vdash (\forall x.\, \square\, P)$

Problem: The $\square$ modality cannot be defined in terms of just the BI connectives for some logics (*e.g.* Iris) [Bizjak&Birkedal,MFPS'17]

We will extend BI logics with a new modality

Persistent and intuitionistic propositions

Recall:

	Not droppable	**Droppable**
Not duplicable	Any	Affine $\langle affine \rangle$
Duplicable	Persistent $\boxdot$	Intuitionistic = (affine & persistent) $\square$

Persistent and intuitionistic propositions

Recall:

	Not droppable	**Droppable**
Not duplicable	Any	Affine $\langle \mathit{affine} \rangle$
Duplicable	Persistent $\boxdot$	Intuitionistic $=$ (affine & persistent) $\square$

The intuitionistic modality $\square$

- Widely used in practice
- We do not know how to axiomatize it using a small set of axioms

Persistent and intuitionistic propositions

Recall:

	Not droppable	**Droppable**
Not duplicable	Any	Affine $\langle \mathit{affine} \rangle$
Duplicable	Persistent $\boxdot$	Intuitionistic = (affine & persistent) $\square$

The intuitionistic modality $\square$

- Widely used in practice
- We do not know how to axiomatize it using a small set of axioms

The persistent modality $\boxdot$

- Can be axiomatized using a small set of axioms
- Can be used to define the $\square$ modality and derive its laws

MoBIs: BIs with a persistence modality

The **persistence modality**:

$$\boxdot\, P \quad \triangleq \quad \text{“}P\text{ holds without ownership of non-exclusive resources”}$$

MoBI $\triangleq$ BI $+ \boxdot +$ additional axioms

MoBIs: BIs with a persistence modality

The **persistence modality**:

$$\boxdot\, P \quad \triangleq \quad \text{“}P \text{ holds without ownership of non-exclusive resources”}$$

MoBI $\triangleq$ BI $+\ \boxdot\ +$ additional axioms

- We can now define:

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

MoBIs: BIs with a persistence modality

The **persistence modality**:

$$\boxdot\, P \quad \triangleq \quad \text{“}P\text{ holds without ownership of non-exclusive resources”}$$

MoBI $\triangleq$ BI $+ \boxdot +$ additional axioms

- We can now define:

$$\begin{aligned} \text{persistent}(P) &\triangleq P \vdash \boxdot\, P \\ \text{intuitionistic}(P) &\triangleq P \vdash \square\, P \qquad\qquad \square\, P \triangleq \langle \mathit{affine} \rangle (\boxdot\, P) \end{aligned}$$

- The axioms are generalized from Iris's to general (non-affine) BI

$$\frac{P \vdash Q}{\boxdot\, P \vdash \boxdot\, Q} \qquad \begin{array}{c} \boxdot\, P \vdash \boxdot(\boxdot\, P) \\ \text{emp} \vdash \boxdot\, \text{emp} \end{array} \qquad \begin{array}{c} (\forall x.\ \boxdot\, P) \vdash \boxdot\,(\forall x.\ P) \\ \boxdot\,(\exists x.\ P) \vdash (\exists x.\ \boxdot\, P) \end{array} \qquad (\boxdot\, P) * Q \vdash \boxdot\, P$$

What's more in the MoSeL paper [Krebbers *et al.*,ICFP'18]?

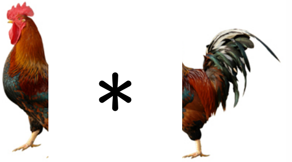
- Instantiations of MoSeL using 5 very different existing logics
  Iris, Fairis, iGPS, CFML, and CHL
- An Instantiation of MoSeL using a new logic
  Based on *ordered resource algebras*—a general model for MoBIs
- Semi-automated tactics using MoSeL for CFML and CHL
  For example, to support read-only permissions in CFML

Thank you!

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

Advertisement. I am currently looking for:

- A PhD student (4 years)
- A post-doc (1 year)



Topics: Separation logics for multilingual programs, asynchronous I/O, non-functional properties, verified compilation, ...

Interested/Know someone? Get in touch!