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

Robbert Krebbers\textsuperscript{1}  Jacques-Henri Jourdan\textsuperscript{2}  Ralf Jung\textsuperscript{3}  Joseph Tassarotti\textsuperscript{4}
Jan-Oliver Kaiser\textsuperscript{3}  Amin Timany\textsuperscript{5}  Arthur Chaguéraud\textsuperscript{6}  Derek Dreyer\textsuperscript{3}

\textsuperscript{1}Delft University of Technology, The Netherlands
\textsuperscript{2}LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France
\textsuperscript{3}MPI-SWS, Germany
\textsuperscript{4}Carnegie Mellon University, USA
\textsuperscript{5}imec-Distrinet, KU Leuven, Belgium
\textsuperscript{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 $P \ast Q$:
The resources consists of separate parts satisfying $P$ and $Q$

Basic example:

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

the $\ast$ 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]
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
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
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]
Lemma test \{A\} (P Q : \text{iProp}) (Ψ : A \to \text{iProp}) :

\[ P \ast (\exists a, Ψ a) \ast Q \rightarrow Q \ast \exists a, P \ast Ψ a. \]

Proof.

\begin{itemize}
  \item iIntros "[H1 [H2 H3]]".
  \item iDestruct "H2" as (x) "H2".
  \item iSplitL "H3".
  \item - iAssumption.
  \item - iExists x.
  \item - iFrame.
\end{itemize}

Qed.
Lemma test \{A\} (P Q : iProp) (\Psi : A \rightarrow iProp) : 
P * (\exists a, \Psi a) * Q \rightarrow Q * \exists a, P * \Psi a.

Proof.
\begin{itemize}
  \item iIntros "[H1 \ [H2 H3]]".
  \item iDestruct "H2" as (x) "H2".
  \item iSplitL "H3".
    \begin{itemize}
      \item iAssumption.
      \item iExists x.
    \end{itemize}
  \item iFrame.
\end{itemize}
Qed.

Lemma in the Iris logic

The hypotheses for the left conjunct

resources should be split
Lemma test \(\{A\} (P Q : \text{iProp}) (\Psi : A \rightarrow \text{iProp}) : P \ast (\exists a, \Psi a) \ast Q \rightarrow Q \ast \exists a, P \ast \Psi a.\)

Proof.

iIntros "[H1 [H2 H3]]".
iDestruct "H2" as (x) "H2".
iSplitL "H3".
- iAssumption.
- iExists x.
iFrame.

Qed.
Lemma test \( \{A\} (P \quad Q : iProp) (\Psi : A \rightarrow iProp) : P \times (\exists \ a, \Psi \ a) \times Q \rightarrow Q \times \exists \ a, \ P \times \Psi \ a. \)

Proof.

\begin{align*}
i\text{Intros }"[H1 \ H2 \ H3]]". \\
i\text{Destruct }"H2" \text{ as } (x) "H2". \\
i\text{SplitL }"H3". \\
- \ i\text{Assumption}. \\
- \ i\exists\text{ists }x. \\
\quad i\text{Frame.}
\end{align*}

Qed.

1 subgoal
A : Type
P, Q : iProp
\Psi : A \rightarrow iProp
x : A

\begin{align*}
"H1" : P \\
"H2" : \Psi \ x \\
"H3" : Q
\end{align*}
**Lemma** test \{A\} (P Q : iProp) (Ψ : A → iProp) :

\[ P \ast (\exists a, Ψ a) \ast Q \rightarrow Q \ast \exists a, P \ast Ψ a. \]

**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

Q * (\exists a : A, P \ast Ψ a)

* means: resources should be split
Lemma test \( \{A\} (P \ Q : iProp) (\Psi : A \rightarrow iProp) : P \ast (\exists a, \Psi a) \ast Q \rightarrow Q \ast \exists a, P \ast \Psi a. \)

Proof.

iIntros "[H1 [H2 H3]]".
iDestruct "H2" as (x) "H2".
iSplitL "H3".
- iAssumption.
- iExists x.
iFrame.
Qed.

