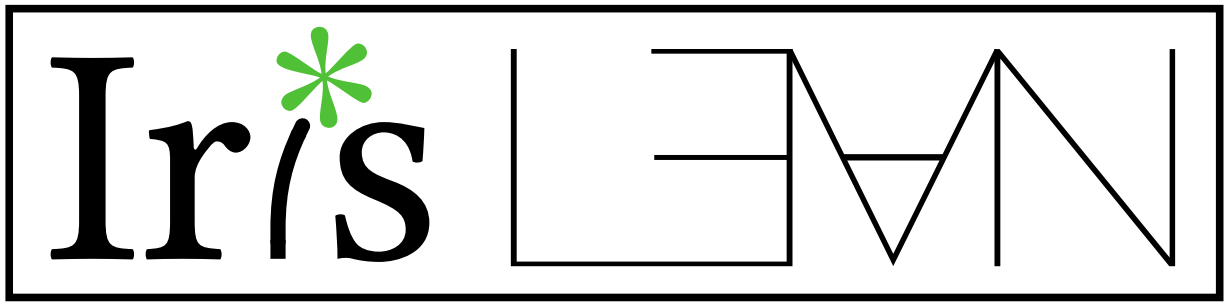
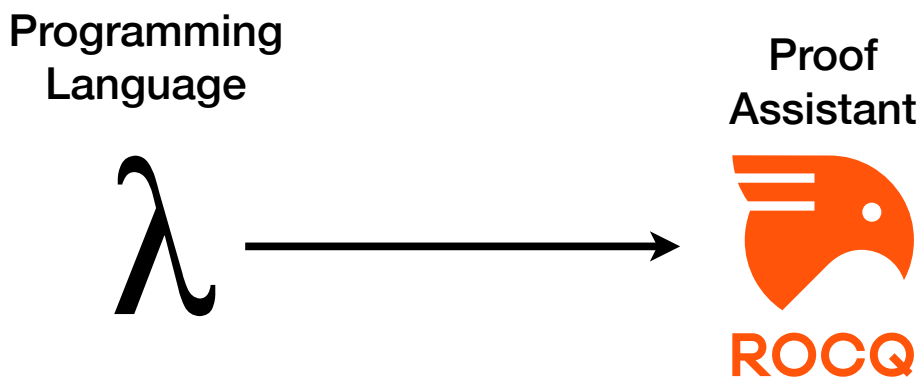


