Nola: Later-free ghost state for verifying termination in Iris

Yusuke Matsushita Kyoto University
Joint work w/ Takeshi Tsukada Chiba University

June 2, 2025 – Iris Workshop 2025 @ Inria, Paris

Brief self-introduction

- **Yusuke Matsushita 松下 祐介**
  - Software scientist, loves **Rust**
    - Assistant Prof. at KyotoU
  - My past work:

RustHorn
Senior thesis '19
ESOP '20 & TOPLAS '21

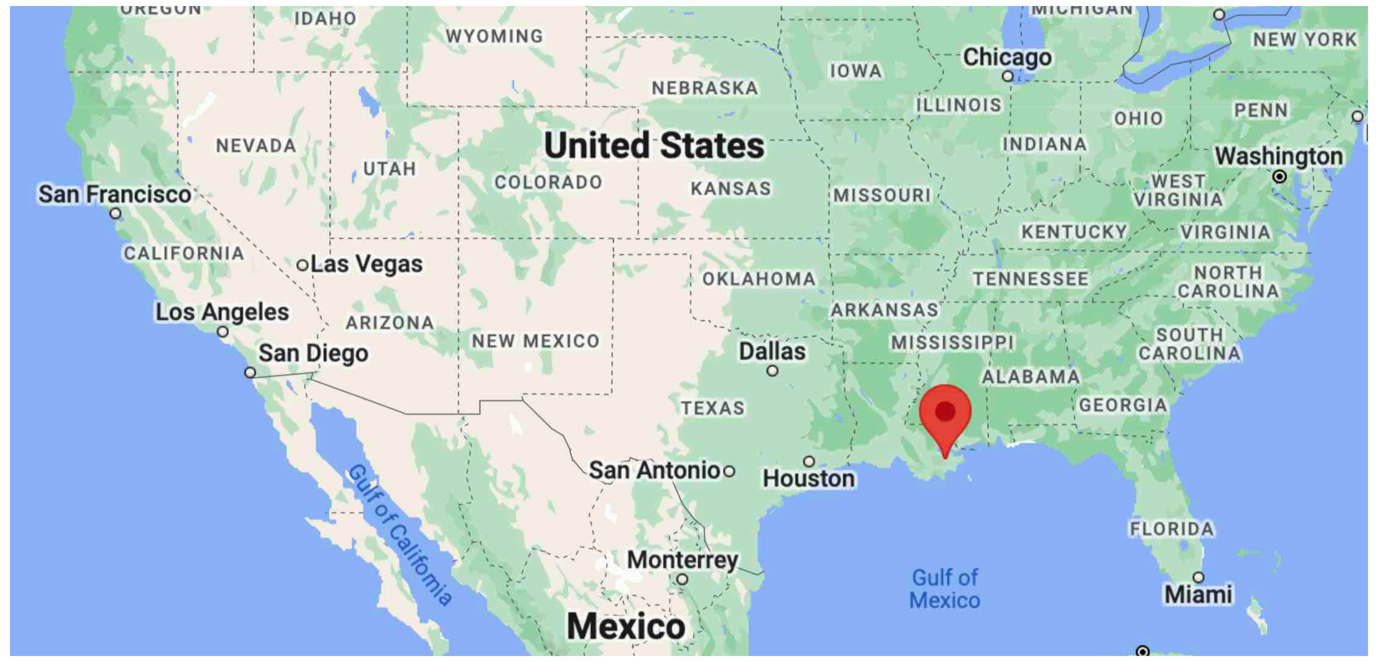
Rust's **borrows** made **"pure"** by **prophecies**

RustHornBelt
Master's thesis '21
PLDI '22

Internship at **Derek's** group, Extends RustBelt

POPL 2020 @ New Orleans, a.k.a. NOLA

POPL 2020 New Orleans



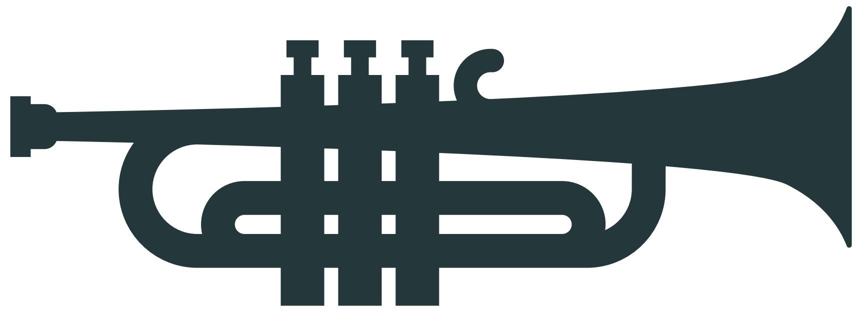
My latest work, Nola

Ph.D. dissertation ’23 & **PLDI ’25**

- ✦ **Later-free** shared mutable state in separation logic
  - ▸ Higher-order ghost state, but clears the notorious **later** $\triangleright$
    - Great for **termination** & **liveness** verification
    - Refines Iris’s **invariants** $\boxed{\triangleright P}$ & RustBelt’s **borrows** $\&^{\alpha} \triangleright P$
  - ▸ Key idea: **Custom syntax** $P \in \mathit{Fml}$ for SL formulas
    - **Extensible** & **Semantic** SL props under later
  - ▸ Case study: **RustHalt**, revised RustHornBelt
  - ▸ Fully **mechanized** as a **library** of **Iris**

Iris

No later



github.com/hopv/nola

On my GitHub pages

Why Nola?