The hypotheses for the left conjunct

2 subgoals
A : Type
P, Q : iProp
\Psi : A \rightarrow iProp
x : A

\-----------(1/2)
"H3" : Q

\-----------*
Q

\-----------(2/2)
"H1" : P
"H2" : \Psi x

\-----------*
\exists a : A, P \ast \Psi a
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.
Lemma test \( \{ A \} \) (\( P \ Q : \text{iProp} \)) (\( \psi : A \rightarrow \text{iProp} \)) :
\( P * (\exists \ a, \psi \ a) * Q \rightarrow Q * \exists \ a, \ P * \psi \ a. \)

Proof.

iIntros "[H1 [H2 H3]]".
by iFrame.

Qed.

We can also solve this lemma automatically
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
  ▶ 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
  `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:

\[
\{ l_1 \mapsto v_1 * l_2 \mapsto v_2 \} \quad l_2 := ! l_1 \{ l_2 \mapsto v_1 \}
\]
Problem #1: Iris propositions are affine

In Iris you may “forget” about resources:

\[ \{ \ell_1 \mapsto v_1 \land \ell_2 \mapsto v_2 \} \ell_2 := ! \ell_1 \{ \ell_2 \mapsto v_1 \} \]

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

\[
\begin{align*}
\text{iClear} & \quad \Pi \vdash Q \\
\text{iAssumption} & \quad \Pi, P \vdash Q
\end{align*}
\]

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

Challenge: How to disentangle the affinity axiom from the Iris tactics?
Problem #1: Iris propositions are affine

In Iris you may “forget” about resources:

\[ \{ \ell_1 \mapsto v_1 \ast \ell_2 \mapsto v_2 \} \ell_2 := ! \ell_1 \{ \ell_2 \mapsto v_1 \} \]

Due to the affinity axiom \( P \ast Q \vdash Q \), which is hard-wired into many tactics:

- **iClear**
  \[
  \Pi \vdash Q \\
  \Pi, P \vdash Q \\
  \Pi, P \vdash Q
  \]

- **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

Coq (Prop)

Iris (iProp)

Proof using standard Coq tactics

Proof using Iris tactics

propositions defined in terms of
Problem #2: No tactical support for derived logics

Coq (Prop)

Proof using standard Coq tactics

Iris (iProp)

Proof using Iris tactics

Derived logic (e.g. iGpsProp)

propositions defined in terms of, $iGpsProp \triangleq \text{View} \xrightarrow{\text{mon}} iProp$
Problem #2: No tactical support for derived logics

\[
\begin{align*}
\text{Coq (Prop)} & \quad \text{Proof using standard Coq tactics} \\
\text{Iris (iProp)} & \quad \text{Proof using Iris tactics} \\
\text{Derived logic (e.g. iGpsProp)} & \quad \text{Proof using ???}
\end{align*}
\]

propositions defined in terms of

\[
i\text{GpsProp} \triangleq \text{View} \xrightarrow{\text{mon}} \text{iProp}
\]
Problem #2: No tactical support for derived logics

Coq (\text{Prop})

\text{propositions defined in terms of}

Iris (\text{iProp})

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

Derived logic (e.g. iGpsProp)

Proof using standard Coq tactics

Proof using Iris tactics

Proof using ???