Markus de Medeiros

NYU

NJPLS, December 5th 2025


Programming
Language
λ
Proof
Assistant
ROCQ

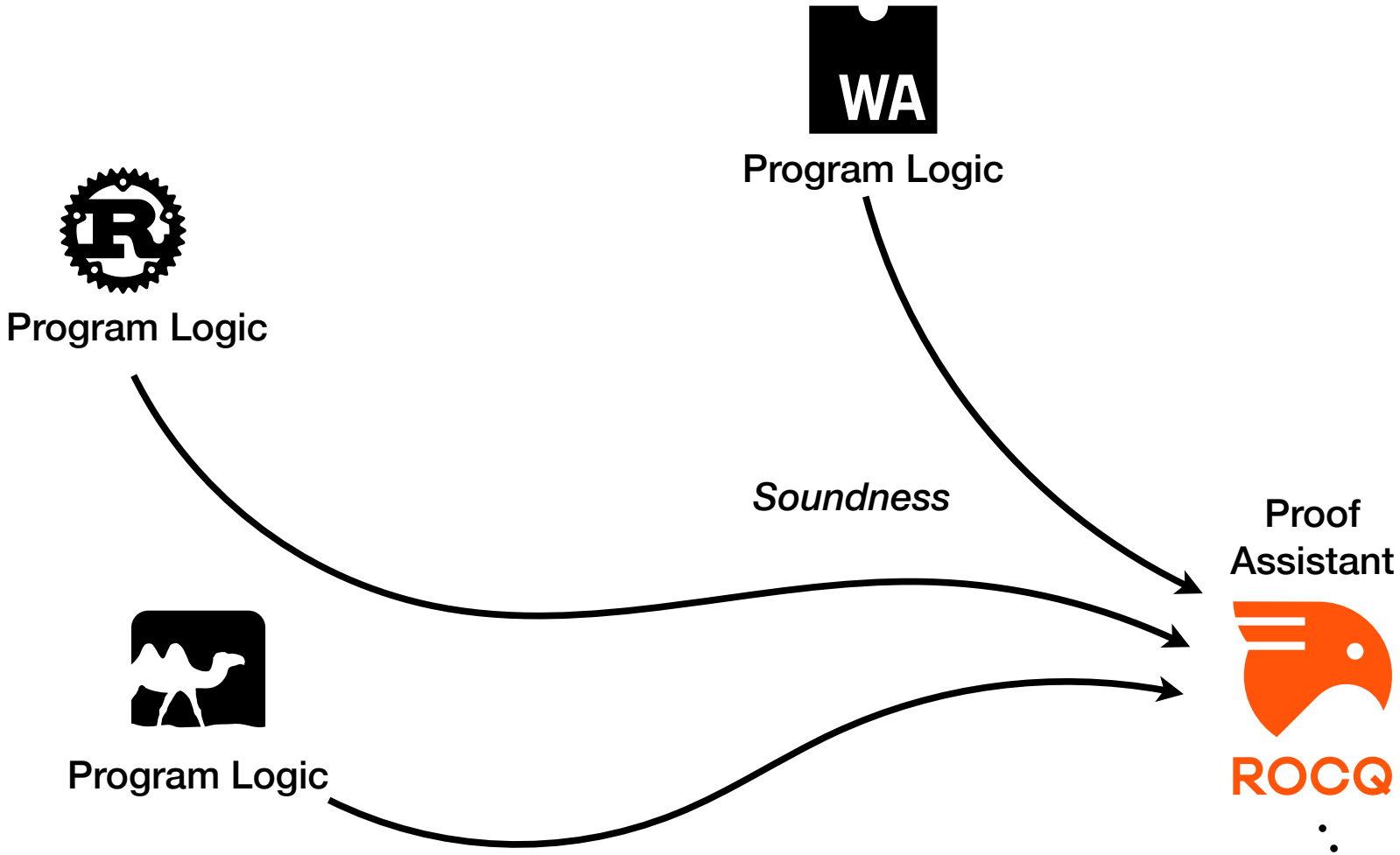
Program Logic

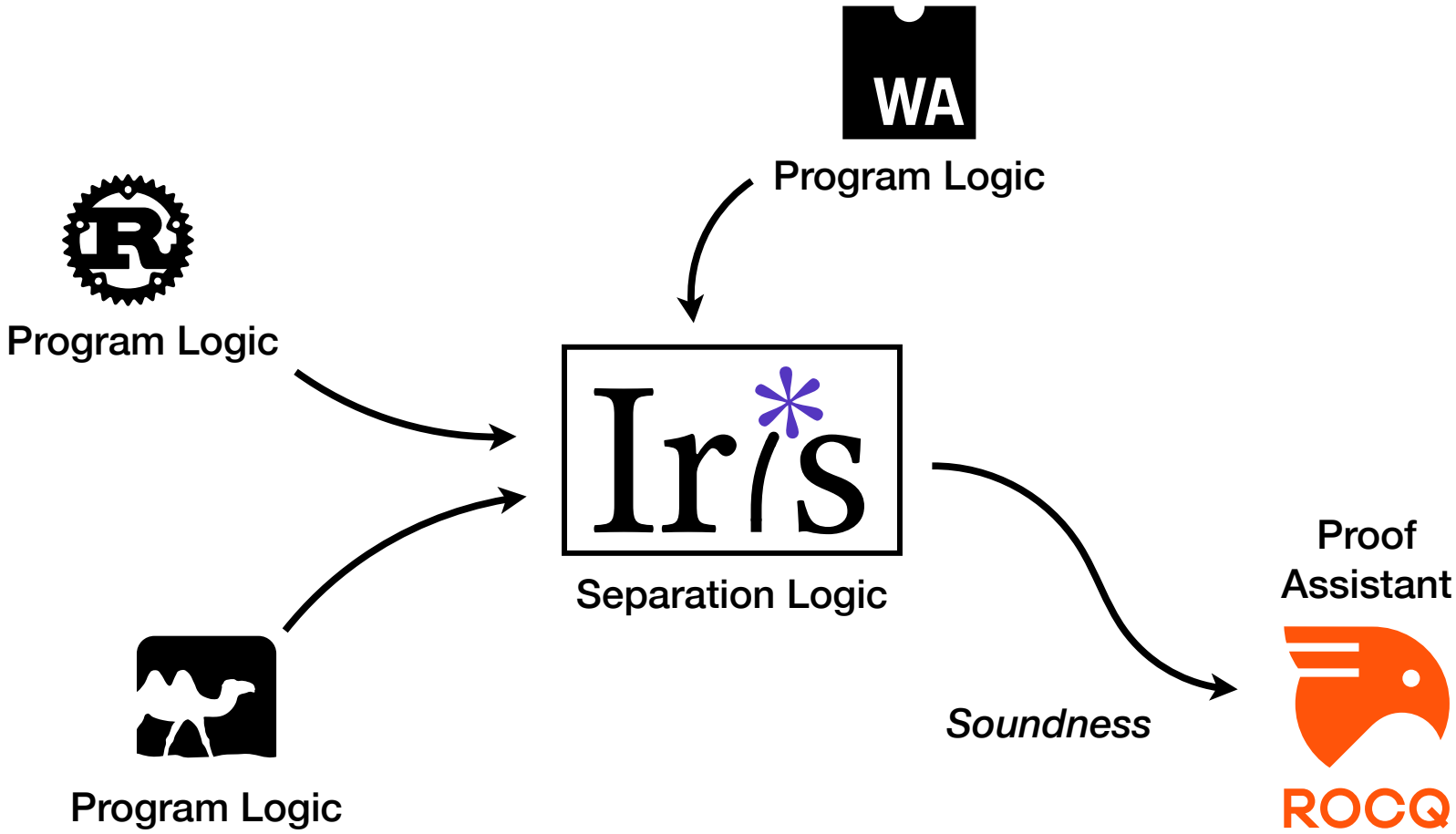
Program Logic

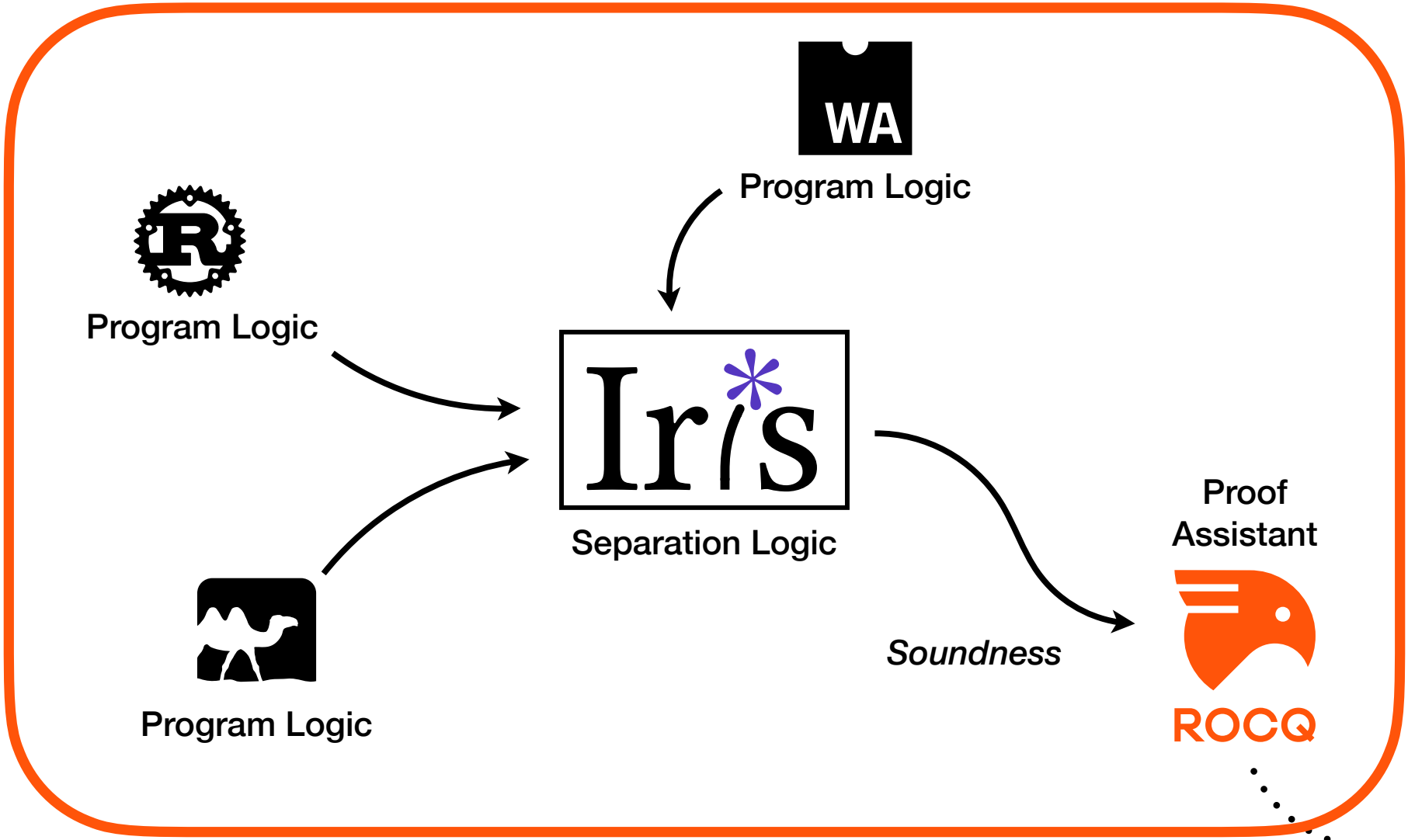
Program Logic

Proof
Assistant

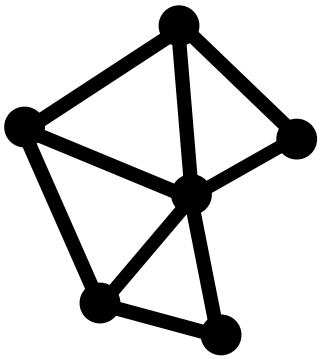
ROCQ


WA
Program Logic
Program Logic
Soundness
Proof
Assistant
Program Logic
ROCQ

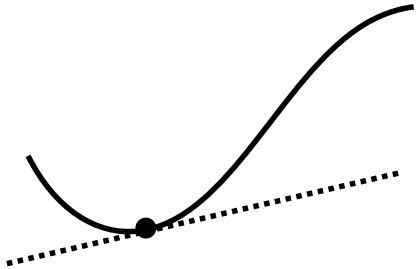

WA
Program Logic
Program Logic
Iris
Separation Logic
Program Logic
Soundness
Proof
Assistant
ROCQ


WA
Program Logic
Program Logic
Iris
Separation Logic
Program Logic
Soundness
Proof
Assistant
ROCQ

Graph Theory

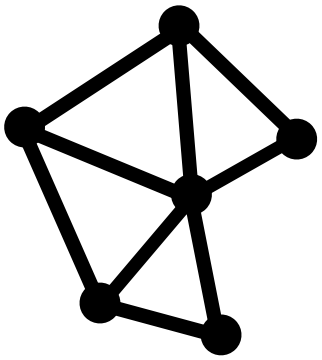


Calculus

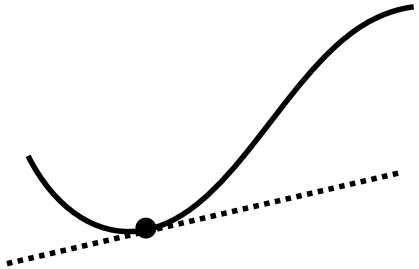


ROCQ

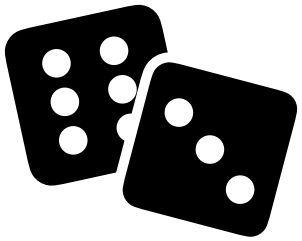
Graph Theory



Calculus



Probability



ROCQ

Formalized Math

	Coquelicot*	2,000	Theorems	40,000	LOC
	Mathcomp-analysis + Mathcomp*	22,000	Theorems	182,000	LOC
	Mathlib[1]	243,000	Theorems	2,000,000	LOC
	Archive of Formal Proofs[2]	300,000	Theorems	2,500,000	LOC
	Iris*	4,800	Theorems	53,000	LOC

1 https://www.isa-afp.org/statistics/
2 https://leanprover-community.github.io/mathlib_stats.html
** Unofficial Estimate (grep)*

Many other projects not included!

Formalized Math

ROCQ	Coquelicot*	2,000	Theorems	40,000	LOC
ROCQ	Mathcomp-analysis + Mathcomp*	22,000	Theorems	182,000	LOC
LEAN THEOREM PROVER	Mathlib[1]	243,000	Theorems	2,000,000	LOC
Isabelle	Archive of Formal Proofs[2]	300,000	Theorems	2,500,000	LOC
Iris	Iris*	4,800	Theorems	53,000	LOC

1 https://www.isa-afp.org/statistics/
2 https://leanprover-community.github.io/mathlib_stats.html
** Unofficial Estimate (grep)*

Many other projects not included!

Formalized Math

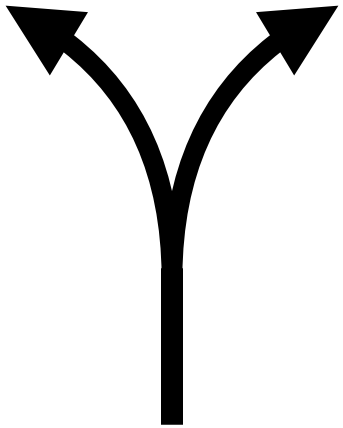
	Coquelicot*	2,000	Theorems	40,000	LOC
	Mathcomp-analysis + Mathcomp*	22,000	Theorems	182,000	LOC
	Mathlib[1]	243,000	Theorems	2,000,000	LOC
	Archive of Formal Proofs[2]	300,000	Theorems	2,500,000	LOC
	Iris*	4,800	Theorems	53,000	LOC

1 https://www.isa-afp.org/statistics/
2 https://leanprover-community.github.io/mathlib_stats.html
** Unofficial Estimate (grep)*

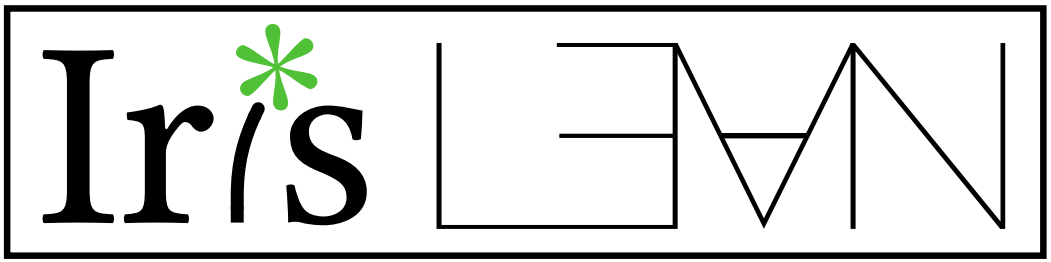
Many other projects not included!