Termination verification should be easy

- **Meta-logic induction & composition** should work
  - And that's the case for **traditional separation logic**

Example 1 $\quad$ `fn decrloop(r) { if *r > 0 { *r = *r − 1; decrloop(r) } }`

Total correctness $\quad \forall n \in \mathbb{N}.\ [\mathtt{r} \mapsto n]\ \mathtt{decrloop(r)}\ [\lambda\_.\ \mathtt{r} \mapsto 0]$

Proof. **Induction** in the **meta-logic** (e.g., Rocq) $\quad$ Case $n = 0 \leftarrow$ Case $n = 1$ $\leftarrow$ Case $n = 2 \leftarrow \cdots$

Example 2 $\quad [\mathtt{r} \mapsto v]\ \mathtt{*r = ndnat;\ decrloop(r)}\ [\lambda\_.\ \mathtt{r} \mapsto 0]$

Unbounded termination

Proof. **Composition** of the former and the rule $[\top]\ \mathtt{ndnat}\ [\lambda v.\ v \in \mathbb{N}]$

Question

What about shared mutable state?

Traditional SL: **Mutable** state is **not sharable**

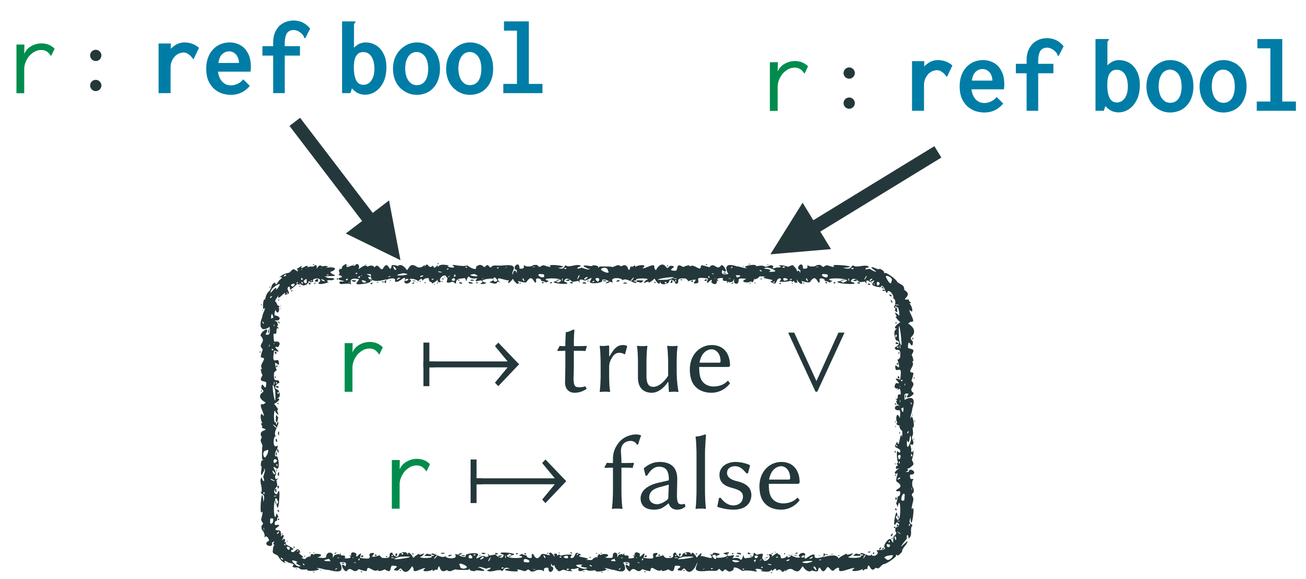
$$\sout{\ell \mapsto v \Rightarrow \ell \mapsto v * \ell \mapsto v}$$

Invariants: Shared mutable state

- **Invariant** $\boxed{P}$: Roughly, the situation P always holds
  - **Mutable** state **shared** across threads etc.
    - Key of **Iris** [Jung+ '15], Typical **higher-order ghost state**

Logical state that depends on SL assertions

Shared mutable ref $\quad \mathtt{r} : \mathbf{ref\ bool} \;\triangleq\; \boxed{\mathtt{r} \mapsto \mathsf{true} \lor \mathtt{r} \mapsto \mathsf{false}}$



$$[\top]\ \mathbf{ref}\ \mathsf{true}\ \left[\lambda \mathtt{r}.\ \boxed{\mathtt{r} \mapsto \mathsf{true} \lor \mathtt{r} \mapsto \mathsf{false}}\right]$$

$$\left[\boxed{\mathtt{r} \mapsto \mathsf{true} \lor \mathtt{r} \mapsto \mathsf{false}}\right]\ {*\mathtt{r}} = \mathsf{false}\ [\top]$$

$$\left[\boxed{\mathtt{r} \mapsto \mathsf{true} \lor \mathtt{r} \mapsto \mathsf{false}}\right]\ {*\mathtt{r}}\ [\lambda \mathtt{v}.\ \mathtt{v} = \mathsf{true} \lor \mathtt{v} = \mathsf{false}]$$

Even **nested** ref! $\quad \mathtt{r} : \mathbf{ref}\ (\mathbf{ref\ bool}) \;\triangleq\; \boxed{\exists \mathtt{s}.\ \mathtt{r} \mapsto \mathtt{s} * (\mathtt{s} : \mathbf{ref\ bool})}$

