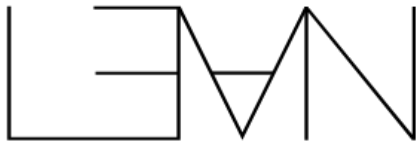
Iris Lean

Zongyuan Liu, Aarhus University and Michael Sammler, ISTA
For the Iris Lean community

June 9th, 2026

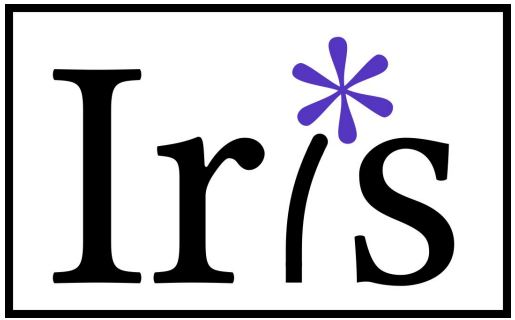
What is Lean?

- Theorem prover launched by Leonardo de Moura in 2013
- Constantly improved by a team of full-time engineers at Lean FRO
- Comprehensive math library - Mathlib
- Same dependent type theory as Rocq

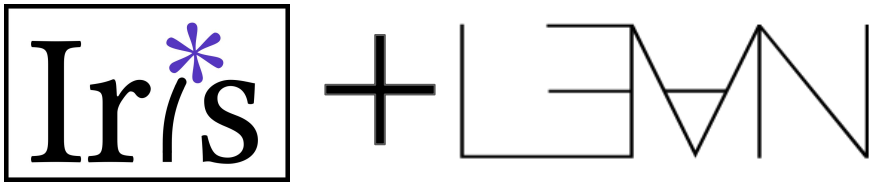


What is Iris?

(just joking)



Iris Lean is a port of *Iris* to *Lean*



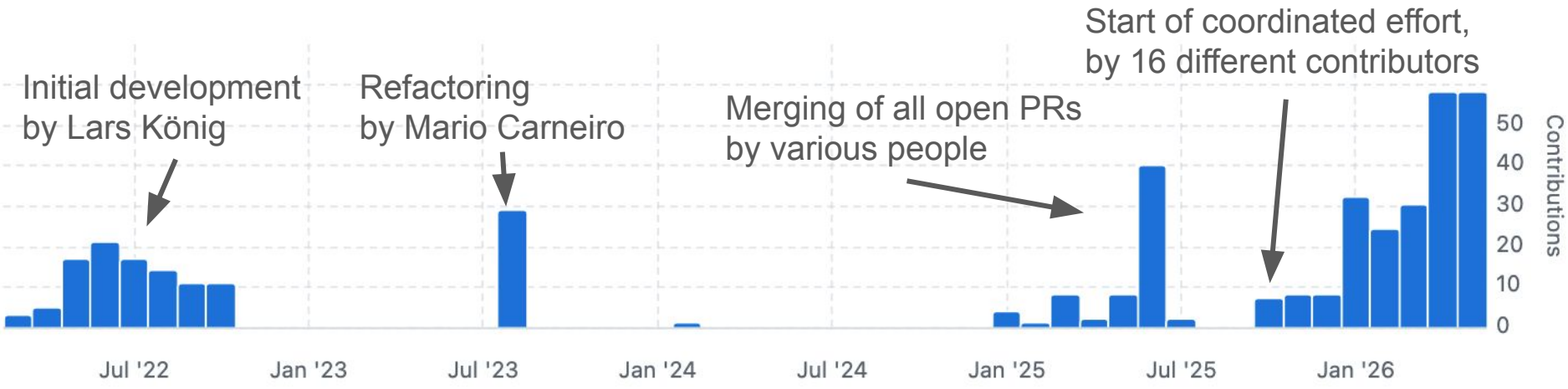
Why port Iris to Lean?