A Crossroads

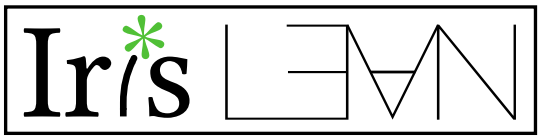
Math in ROCQ

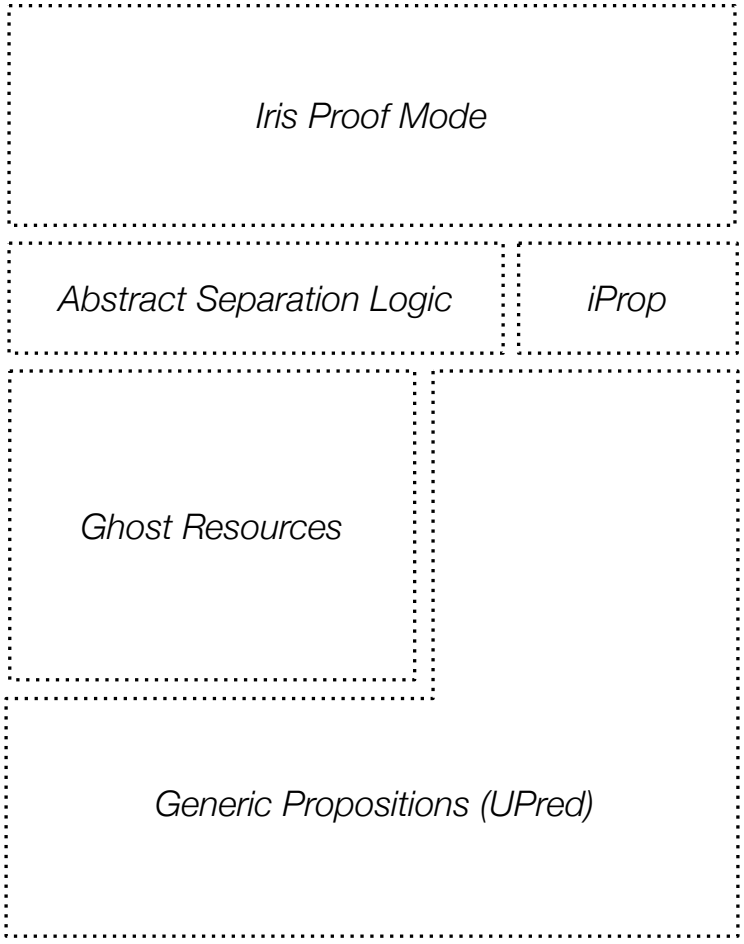


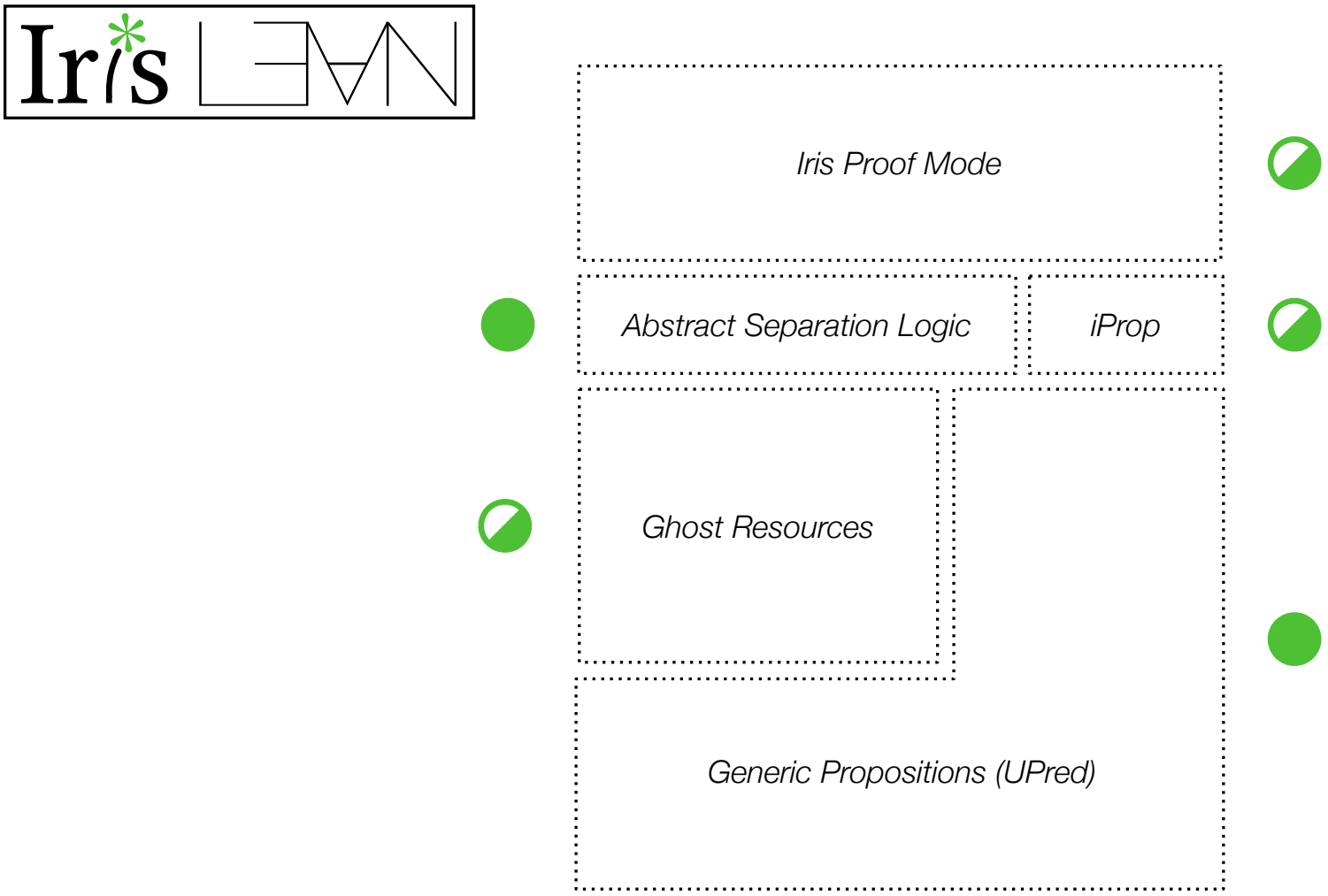
Iris in ?

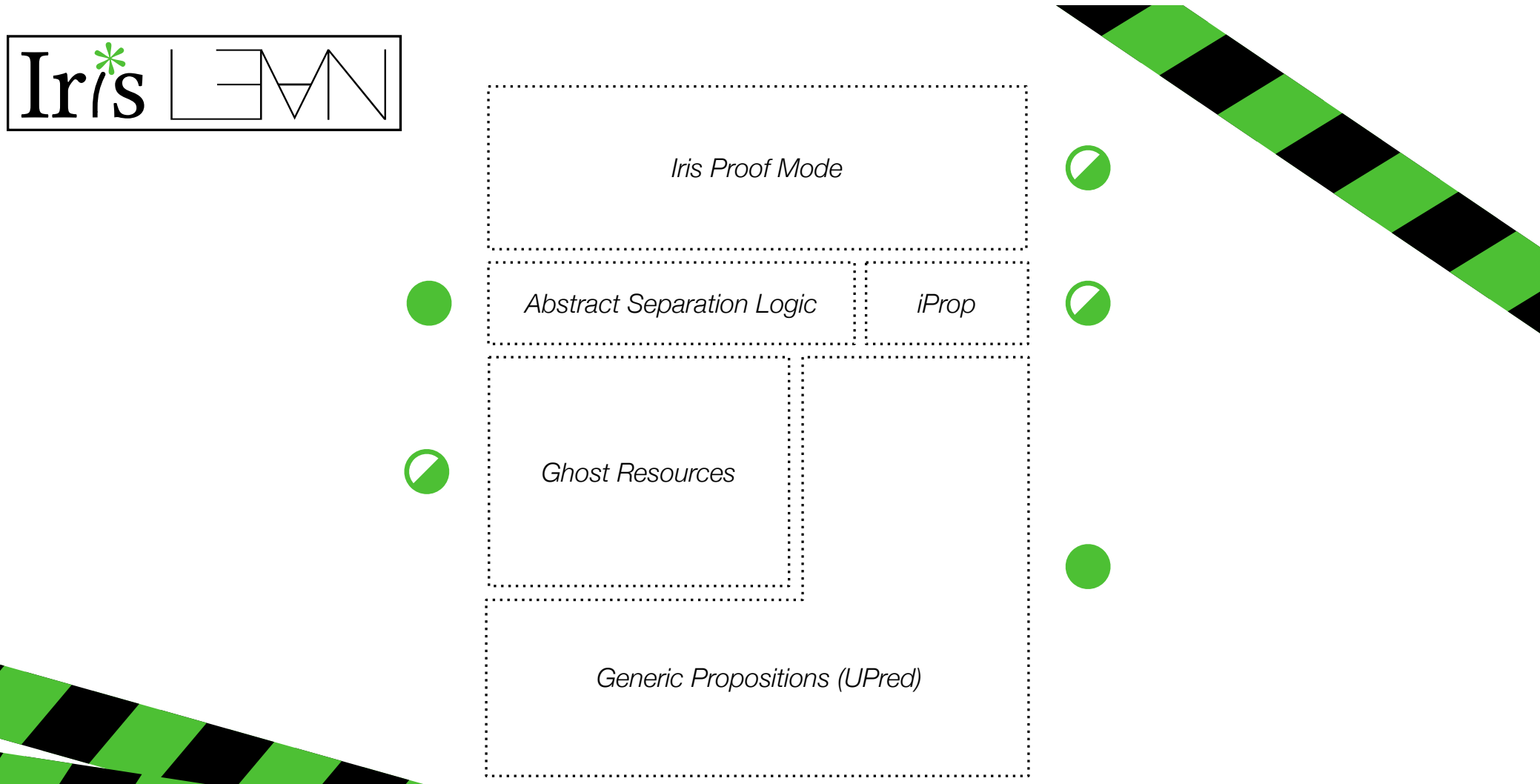


- Possible in Lean?
- Iris-Lean adaptations?