\textbf{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 \((\text{Prop}, \vdash)\) with:

- Operations \textbf{True}, \textbf{False}, \& (\land), \lor (\lor), \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 \((\text{Prop}, \vdash)\) with:

- Operations \(\text{True}, \text{False}, \land, \lor, \Rightarrow, \forall, \exists\) satisfying the axioms of intuitionistic logic
- Operations \(\text{emp}, *, -*\) satisfying:

\[
\begin{align*}
\text{emp} * P \nvdash P \\
P * Q \vdash Q * P \\
(P * Q) * R \vdash P * (Q * R) \\
\end{align*}
\]

\[
\begin{align*}
P_1 \vdash Q_1 & \quad P_2 \vdash Q_2 \quad P * Q \vdash R \\
P_1 * P_2 \vdash Q_1 * Q_2 \quad P \vdash Q \nvdash R
\end{align*}
\]
Making MoSeL separation logic independent

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

- Operations **True, False, \&\&, ||, \forall, \exists** satisfying the axioms of intuitionistic logic
- Operations **emp, *, →** satisfying:

\[
\begin{align*}
\text{emp} \ast P & \not\vdash P \\
P \ast Q & \vdash Q \ast P \\
(P \ast Q) \ast R & \vdash P \ast (Q \ast R)
\end{align*}
\]

\[
\begin{align*}
P_1 \vdash Q_1 & \quad P_2 \vdash Q_2 \\
P_1 \ast P_2 \vdash Q_1 \ast Q_2 \\
P \vdash Q \not\ast R
\end{align*}
\]

**Structure**

\[
\text{bi} := \text{Bi} \{ \\
\quad \text{bi}_\text{car} :> \text{Type}; \\
\quad \text{bi}_\text{entails} : \text{bi}_\text{car} \rightarrow \text{bi}_\text{car} \rightarrow \text{Prop}; \\
\quad \text{bi}_\text{forall} : \forall A, (A \rightarrow \text{bi}_\text{car}) \rightarrow \text{bi}_\text{car}; \\
\quad \text{bi}_\text{sep} : \text{bi}_\text{car} \rightarrow \text{bi}_\text{car} \rightarrow \text{bi}_\text{car}; \\
\quad (* \text{other separation logic operators and axioms} *)
\}
\]
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 * (Ψ a, ∃ 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 * (Ψ a, ∃ a, P * Ψ a).
Proof.
  iIntros "[H1 [H2 H3]]".
  iDestruct "H2" as (x) "H2".
  iSplitL "H3".
  - iAssumption.
  - iExists x.
  iFrame.
Qed.

Lemma for another logic than Iris
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]]\".
  Destruct "H2" as (x) "H2".
  SplitL "H3".
  − iAssumption.
  − iExists x.
  iFrame.

Lemma for another logic than Iris

Lemma universally quantified in the BI logic
Addressing challenge #1:
Disentangling the affinity axiom

\[ P \ast Q \vdash Q \]
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:

  \[
  \text{GC locations (affine)} \quad \ell \mapsto_{\text{gc}} v
  \]
  \[
  \text{Non-GC locations (not affine)} \quad \ell \mapsto v
  \]

  (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:

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

The new tactics:

(iClear)

\[
\Pi \not\vdash Q \quad \text{affine}(P)
\]

\[
\Pi, P \vdash Q
\]

(iAssumption)

\[
\text{affine}(\Pi)
\]

\[
\Pi, Q \vdash Q
\]
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**: 

\[ \langle \text{affine} \rangle P \triangleq P \land \text{emp} \]

\[ \approx \ “P \text{ holds using just affine resources”} \]

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

- Commutes with most operators, *e.g.*
  \[ \langle \text{affine} \rangle (P \lor Q) \vdash \langle \text{affine} \rangle P \lor \langle \text{affine} \rangle Q \]

- Gives rise to an alternative classification of affine propositions
  \( \text{affine}(P) \) iff \( P \vdash \langle \text{affine} \rangle P \)
The idea of carving out classes of propositions and defining their corresponding modalities is widely applicable:

- Persistent propositions
- Intuitionistic propositions
- Absorbing propositions \( \langle \text{absorb} \rangle \)
- Timeless propositions (in step-indexed logics) \( \triangledown, \lozenge \)
- Objective propositions (in iGPS) \( \langle \text{obj} \rangle, \langle \text{subj} \rangle \)
- Normal propositions (in CFML) \( \langle \text{normal} \rangle \)
- ...

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!