Sad news: Naive later-free invariant is unsound

- Naive rule causes unsound “**infinite loops**” in logic

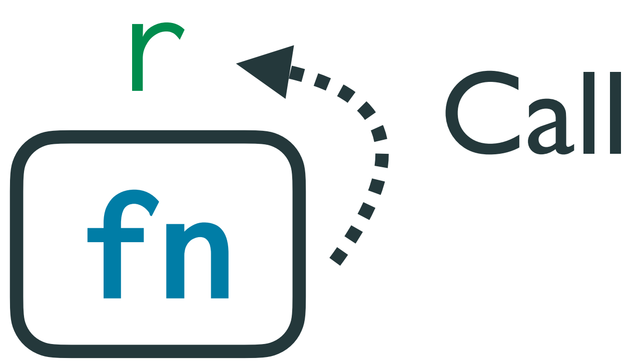
Naive access rule

$$\frac{[P * Q]\ \mathtt{ae}\ [\lambda \mathtt{v}.\ P * \Psi\,\mathtt{v}]}{[\boxed{P} * Q]\ \mathtt{ae}\ [\Psi]}$$

⇩

Paradox

Landin’s knot: **Loop** by a **shared mutable** ref of a **closure**




```
landin ≜ let r = ref fn(){} in
         *r = fn(){ (*r)() }; (*r)()
```

With the naive access rule, we can **wrongly** prove $[\top]\ \mathtt{landin}\ [\top]$

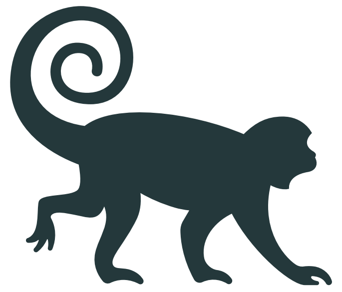
Proof. Via an **invariant** with a **Hoare triple** $\boxed{\exists \mathtt{f}.\ \mathtt{r} \mapsto \mathtt{f} * [\top]\ \mathtt{f()}\ [\top]}$

Existing approach: Later

- **Weakened invariant** $\boxed{\triangleright P}$ **: The situation** $\triangleright P$ **always holds**
  - **Later** modality $\triangleright$: "Holds one index later" [Nakano '00]
    - For **soundness**, Notorious **obstacle** of verification in **Iris**

$$\frac{\left[P * Q\right]\ \mathsf{ae}\ \left[\lambda \mathrm{v}.\ P * \Psi\, \mathrm{v}\right]}{\left[\boxed{P} * Q\right]\ \mathsf{ae}\ \left[\Psi\right]}$$

Naive but **unsound**



$$\frac{\left[\triangleright P * Q\right]\ \mathsf{ae}\ \left[\lambda \mathrm{v}.\ \triangleright P * \Psi\, \mathrm{v}\right]}{\left[\boxed{\triangleright P} * Q\right]\ \mathsf{ae}\ \left[\Psi\right]}$$

Sound but **weakened**

Iris

Problem about later

- **Laters** $\triangleright$ **are in the way**
  - Very basic SL props like $\ell \mapsto v$ are **timeless** $\triangleright P \equiv P$ (up to $\diamond$)
  - But many SL props, including **invariants** $\boxed{\triangleright P}$, are **not timeless**
  - **Later** $\triangleright$ **blocks access** to esp. inside of **nested refs**

$$\mathtt{r} : \mathtt{ref}\,(\mathtt{ref\ bool}) \triangleq \boxed{\triangleright\, \exists \mathtt{s}.\ \mathtt{r} \mapsto \mathtt{s} * (\mathtt{s} : \mathtt{ref\ bool})}$$

$$[\, \mathtt{r} : \mathtt{ref}\,(\mathtt{ref\ bool})\,] *\mathtt{r}\ [\, \lambda \mathtt{s}.\ \triangleright (\mathtt{s} : \mathtt{ref\ bool})\,]$$

Blocks access!

Existing workaround & Its limitation

- **Step-indexing:** Tie program steps with laters $\triangleright$

$$\frac{e \hookrightarrow e' \qquad \{P\}\ e'\ \{\Psi\}}{\{\triangleright P\}\ e\ \{\Psi\}}$$

Wait for one program step,
then you can strip off one later

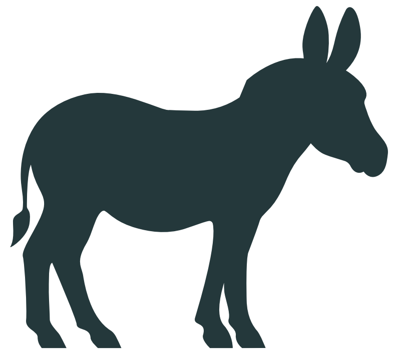
- Does not work well in **termination** verification

Paradox

Step-indexing on **total** Hoare triple $\frac{e \hookrightarrow e' \qquad [P]\ e'\ [\Psi]}{[\triangleright P]\ e\ [\Psi]}$

lets you **wrongly** prove $[\top]\ \texttt{loop}\ [\bot]$ under $\texttt{loop} \hookrightarrow \texttt{loop}$