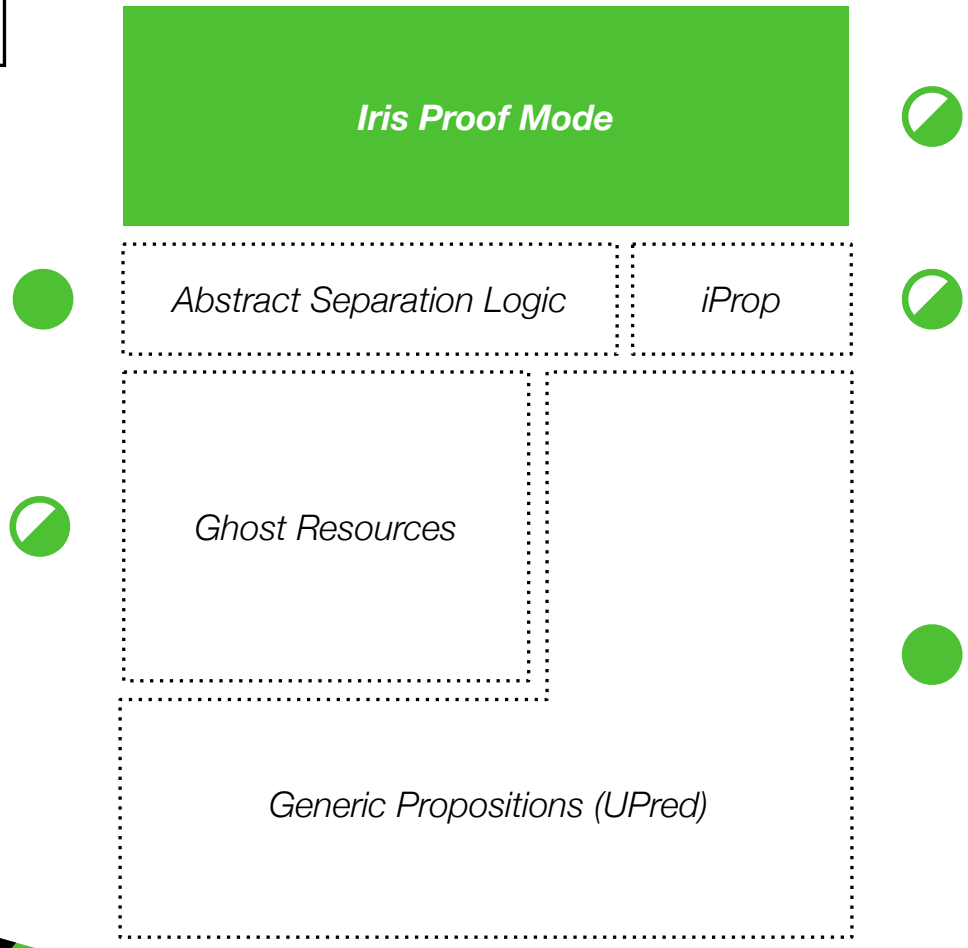

Iris

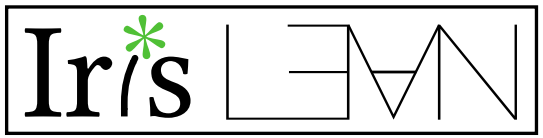

Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)


Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)


Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)


Iris


Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)



Iris Proof Mode

Karlsruher Institut für Technologie

Institut für Programmstrukturen
und Datenorganisation (IPD)

Lehrstuhl Prof. Dr.-Ing. Snelting

An Improved Interface for Interactive Proofs in Separation Logic

Masterarbeit von

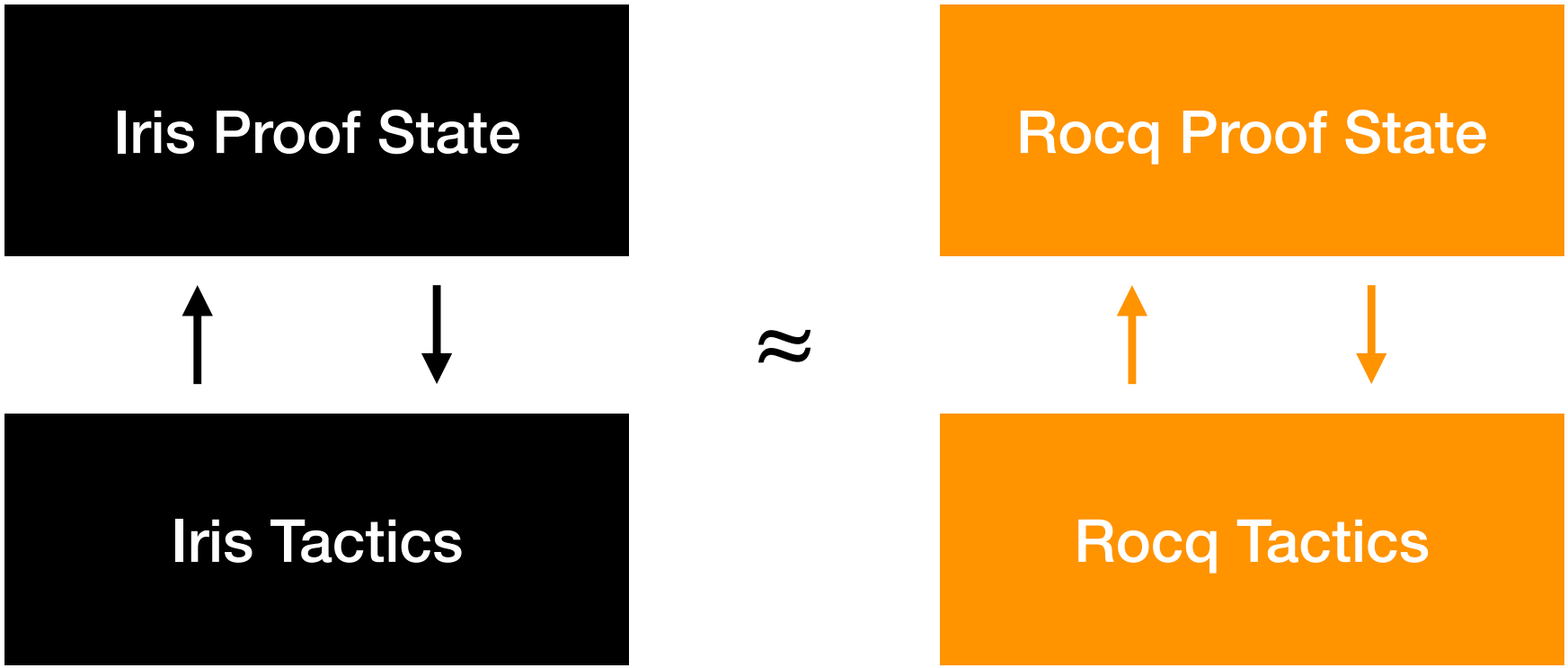
Lars König

an der Fakultät für Informatik

```
theorem proof_example [BI PROP] (P Q R : PROP) (Φ : α → PROP) :
    P ∗ Q ∗ □ R ⊢ □ (R -∗ ∃ x, Φ x) -∗ ∃ x, Φ x ∗ P ∗ Q
:= by                                   Iris Proof Mode
    iintro ⟨HP, HQ, #HR⟩ #HRΦ           -------------------------
    ispecialize HRΦ HR as HΦ            HR : R
    icases HΦ with ⟨x, HΦ⟩              HRΦ : R -∗ ∃ x, Φ x
    iexists x                           HΦ : ∃ x, Φ x
    isplit r                            ------------------------- □
    · iassumption                       HP : P
    isplit l [HP]                       HQ : Q
    · iexact HP                         ------------------------- ∗
    · iexact HQ                         ∃ x, Φ x ∗ P ∗ Q
```

Erstgutachter:	Prof. Dr.-Ing. Gregor Snelting
Zweitgutachter:	Prof. Dr. rer. nat. Bernhard Beckert
Betreuender Mitarbeiter:	M. Sc. Sebastian Ullrich
Abgabedatum:	30. September 2022