Past, present, and future of Iris Lean

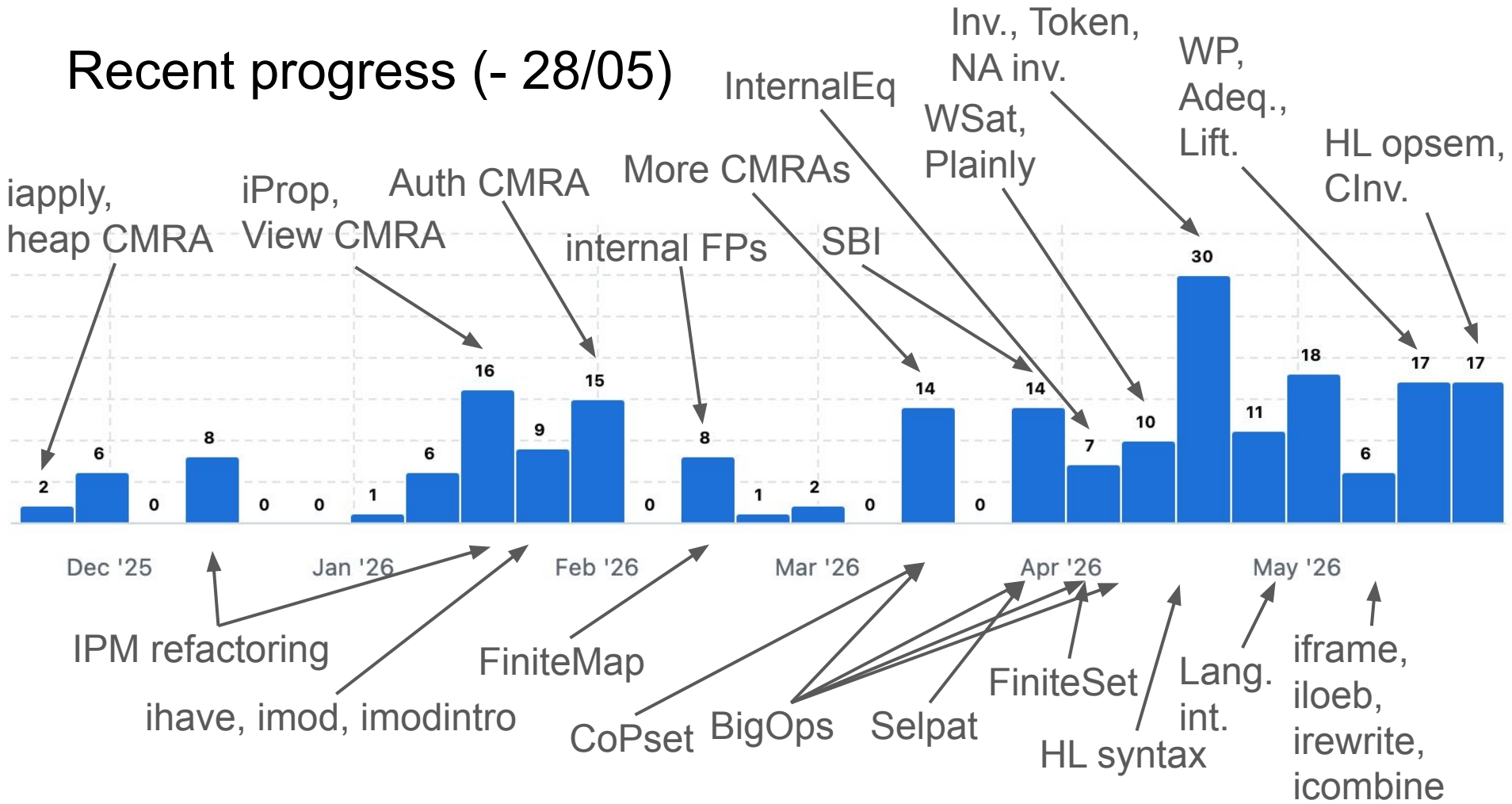
Past:
Brief History of Iris Lean

Brief history of Iris Lean

- Master thesis of Lars König in 2022
- Maintained by Mario Carinero until 2025
- Markus de Meideros becomes interested in 2025
- More and more people become interested and start working on it in late 2025
- Porting effort significantly picking up steam at beginning of 2026



Recent progress (- 28/05)



Present:
Current Status of Iris Lean

Contributors to Iris Lean


Markus de Medeiros


Michael Sammler


Fernando Leal


Oliver Soeser


Remy Seassau


Quang Dao


Alok Singh


Viet Anh Nguyen


Lars König


Zongyuan Liu


suhr


Alex Bai


Léo Stefanesco


Seng Joe Watt


Haokun Li


Mario Carneiro


Sergei Stepanenko


Shreyas Srinivas


Yunsong Yang


Puming Liu


GenericMonkey


Alvin Tang

Porting status

Tracking webpage: https://leanprover-community.github.io/iris-lean/

Main missing features

- monotone predicates / embedding
- Some CMRAs (e.g. STS)
- Some base_logic libraries
  (e.g. saved prop, boxes)
- Atomic updates / triples
- Total weakest precondition
- iinv (+ more seldomly used tactics like iaccu)

Iris Lean in Action

Benefits of Lean

- Automation for pure goals/lemmas

```
· ipureintro
  have : ls2'.all (head < ·) := by grind
  grind [List.pairwise_cons]
· ipureintro
  grind [List.filter_append_perm]
```

- More control in implementing tactics
  - Custom `ProofModeM` monad for writing tactics
  - Custom typeclass synthesis algorithm
- No strings in proof mode tactics