Proof. By **Löb induction** $\frac{\triangleright P \Rightarrow P}{P}$



Recent approaches to termination & liveness

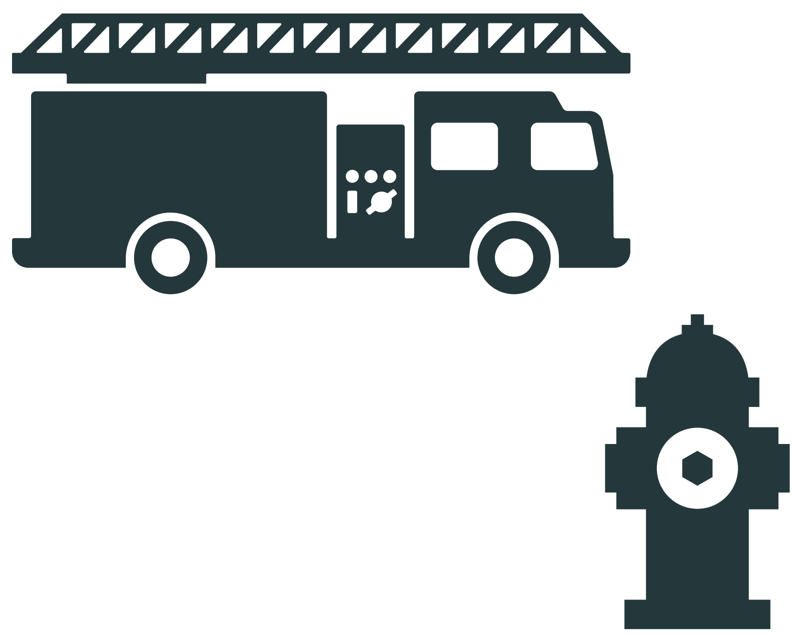
1. **Give up** invariants on non-timeless propositions
   - Many recent SLs for **advanced liveness properties**:
     Simuliris [Gäher+ '22], CCR [Song+ '23], Fairness Logic [Lee+ '23], etc.
2. **Finitely** bound program steps [Mevel+ '19]
   - **Bounded** termination $\$n$, not genuine liveness

3. Use **transfinite** step-indexing [Spies+ '21]
   - Still need to transfinitely **bound** program steps $\$\alpha$
   - **Lose** good properties of later $\triangleright(P * Q) \Leftrightarrow \triangleright P * \triangleright Q$

Used by, e.g.,
RustBelt's lifetime logic

Goal: Natural & modular liveness verification

- Verify **liveness** naturally for **shared mutable** state
  - Want to use natural **meta-logic induction**
    - Strong **composability** of proof, **No bounding**
  - **Sound later-free higher-order ghost state**
    - **Invariants** over **closures** etc. should be handled with care, due to Landin's knot paradox
    - But **nested invariants** should **not** suffer from the later



Is that even possible?

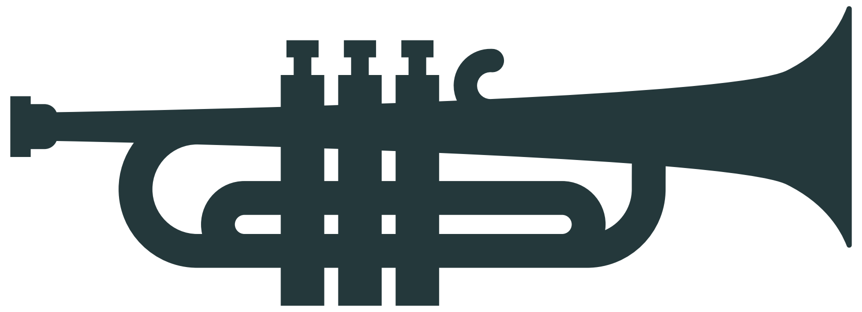
Solution, Nola

Solution, Nola

Ph.D. dissertation '23 & **PLDI '25**

- ✦ **Later-free** shared mutable state in separation logic
  - ▸ Higher-order ghost state, but clears the notorious **later** $\triangleright$
    - Great for **termination** & **liveness** verification
    - Refines Iris's **invariants** $\boxed{\triangleright P}$ & RustBelt's **borrows** $\&^{\alpha} \triangleright P$
  - ▸ Key idea: **Custom syntax** $P \in \mathit{Fml}$ for SL formulas
    - **Extensible** & **Semantic** SL props under later
  - ▸ Case study: **RustHalt**, revised RustHornBelt
  - ▸ Fully **mechanized** as a **library** of **Iris**

No later

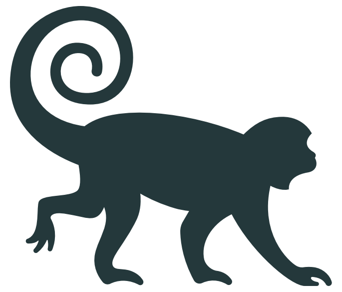


github.com/hopv/nola

On my GitHub pages

Key idea: Custom syntax for SL formulas

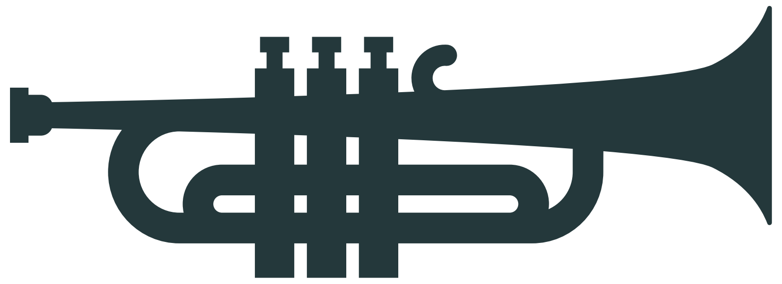
- **Custom syntax** Fml **& semantics** $[\![\ ]\!]$ **for SL formulas**
  - Intuitively, $[\![\ ]\!] \colon \mathit{Fml} \to \mathit{iProp}$ is **well-behaved** substitute for $\triangleright$
    - So many SL props become **“timeless”** under well-designed $[\![\ ]\!]$



$$\frac{[P * Q]\ \mathsf{ae}\ [\lambda \mathrm{v}.\ P * \Psi\,\mathrm{v}]}{[\boxed{P} * Q]\ \mathsf{ae}\ [\Psi]}$$