Iris Proof Mode



Iris Proof Mode

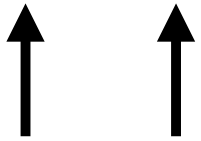
```
Theorem sep_mp {P Q : iProp Σ} :
  ⊢ P ∗ (P -∗ Q) -∗ Q.
Proof.
  iIntros "[HP Hwand]".
  iApply "Hwand".
  iExact "HP".
Qed.
```

```
P, Q : iProp Σ
————————————————————————————————
P ∗ (P -∗ Q) -∗ Q
```

Iris Proof Mode

```
Theorem sep_mp {P Q : iProp Σ} :
  ⊢ P ∗ (P -∗ Q) -∗ Q.
Proof.
  iIntros "[HP Hwand]".
  iApply "Hwand".
  iExact "HP".
Qed.
```

```
P, Q : iProp Σ
————————————————————————————
P ∗ (P -∗ Q) -∗ Q
```



Iris Proof Mode

```
Theorem sep_mp {P Q : iProp Σ} :
  ⊢ P ∗ (P -∗ Q) -∗ Q.
Proof.
  iIntros "[HP Hwand]".
  iApply "Hwand".
  iExact "HP".
Qed.
```

```
P, Q : iProp Σ
──────────────────────────────────
  "HP" : P
  "Hwand" : P -∗ Q
  ──────────────────────────────∗
  Q
```

Iris Proof Mode

```
Theorem sep_mp {P Q : iProp Σ} :
  ⊢ P ∗ (P -∗ Q) -∗ Q.
Proof.
  iIntros "[HP Hwand]".
  iApply "Hwand".
  iExact "HP".
Qed.
```

```
P, Q : iProp Σ
──────────────────────────────
  "HP" : P
 ──────────────────────────∗
  P
```

Iris Proof Mode

```
Theorem sep_mp {P Q : iProp Σ} :
  ⊢ P ∗ (P -∗ Q) -∗ Q.
Proof.
  iIntros "[HP Hwand]".
  iApply "Hwand".
  iExact "HP".
Qed.
```

Iris Proof Mode

Iris Proof State

```
"HP" : P
"Hwand" : P -* Q
————————————————————*
Q
```

Iris Tactics

```
iIntros "[HP Hwand]".
iApply "Hwand".
iExact "HP".
```

Iris Proof Mode

Iris Proof State

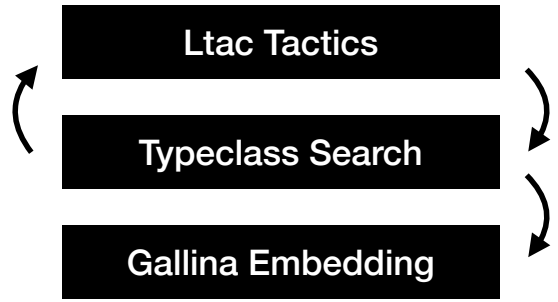
```
"HP" : P
"Hwand" : P -* Q
————————————————————*
Q
```

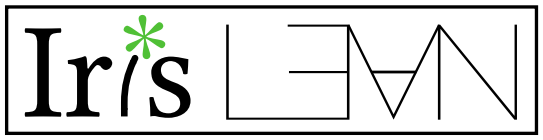
Iris Tactics

```
iIntros "[HP Hwand]".
iApply "Hwand".
iExact "HP".
```

Rocq Complexity

```
envs_entails {|
  env_intuitionistic := Enil;
  env_spatial := Esnoc (Esnoc Enil
    "HP" P)
    "Hwand" (bi_wand P Q);
   env_counter := 2%positive |}
Q
```





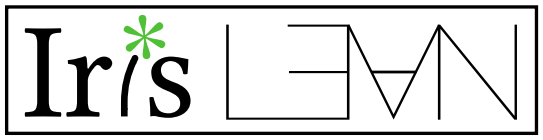
Iris Proof Mode

Iris Proof State

```
*HP : P
*Hwand : P -* Q
  ⊢ Q
```

Iris Tactics

```
iintros ⟨HP, Hwand⟩
iapply Hwand
iexact HP
```



Iris Proof Mode

Iris Proof State

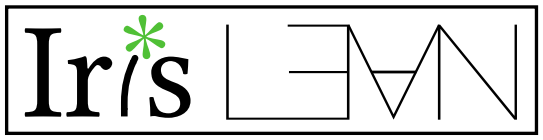
```
*HP : P
*Hwand : P -* Q
  ⊢ Q
```

Iris Tactics

```
iintros ⟨HP, Hwand⟩
iapply Hwand
iexact HP
```

```
⊢ Entails (sep P (wand P Q))) Q
```

\+ Simpler encoding



Iris Proof Mode

Iris Proof State

```
*HP : P
*Hwand : P -* Q
  ⊢ Q
```

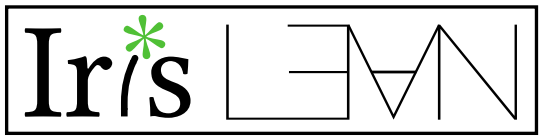
```
⊢ Entails (sep P (wand P Q))) Q
```

+ Simpler encoding

Iris Tactics

```
iintros ⟨HP, Hwand⟩
iapply Hwand
iexact HP
```

+ Monadic, imperative metaprograms



Iris Proof Mode

Iris Proof State

```
*HP : P
*Hwand : P -* Q
  ⊢ Q
```

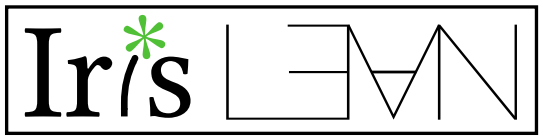
Iris Tactics

```
iintros ⟨HP, Hwand⟩
iapply Hwand
iexact HP
```

```
⊢ Entails (sep P (wand P Q))) Q
```

- \+ Simpler encoding

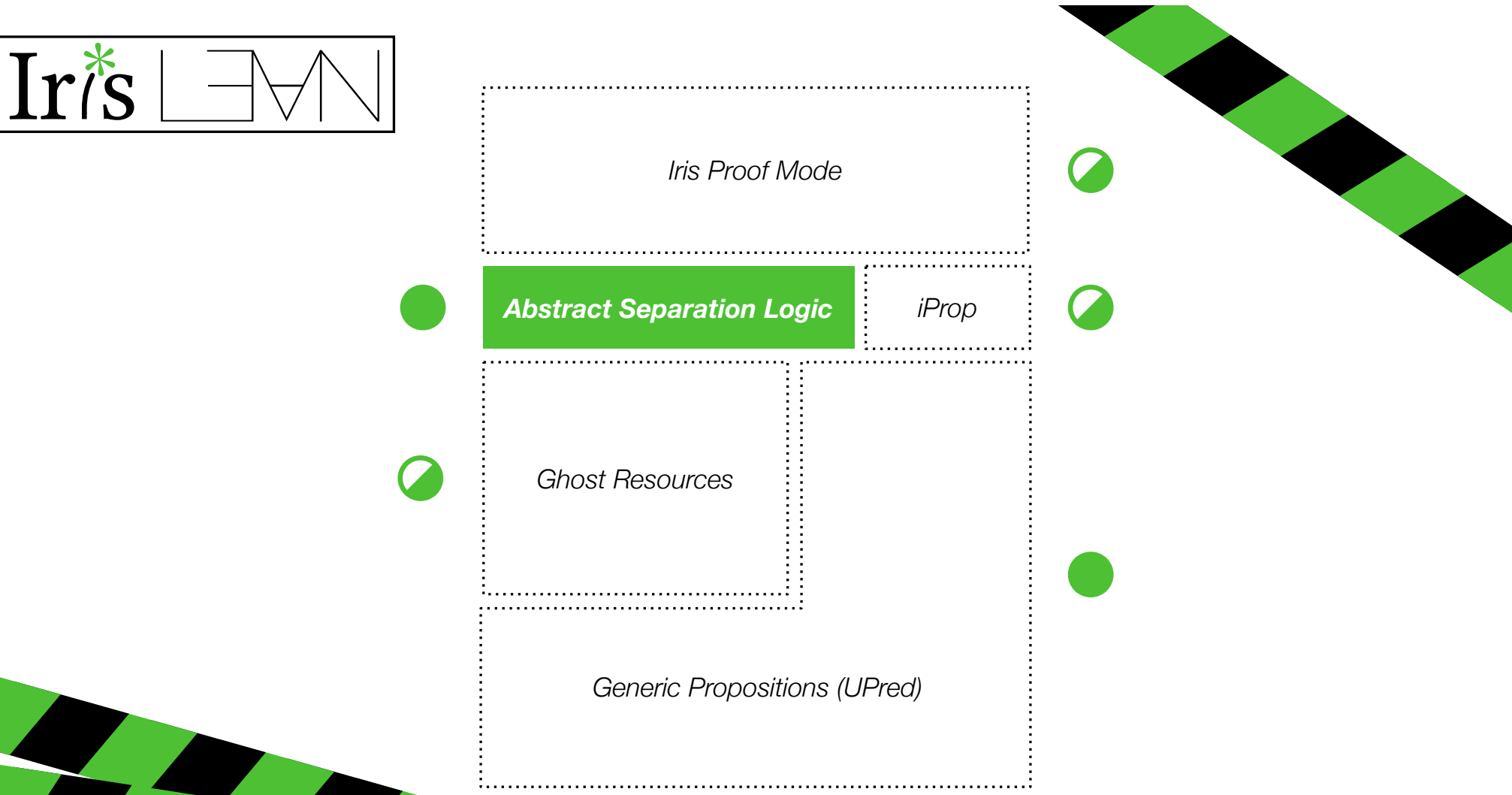
- \+ Monadic, imperative metaprograms
- ? Typeclasses?
- ? Faster?

