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 25, 2018 @ ICFP, St. Louis, United States

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]

Iris Proof Mode [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 [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 [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 [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 [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.
Qed.
```

The hypotheses for the left conjunct

```
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 [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 [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 [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.
```

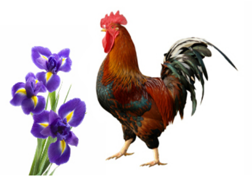
We can also solve this lemma automatically

```
No more subgoals.
```

The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

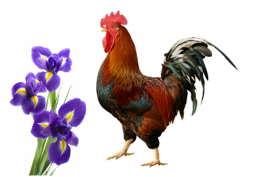
- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`



The **good things** about Iris Proof Mode

It enabled mechanized proofs in many papers because:

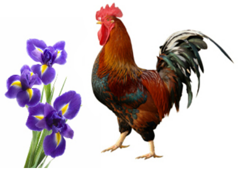
- **Proofs have the look and feel of ordinary Coq proofs**
  For many Coq tactics `tac`, it has a variant `iTac`
- **Support for advanced features of separation logic**
  Higher-order quantification, invariants, ghost state, later $\triangleright$ modality, ...



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
  (see also the next ICFP talk!)



The **bad thing** about Iris Proof Mode

The implementation is tied to Iris

~~Iris~~ Proof Mode

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`

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

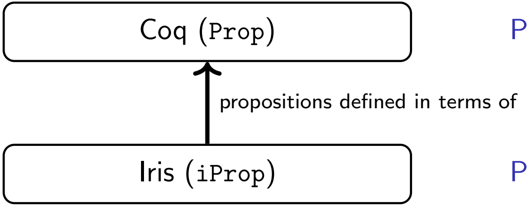
`iAssumption`

$$\Pi, P \Vdash P$$

Not having the affinity axiom is useful: precise accounting of resources

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, iGpsProp $\triangleq$ View $\xrightarrow{\text{mon}}$ 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, $\mathsf{iGpsProp} \triangleq \mathsf{View} \xrightarrow{\text{mon}} \mathsf{iProp}$

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, $\text{iGpsProp} \triangleq \text{View} \xrightarrow{\text{mon}} \text{iProp}$

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

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

Contributions

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

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 5 very different existing separation logics

CFML CHL Fairis iGPS Iris

Making MoSeL separation logic independent

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

Making MoSeL separation logic independent

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:

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

Making MoSeL separation logic independent

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:

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

```
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]]".
```

Lemma for another logic than Iris

```
    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.
```

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:

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

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**:

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

- 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 $\Box$
- 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

Thank you!

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

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 5 very different existing separation logics

CFML CHL Fairis iGPS Iris

Use MoSeL for your separation logic too!