Iris

$$\frac{[\triangleright P * Q]\ \mathsf{ae}\ [\lambda \mathrm{v}.\ \triangleright P * \Psi\,\mathrm{v}]}{[\boxed{\triangleright P} * Q]\ \mathsf{ae}\ [\Psi]}$$



$$\frac{[[\![P]\!] * Q]\ \mathsf{ae}\ [\lambda \mathrm{v}.\ [\![P]\!] * \Psi\,\mathrm{v}]^{\mathrm{Winv}\,[\![\ ]\!]}}{[\boxed{P} * Q]\ \mathsf{ae}\ [\Psi]^{\mathrm{Winv}\,[\![\ ]\!]}}$$

$P \in \mathit{Fml}$

$[\![\ ]\!] \colon \mathit{Fml} \to \mathit{iProp}$

Power of SL formulas

- **SL formulas can be really expressive & semantic**

Non-monotone $[\![P \mathrel{-\!\!*} Q]\!] \triangleq [\![P]\!] \mathrel{-\!\!*} [\![Q]\!]$

$[\![\forall\, \Phi]\!] \triangleq \forall a.\, [\![\Phi\, a]\!]$ HOAS

E.g., $Fml \ni P, Q \ ::=_{\nu,\mu}\ P * Q \mid P \mathrel{-\!\!*} Q \mid P \lor Q \mid \forall_A \Phi \mid \exists_A \Phi$

$\mid \phi \ (\phi \in Prop) \mid \mathsf{r} \mapsto \mathsf{v} \mid \boxed{P} \ (P \in_\nu Fml) \mid \check{\triangleright}\, \hat{P} \ (\hat{P} \in \triangleright\, iProp)$

$[\![\boxed{P}]\!] \triangleq \boxed{P}$ **No later!** + **Productivity**

Any **semantic** SL props under later!

$[\![\check{\triangleright}\, \hat{P}]\!] \triangleq \check{\triangleright}\, \hat{P}$

$\triangleright P \triangleq \check{\triangleright}\, \mathrm{next}\, P$

$\check{\triangleright}\, \mathrm{next}\, P \triangleq \triangleright P$

- **SL formulas can even be extensible**
  - By **parameterizing** over the constructors, just like **iProp's Σ**

Later-free access to nested refs

- With Nola, we can go inside **nested refs w/o later!**
  - Allows **natural termination verification**

$$\mathsf{r} : \mathbf{ref\ bool} \quad \triangleq \quad \boxed{\mathsf{r} \mapsto \mathrm{true} \vee \mathsf{r} \mapsto \mathrm{false}}$$

$$\mathsf{r} : \mathbf{ref}\ (\mathbf{ref\ bool}) \quad \triangleq \quad \boxed{\exists\, \mathsf{s}.\ \mathsf{r} \mapsto \mathsf{s} * (\mathsf{s} : \mathbf{ref\ bool})}$$

$$\big[\, [\![\mathsf{r} : \mathbf{ref}\ (\mathbf{ref\ bool})]\!] \,\big] \ast\!\mathsf{r}\ \big[\, \lambda \mathsf{s}.\ [\![\mathsf{s} : \mathbf{ref\ bool}]\!] \,\big]^{\mathrm{Winv}\,[\![\ ]\!]}$$

Later-free access!

Verification example: Infinite list

- **Termination** of iteration can be **naturally** verified

$$\texttt{list}\ \Phi\ \texttt{r} \triangleq \boxed{\Phi\ \texttt{r}} * \boxed{\exists \texttt{s}.\ \texttt{r}+1 \mapsto \texttt{s} * \texttt{list}\ \Phi\ \texttt{s}}$$

$$\big[\,[\![\texttt{list}\ \Phi\ \texttt{r}]\!]\,\big]\ *(\texttt{r}+1)\ \big[\lambda \texttt{s}.\ [\![\texttt{list}\ \Phi\ \texttt{s}]\!]\big]^{\mathrm{Winv}\,[\![\ ]\!]}$$

$$\texttt{llist}\ \Phi\ \texttt{r} \triangleq \boxed{\triangleright \Phi\ \texttt{r}} * \boxed{\triangleright \exists \texttt{s}.\ \texttt{r}+1 \mapsto \texttt{s} * \texttt{llist}\ \Phi\ \texttt{s}}$$

$$\big[\,\texttt{llist}\ \Phi\ \texttt{r}\,\big]\ *(\texttt{r}+1)\ \big[\lambda \texttt{s}.\ \triangleright\ \texttt{llist}\ \Phi\ \texttt{s}\big]$$

Iteration

```
fn iterc(f,c,r) { if *c > 0 {
    f(r); *c = *c − 1; iterc(f,c,*(r+1)) } }
```

Termination!