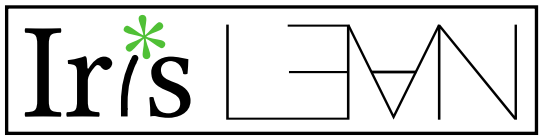


Iris Proof Mode

- Intro Patterns
- Cases Patterns
- Selection Patterns

- `iintro`
- `iexists`
- `ileft/iright`
- `ipure_intro`
- `iex_falso`
- `iexact`
- `ispecialize`
- `isplit`
- `icases`
- `iapply`
- `imod`
- `iinduction`


Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)

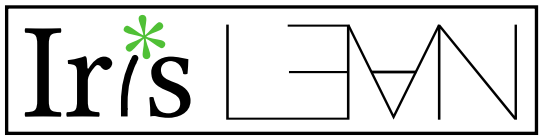


Abstract Separation Logic

- Interface for the syntax of a separation logic

Terms of the logic

```
class BIBase (PROP : Type u) where
  Entails : PROP → PROP → Prop
  emp : PROP
  pure : Prop → PROP
  and : PROP → PROP → PROP
  or : PROP → PROP → PROP
  imp : PROP → PROP → PROP
  …
```

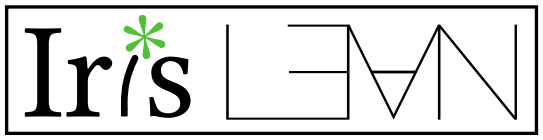


Abstract Separation Logic

“There exists (a : A) such that f(a) holds”

```
exists {A} : (f : A → PROP) → PROP
```

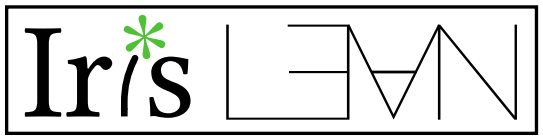
Function



Abstract Separation Logic

“There exists (a : A) such that f(a) holds”

```
exists {A} : (f : A → PROP) → PROP
```



Abstract Separation Logic

“There exists (a : A) such that f(a) holds”

```
exists {A} : (f : A → PROP) → PROP
```

Universe Levels

“There exists some (a ∈ S) which holds”

```
sExists : (S : Set PROP) → PROP
```

Iris Lean

Abstract Separation Logic

"There exists (a : A) such that f(a) holds"

```
exists {A} : (f : A → PROP) → PROP
```

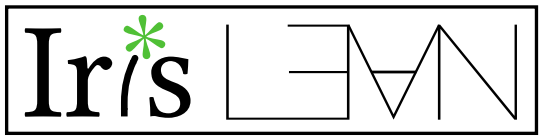
Universe Levels

"There exists some (a ∈ S) which holds"

```
sExists : (S : Set PROP) → PROP
```

```
exists (f : A → PROP) :=
  sExists (range f)
```

All Universes OK



“There exists (a : A) such that f(a) holds”

```
exists {A} : (f : A → PROP) → PROP
```

Universe Levels

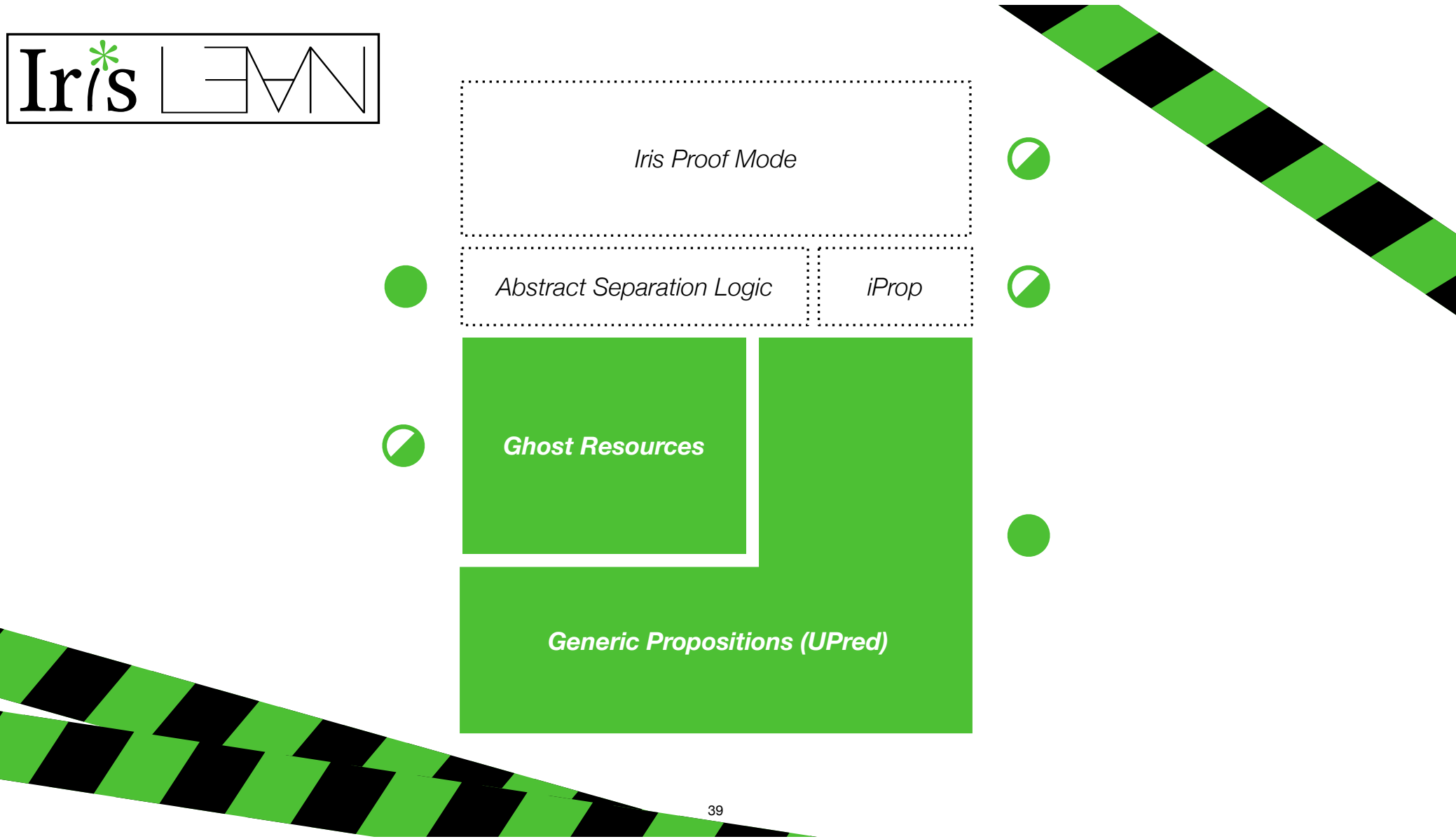
```
exists (f : A → PROP) :=
  sExists (range f)
```

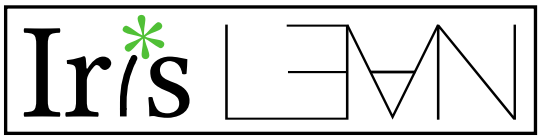
All Universes OK

“There exists some (a ∈ S) which holds”

```
sExists : (S : Set PROP) → PROP
```

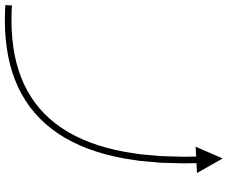
- ? Does everything generalize?
- ? Set quantifiers in Iris-Rocq?

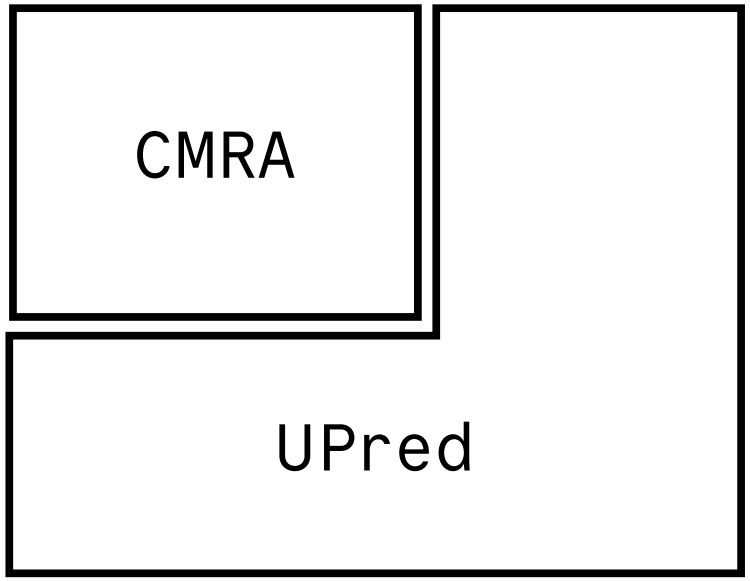

Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)