Design choices

- More axioms / classical logic
  - `UPred` tweaked to become `Leibniz` (because of `propext`, `funext`)

```
theorem IProp.ext {P Q : IProp GF} : P ⊣⊢ Q → P = Q
```

- Typeclasses instead of bundled hierarchies / canonical structures

```
class IsCOFE (α : Type _) [OFE α] where
 compl : Chain α → α
 conv_compl {c : Chain α} : compl c ≡{n}≡ c n
```

New challenges

Problem	Solution
Restrictive typeclass synthesis	Implement custom synthesis to simulate Rocq TC
Less comprehensive batteries and no mathlib dependency	Upstream from mathlib or implement with minor generalization, e.g. FiniteMap
Missing generalized rewriting	Use mathlib's `grw` (?)
PRs with fully AI generated code	AI reviews (?)

Projects based on Iris Lean

Approxis-based relational logic for real-valued programs	Markus | Mathlib
iaesop - `aesop` white box automation for Iris	Yunsong | Metaprogramming, automation
Lithium-based translation validation for LLVM IR optimization	Zongyuan | Metaprogramming, AI
Lilac program logic	Edwin Fernando | Mathlib

Relationship between Iris Lean and Iris Rocq

In principle following Rocq implementation

Not split the community, both should be compatible

- Track correspondence with `rocq_alias`
- Nightly CI to monitor renaming in Rocq
- Still get the benefits of Lean

Relationship between Iris Lean and Iris Rocq

```
@[rocq_alias cmra]
class CMRA (α : Type _) extends OFE α where
 pcore : α → Option α
 op : α → α → α
 ValidN : α → Prop
 pcore_op_mono : pcore x = some cx → ∀ y, ∃ cy, pcore
(op x y) ≡ some (op cx cy)
 validN_op_left : ValidN n (op x y) → ValidN n x
 extend : ValidN n x → x ≡{n}≡ op y₁ y₂ →
   Σ' z₁ z₂, x ≡ op z₁ z₂ ∧ z₁ ≡{n}≡ y₁ ∧ z₂ ≡{n}≡ y₂
 -- ...

#rocq_ignore Op "Use the CMRA.op field."
#rocq_ignore PCore "Use the CMRA.pcore field."
#rocq_ignore Valid "Use the CMRA.Valid field."
#rocq_ignore CmraMixin "Use the CMRA type class."
#rocq_ignore cmra_ofeO "Not needed."
```

Lean v.s. Rocq

```
Record CmraMixin {SI : sidx} A
     `{!Dist A, !Equiv A, !PCore A, !Op A, !Valid A, !ValidN A} := {
   mixin_cmra_pcore_mono (x y : A) cx :
     x ≼ y → pcore x = Some cx → ∃ cy, pcore y = Some cy ∧ cx ≼ cy;
   mixin_cmra_validN_op_l n (x y : A) : ✓{n} (x · y) → ✓{n} x;
   mixin_cmra_extend n (x y1 y2 : A) :
     ✓{n} x → x ≡{n}≡ y1 · y2 →
     { z1 : A & { z2 | x ≡ z1 · z2 ∧ z1 ≡{n}≡ y1 ∧ z2 ≡{n}≡ y2 } }
   (* ... *) }.
Structure cmra {SI : sidx} := Cmra' {
 cmra_car :> Type;
 cmra_pcore : PCore cmra_car;
 cmra_op : Op cmra_car;
 cmra_validN : ValidN cmra_car;
 cmra_ofe_mixin : OfeMixin cmra_car;
 cmra_mixin : CmraMixin cmra_car;
 (* ... *) }.
Coercion cmra_ofeO {SI : sidx} (A : cmra) : ofe := Ofe A (cmra_ofe_mixin A).
Canonical Structure cmra_ofeO.
```

Future:
Next steps

Future directions

Further leverage Lean metaprogramming for Iris automation

Combination of Iris and mathlib

Connecting to Lean's `mvcgen`

Explore integration with Lean-based AI efforts

Relationship to CSlib

Thanks to the community ❤️

Alex Bai
Alok Singh
Alvin Tang
Fernando Leal
Haokun Li
Joseph Tassarotti
Lars Birkedal
Lars König
Léo Stefanesco
Mario Carneiro
Markus de Medeiros
Max Vistrup
Michael Sammler
Oliver Soeser
Puming Liu
Quang Dao
Ralf Jung
Remy Seassau
Seng Joe Watt
Sergei Stepanenko
Shreyas Srinivas
Viet Anh Nguyen
Yunsong Yang
Zongyuan Liu
@suhr
@GenericMonkey

If you are curious about Iris Lean

- Join the Iris Lean Zulip channel on the Lean Zulip
- We have bi-weekly meetings, let us know if you are interested in joining