By meta-logic induction

$$\frac{\forall \texttt{r}.\ \big[\boxed{\Phi\ \texttt{r}}\big]\ \texttt{f(r)}\ \big[\top\big]^{\mathrm{Winv}\,[\![\ ]\!]}}{\big[\,[\![\texttt{list}\ \Phi\ \texttt{r}]\!] * \texttt{c} \mapsto n\,\big]\ \texttt{iterc(f,c,r)}\ \big[\texttt{c} \mapsto 0\big]^{\mathrm{Winv}\,[\![\ ]\!]}}$$

Custom view shifts & Hoare triples

- Enable **customizing** the **world satisfaction**
  - Or the **"mother invariant"** for higher-order ghost state

$$P \Rrightarrow^{W} Q \;\triangleq\; P * W \Rrightarrow Q * W$$

World satisfactions can be **combined**

$$\frac{P \Rrightarrow^{W} Q}{P \Rrightarrow^{W * W'} Q} \qquad \frac{[P]\ \mathsf{e}\ [\Psi]^{W}}{[P]\ \mathsf{e}\ [\Psi]^{W * W'}}$$

$$\frac{P \Rrightarrow^{W} P' \qquad [P']\ \mathsf{e}\ [\Psi]^{W}}{[P]\ \mathsf{e}\ [\Psi]^{W}} \qquad \frac{[P]\ \mathsf{e}\ [\Psi']^{W} \qquad \forall \mathsf{v}.\ \Psi'\ \mathsf{v} \Rrightarrow^{W} \Psi\ \mathsf{v}}{[P]\ \mathsf{e}\ [\Psi]^{W}}$$

Model of Nola's invariant

- **Nola's invariant generalizes Iris's invariant**
  - Fml generalizes $\blacktriangleright\, \mathit{iProp}$, $[\![\ ]\!]$ generalizes $\check{\triangleright} : \blacktriangleright\, \mathit{iProp} \to \mathit{iProp}$

$$\textsc{Inv}\ \mathit{Fml} \triangleq \textsc{Auth}\left(\mathbb{N} \xrightarrow{\text{fin}} \textsc{Ag}\ \mathit{Fml}\right)$$

$$\boxed{P} \triangleq \exists \iota.\ \left[\circ\, [\iota \leftarrow \mathrm{ag}\ P]\right]^{\gamma_{\textsc{Inv}}}$$

$$\mathrm{Winv}\ [\![\ ]\!] \triangleq \exists\, I : \mathbb{N} \xrightarrow{\text{fin}} \mathit{Fml}.\ \left[\bullet\, \mathrm{ag}\ I\right]^{\gamma_{\textsc{Inv}}} * \mathop{\Large *}_{\iota \in \mathrm{dom}\, I} \left(\left([\![I\ \iota]\!] * [\mathrm{D}]_\iota\right) \vee [\mathrm{E}]_\iota\right)$$

Iris

$$\textsc{lInv} \triangleq \textsc{Auth}\left(\mathbb{N} \xrightarrow{\text{fin}} \textsc{Ag}\,(\blacktriangleright\, \mathit{iProp})\right)$$

$$\boxed{\triangleright P} \triangleq \exists \iota.\ \left[\circ\, [\iota \leftarrow \mathrm{ag}\,(\mathrm{next}\ P)]\right]^{\gamma_{\textsc{lInv}}}$$

$$\mathrm{Wlinv} \triangleq \exists\, \hat{I} : \mathbb{N} \xrightarrow{\text{fin}} \blacktriangleright\, \mathit{iProp}.\ \left[\bullet\, \mathrm{ag}\ \hat{I}\right]^{\gamma_{\textsc{lInv}}} * \mathop{\Large *}_{\iota \in \mathrm{dom}\, \hat{I}} \left(\left(\check{\triangleright}\, \hat{I}\ \iota * [\mathrm{D}]_\iota\right) \vee [\mathrm{E}]_\iota\right)$$

Soundness & Expressivity

- **Well-definedness** is the key to **soundness**
  - *Fml*'s reference to *iProp* should be **contractive**
    - For well-definedness of $iProp = (\textsc{Inv}\ Fml \times \cdots) \to Prop$
  - $[\![\ ]\!]$ should be **well-defined** & non-expansive
    - Landin's knot **paradox** does **not** occur

$$[\![[\top]\ \mathsf{e}\ [\top]]\!] \triangleq_{?} \left[\top\right]\ \mathsf{e}\ \left[\top\right]^{\mathrm{Winv}\,[\![\ ]\!]}$$