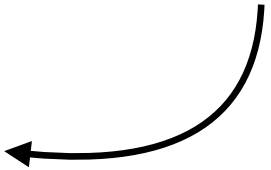
Base Logic

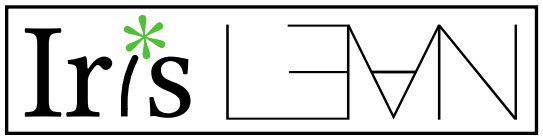
Algebraic abstraction





Step-indexed, higher order, separation logic





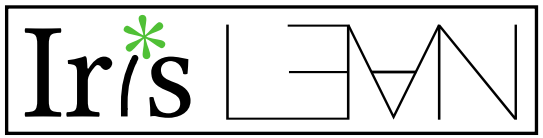
Base Logic

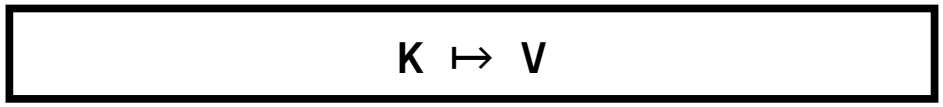
A library of CMRA combinators

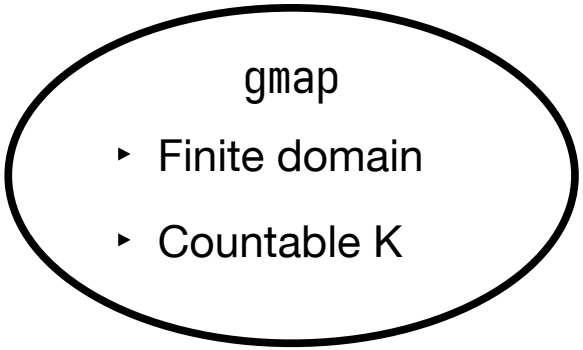
- `auth`
- `csum`
- `gset`
- `reservation_map`
- `vectors`

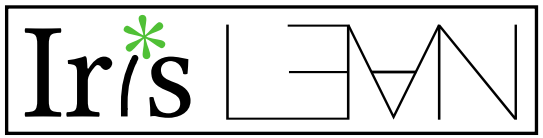
… & many more

- `excl`
- `agree`
- `frac` ***Generalized!***
- `dfrac`
- `view`
- `gen_map` ***New!***


Iris
Ghost Resources

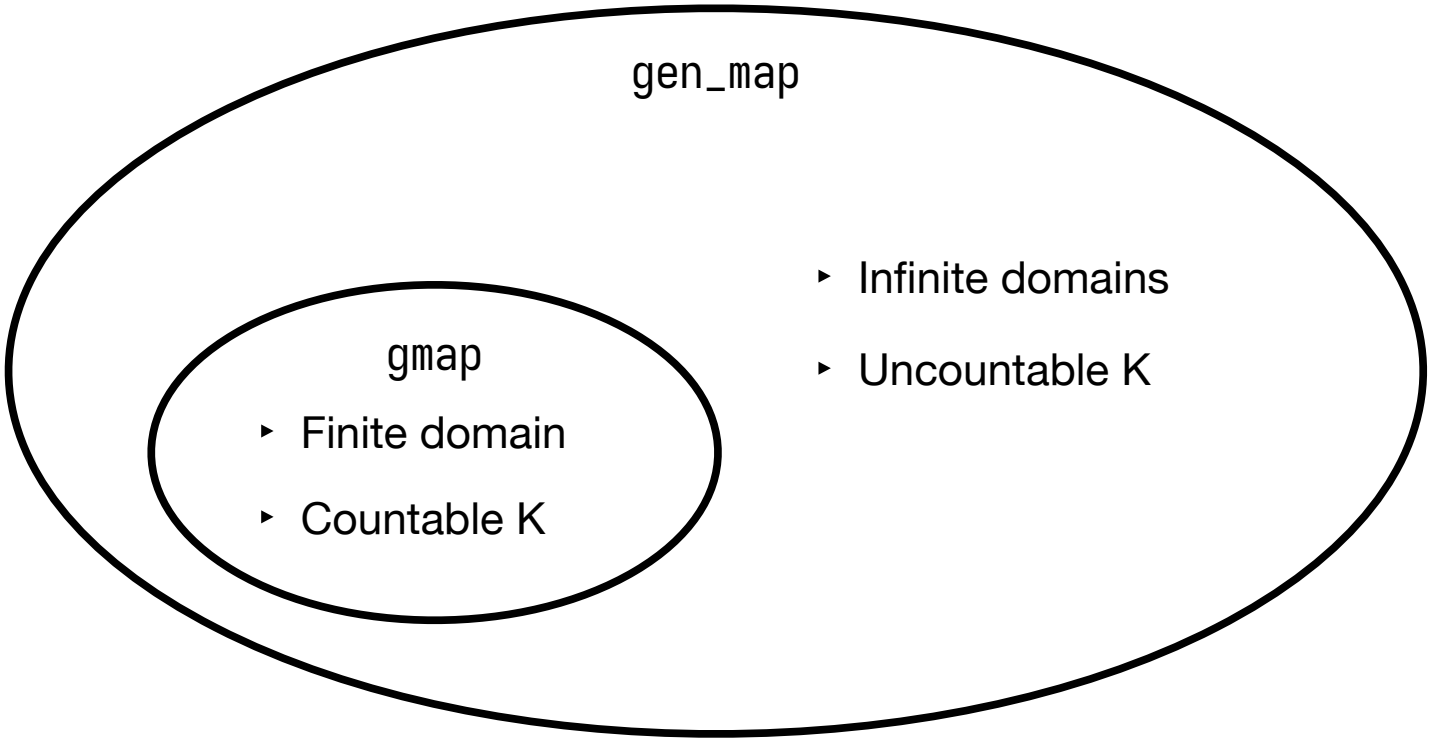

K ↦ V


gmap
Finite domain
Countable K



Ghost Resources

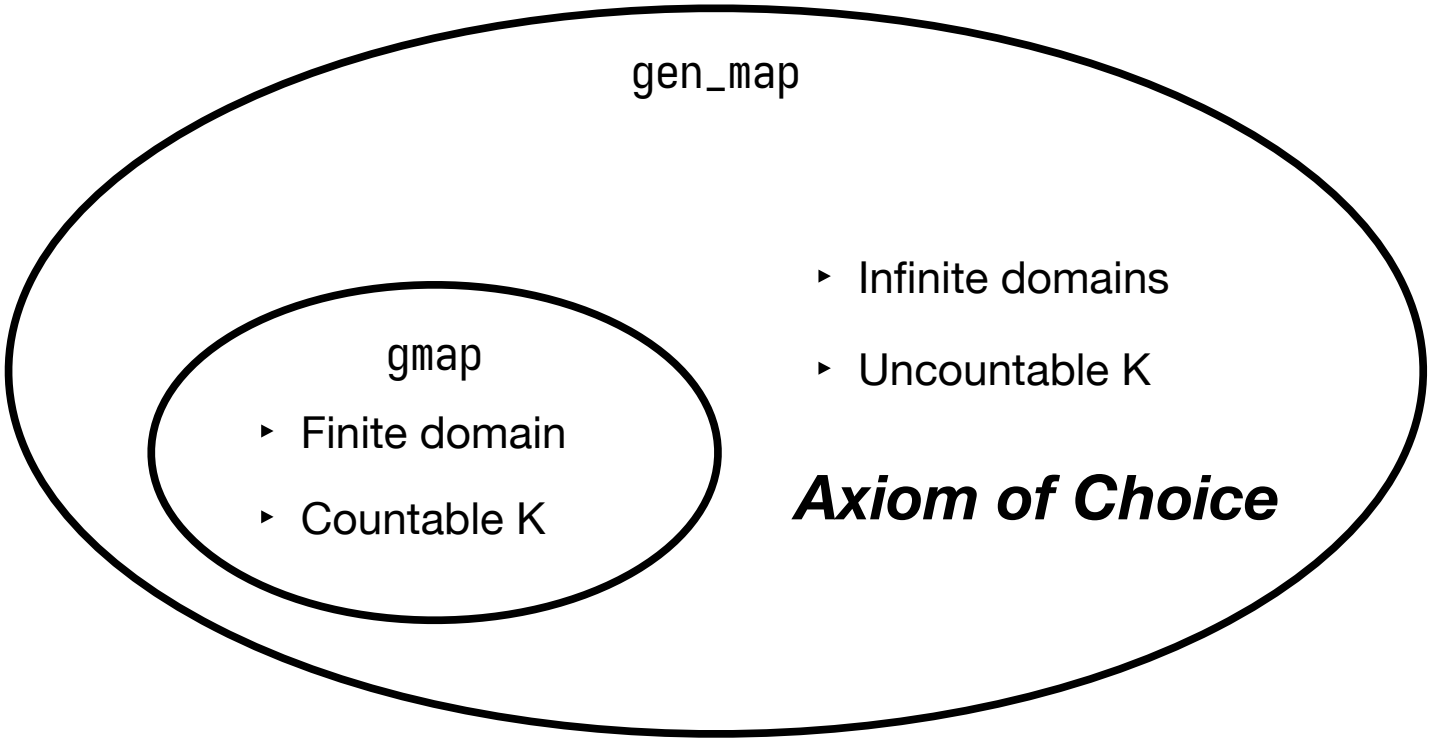
$K \mapsto V$

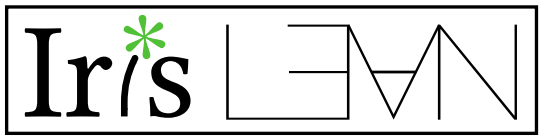


Iris

Ghost Resources

$K \mapsto V$

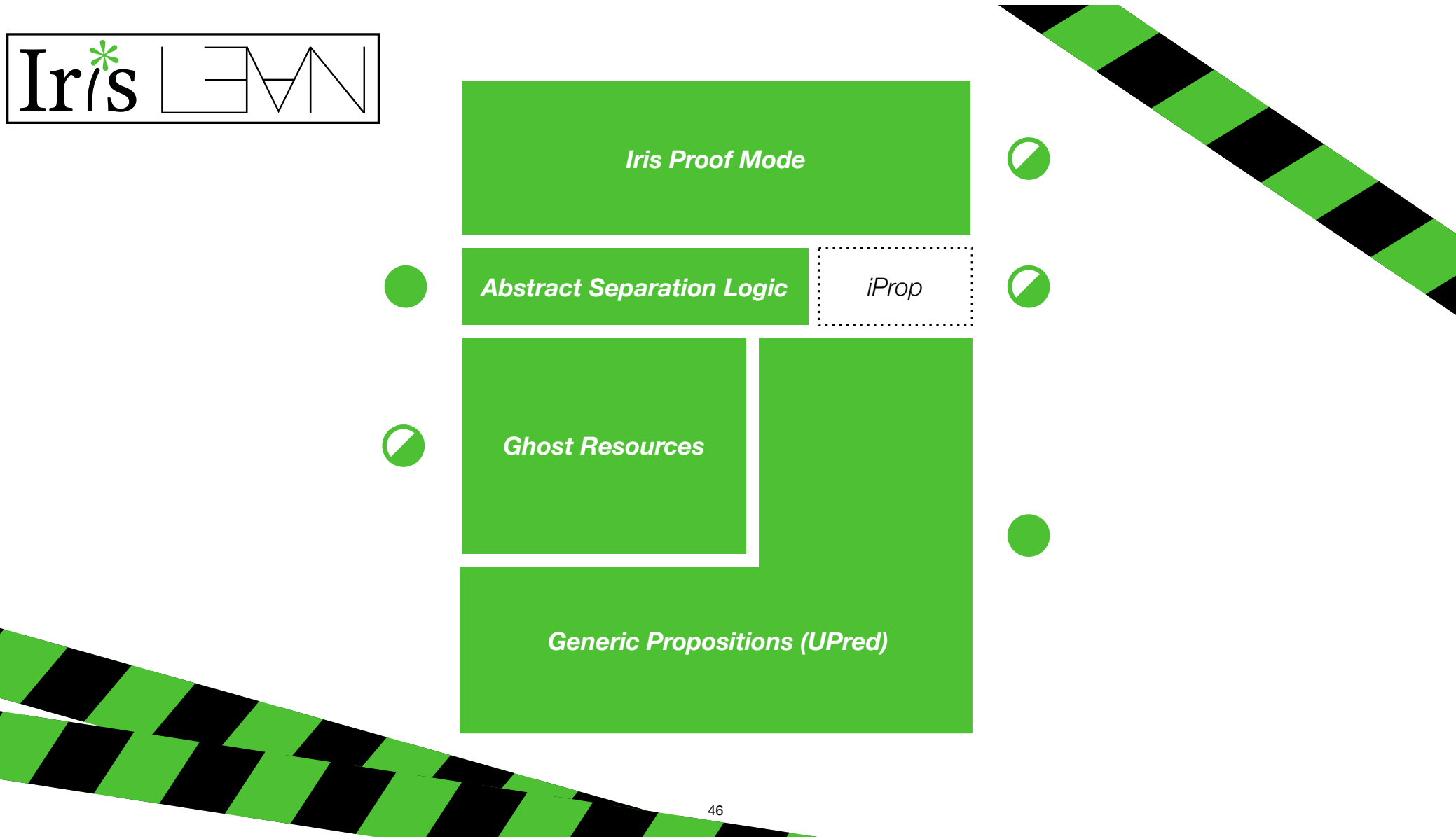


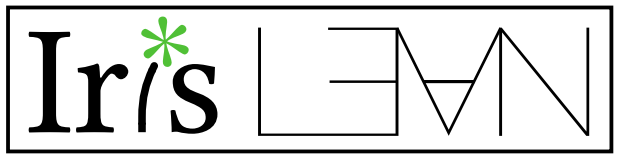


Ghost Resources

Mathematical Resources

```
variable {α : Type _} [MeasurableSpace α]
instance : CMRA (Measure α) where
 op μ₁ μ₂ := μ₁ + μ₂
  Valid μ := μ .univ ≤ 1
  validN_op_left :=
     (le_of_add_le_of_nonneg_left · <| zero_le _)
  assoc := by simp [add_assoc]
  comm := by simp [add_comm]
```


Iris Proof Mode
Abstract Separation Logic
iProp
Ghost Resources
Generic Propositions (UPred)



Works in Progress

Bluebell[1] Port

- Ethereum Foundation
- Iris + Mathlib Probability
- Independence CMRA

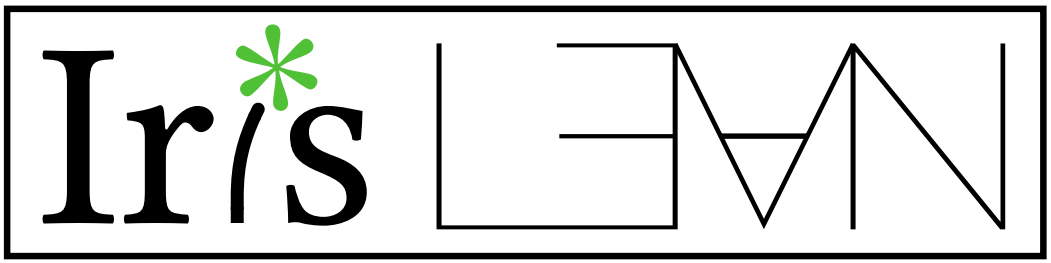
Nola[2] Port

- Microsoft Research
- With Aeneas (Rust)

Neuron Relational Logic

- Amazon Web Services
- Simple relational logic
- Full soundness proof

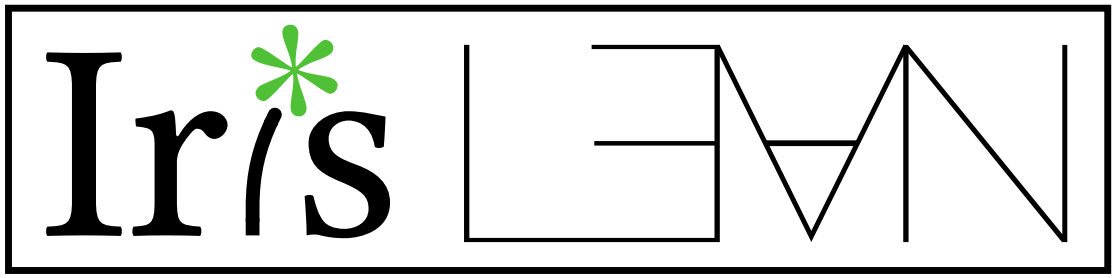
1 Jialu, D'Osualdo, Azadeh. Bluebell: An Alliance of Relational Lifting and Independence for Probabilistic Reasoning
2 Matsushita, Tsukada. Nola: Later-Free Ghost State for Verifying Termination in Iris



- Possible in Lean?
  - Yes!
- Iris-Lean adaptations?
  - Simplified Proof Mode
  - Set-Based Quantifiers
  - Generalized Maps

Contributors

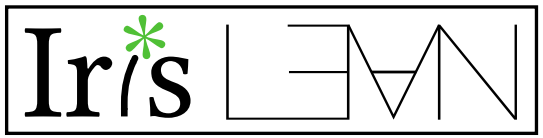
- *Mario Carneiro*
- *Сухарик (Suhr)*
- *Zongyuan Liu*
- *Shreyas Srinivas*
- *Oliver Soeser*
- *Léo Stefanesco*
- *Michael Sammler*
- *Remy Seassau*
- *Son Ho*
- *Alex Bai*
- *Quang Dao*
- *Joe Watt*
- *Mackie Loeffel*
- *Lars König*
- *Sebastian Ullrich*
- *Markus de Medeiros*



Github: `leanprover-community/iris-lean`

Lean Zulip: `#iris-lean`

Email: `mjd9606@nyu.edu`



iProp

```
abbrev F0 : OFunctorPre := constOF (Agree (LeibnizO String))
variable {GF} [E0 : ElemG GF F0]

example : ⊢ |==> ∃ (γ0 γ1 : GName) (s0 s1 : String),
    iOwn (E := E0) γ0 (toAgree ⟨s0⟩) ∗
    iOwn (E := E0) γ1 (toAgree ⟨s1⟩) := by
  let v0 : F0.ap (IProp GF) := toAgree ⟨"string0"⟩
  let v1 : F0.ap (IProp GF) := toAgree ⟨"string1"⟩
  refine emp_sep.mpr.trans <| (sep_mono (iOwn_alloc v1 (fun _ => trivial)) .rfl).trans ?_
  refine emp_sep.mpr.trans <| (sep_mono (iOwn_alloc v0 (fun _ => trivial)) .rfl).trans ?_
  -- Eliminate the bupds (by hand, until iMod is implemented)
  -- …
  istart
  iintro ⟨⟨γ0, Hγ0⟩, ⟨γ1, Hγ1⟩, -⟩
  iexists γ0
  iexists γ1
  iexists "string0"
  iexists "string1"
  isplit l [Hγ0]
  · iexact Hγ0
  · iexact Hγ1
```