← Invalid circular ref to $[\![\ ]\!]$

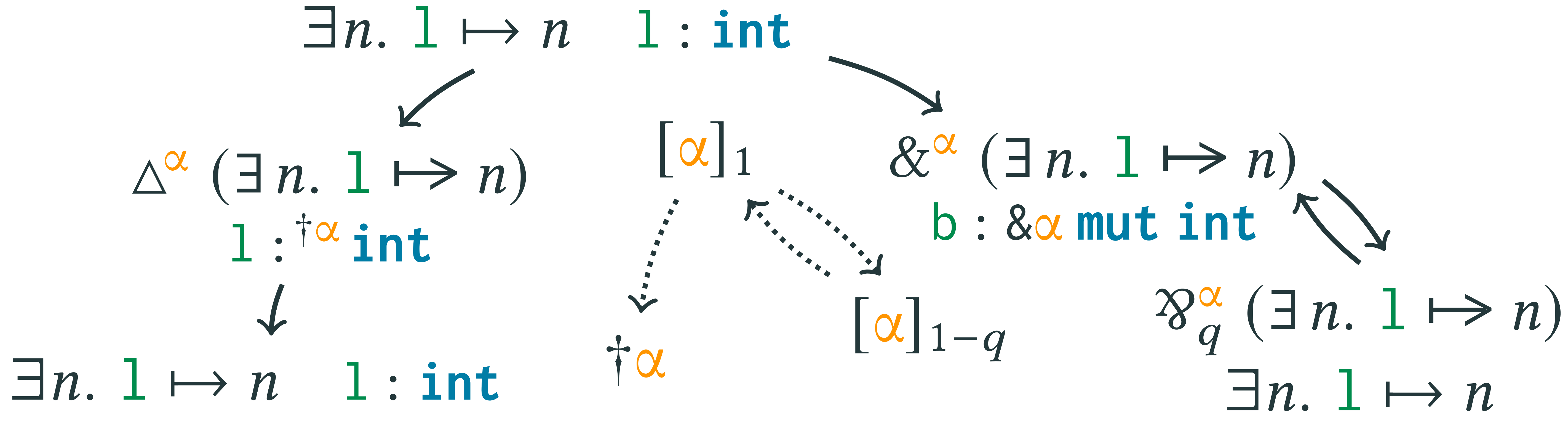
- Allows **flexible** construction for **extra expressivity**
  - E.g., **Stratification** $[\![\ ]\!]_i : Fml_i \to iProp$ $\quad [\![[P]\ \mathsf{e}\ [\Phi]]\!]_1 \triangleq \left[[\![P]\!]_1\right]\ \mathsf{e}\ \left[[\![\Phi]\!]_1\right]^{\mathrm{Winv}\,[\![\ ]\!]_0}$

Rust-style borrows

- **Later-free** version of RustBelt's **lifetime** logic [Jung+ '18]
  - **Advanced** higher-order ghost state, but **analogous** to invariants

α

```
let mut l = 0;
let b = &mut l;
*b += 7;
print(l);
```



$$[\![P]\!] \Rrightarrow^{\mathrm{Wbor}\,[\![\;]\!]} \&^{\alpha} P * \triangle^{\alpha} P \qquad \dagger\alpha * \triangle^{\alpha} P \Rrightarrow^{\mathrm{Wbor}\,[\![\;]\!]} [\![P]\!]$$

$$\&^{\alpha} P * [\alpha]_q \Rrightarrow^{\mathrm{Wbor}\,[\![\;]\!]} \parr^{\alpha}_q P * [\![P]\!] \qquad \parr^{\alpha}_q P * [\![P]\!] \Rrightarrow^{\mathrm{Wbor}\,[\![\;]\!]} \&^{\alpha} P * [\alpha]_q$$

Last challenge: Semantic alteration of syntax

- Want to prove **subtyping** on **shared mutable refs**
  - Need **semantic alteration** of **SL formulas** for **invariants**

Goal
$$\frac{\mathsf{T} \le \mathsf{U} \qquad \mathsf{U} \le \mathsf{T}}{\texttt{ref}\ \mathsf{T} \le \texttt{ref}\ \mathsf{U}}$$

So **need** something like
$$\frac{[\![P]\!] \Leftrightarrow [\![Q]\!]}{[\![\boxed{P}]\!] \Leftrightarrow [\![\boxed{Q}]\!]}$$

- **Semantic** equivalence of **syntax** SL formulas?

$$[\![\boxed{P}]\!] \triangleq \boxed{P}$$

Not **semantic**!

$$[\![\boxed{P}]\!] \triangleq_? \exists Q \text{ s.t. } [\![P]\!] \Leftrightarrow [\![Q]\!].\ \boxed{Q}$$

Invalid **circular self-ref**

Solution: Magic derivability

- **Fixpoint**-like **semantic** construction of **derivability**
  - Key: **Parameterize** the semantics over derivability **candidates**

$$Judg \ni J \ ::= \ P \Leftrightarrow Q \qquad [\![\boxed{P}]\!]_\delta \triangleq \exists Q \text{ s.t. } \delta(P \Leftrightarrow Q).\ \boxed{Q}$$

$$[\![P \Leftrightarrow Q]\!]^+_\delta \triangleq [\![P]\!]_\delta \Leftrightarrow [\![Q]\!]_\delta \qquad [\![\ ]\!]^+ : (Judg \to iProp) \to (Judg \to iProp)$$

$$\text{der } J \Rightarrow [\![J]\!]^+_{\text{der}} \qquad \frac{\forall \delta \in Deriv.\ [\![P]\!]_\delta \Leftrightarrow [\![Q]\!]_\delta}{\forall \delta \in Deriv.\ [\![\boxed{P}]\!]_\delta \Leftrightarrow [\![\boxed{Q}]\!]_\delta}$$

$$\text{der} \in Deriv$$

Model

Deriv is the closure of $[\![\ ]\!]^+$ & conjunction,
der is the smallest element of *Deriv*

If $[\![\ ]\!]^+$ is monotone,
der is exactly the fixpoint

Case study: RustHalt

- **Semantic** foundation for verifying **Rust termination**
  - Refined **RustHornBelt** [M+ '22] w/ **Nola**'s invariants & borrows
    - Each **Rust type** is modeled as a parameterized **SL formula** *Fml*
  - **Semantic** typing / **logical** relation that enjoys **extensibility**

Example `fn iter(f,l) { match l { Nil ⇒ (), Cons(a,l′) ⇒ { f(a); iter(f,*l′) } } }`

$$\frac{\forall \mathtt{a}.\ \ \mathtt{a} : \&\alpha\ \mathbf{mut}\ \mathtt{T} \vdash \mathtt{f(a)} \dashv \_.\ \rightsquigarrow \lambda post, [(a, a')].\ a' = f\,a \to post\,[]}{\mathtt{l} : \&\alpha\ \mathbf{mut}\ \mathtt{List<T>} \vdash \mathtt{iter(f,l)} \dashv \_.\ \rightsquigarrow \lambda post, [(l, l')].\ l' = \mathrm{map}\,f\,l \to post\,[]}$$

$$[\![\Gamma \vdash_\gamma \mathtt{e} \dashv \mathtt{r}.\Gamma' \rightsquigarrow pre]\!] \triangleq \forall \hat{post}, t, q.\ \left[\ \exists \bar{\hat{a}}.\ \langle \lambda\pi.\ pre\,(\hat{post}\,\pi)\,(\overline{\hat{a}\,\pi})\rangle * [\gamma]_q * [t] * [\![\Gamma]\!](\bar{\hat{a}}, t)\ \right]$$

$$\mathtt{e}\ \left[\ \lambda \mathtt{r}.\ \exists \bar{\hat{b}}.\ \langle \lambda\pi.\ \hat{post}\,\pi\,(\overline{\hat{b}\,\pi})\rangle * [\gamma]_q * [t] * [\![\Gamma']\!](\bar{\hat{b}}, t)\ \right]^{\mathrm{Wrh}\,[\![\ ]\!]}$$

Recent application: Lilo Lee+ OOPSLA '25

- **Fair liveness** verification for **shared mutable state**
  - Extends **fairness logic** [Lee+ '23] with **Nola**-style invariants
    - **Stratification** is used for higher-order features

Example

$$\begin{array}{l}\texttt{while}\ (1)\ \{\ \mathcal{Y};\ X \coloneqq 1;\\ \quad \texttt{do}\ \{\ \mathcal{Y};\ a \coloneqq X;\ \}\ \texttt{while}\ (a = 1);\ \mathcal{Y};\ \mathit{print}(a);\ \}\end{array} \ \Big\|\ \begin{array}{l}\texttt{while}\ (1)\ \{\ \mathcal{Y};\ X \coloneqq 2;\\ \quad \texttt{do}\ \{\ \mathcal{Y};\ b \coloneqq X;\ \}\ \texttt{while}\ (b = 2);\ \mathcal{Y};\ \mathit{print}(b);\ \}\end{array}$$

refines

$$\texttt{while}\ (1)\ \{\ \mathcal{Y};\ \mathit{print}(2);\ \}\ \Big\|\ \texttt{while}\ (1)\ \{\ \mathcal{Y};\ \mathit{print}(1);\ \}$$

under scheduler **fairness**

Takeaway: Syntax vs. Semantics

- **Syntactic** formulas & **Semantic** proof system
  - Unlike the syntax of logic, where the **proof system** is also syntactic
- **Syntax** is great in **flexibility**
  - In Nola, syntax **removes** the need for the **later modality**
  - Syntax can be designed for various use cases (e.g., **strafication**)

$P \leftrightarrow P$

- **Syntax** can be quite **extensible & semantic**
  - **Semantic propositions** can be embedded in formulas under later

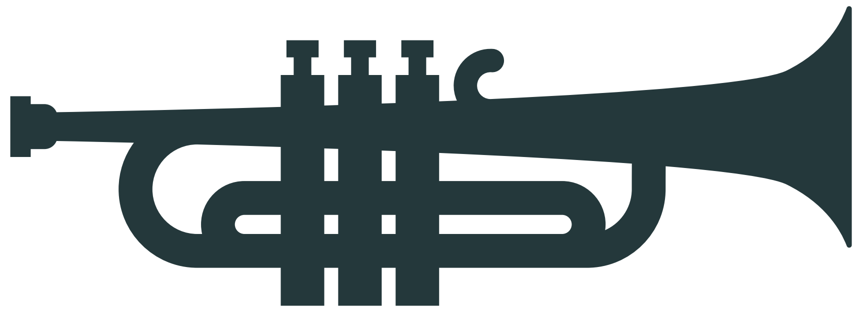
Summary of Nola

Ph.D. dissertation '23 & **PLDI '25**

- ✦ **Later-free** shared mutable state in separation logic
  - ▸ Higher-order ghost state, but clears the notorious **later** ▷
    - Great for **termination** & **liveness** verification
    - Refines Iris's **invariants** $\boxed{\triangleright P}$ & RustBelt's **borrows** $\&^{\alpha} \triangleright P$
  - ▸ Key idea: **Custom syntax** $P \in \mathit{Fml}$ for SL formulas
    - **Extensible** & **Semantic** SL props under later
  - ▸ Case study: **RustHalt**, revised RustHornBelt
  - ▸ Fully **mechanized** as a **library** of **Iris**

Iris

No later



github.com/hopv/nola

On my GitHub pages