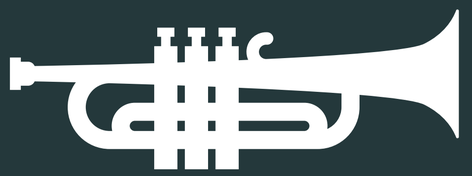


Nola: Later-free ghost state for verifying termination in Iris

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:

Hakubi
Kyoto Univ.

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

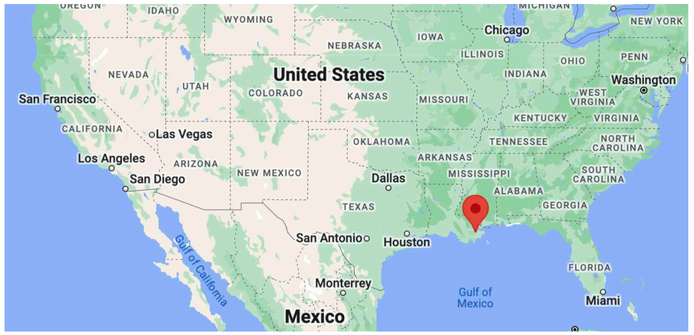
Rust’s **borrows** made “**pure**” by **prophecies**

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



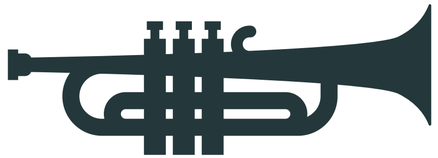
My latest work, Nola

Ph.D. dissertation ’23 & **PLDI** ’25

- ✦ **Later-free** shared mutable state in separation logic
  - ▸ **Higher-order ghost state**, but clears the notorious **later** ▷
    - 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 $\quad \forall n \in \mathbb{N}.\ \left[\mathtt{r} \mapsto n\right]\ \mathtt{decrloop(r)}\ \left[\lambda\_.\, \mathtt{r} \mapsto 0\right]$

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

Unbounded termination

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

Question

What about shared mutable state?

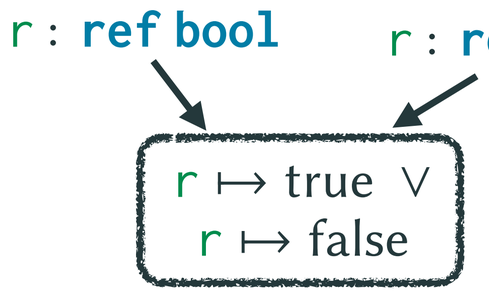
Traditional SL: **Mutable** state is **not sharable**

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**

Locial state that depends on SL assertions

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



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

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

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

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

Sad news: Naive later-free invariant is unsound

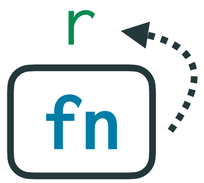
- **Naive rule causes unsound "infinite loops" in logic**

Naive access rule

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

⇩

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



Call

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

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

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

Existing approach: Later

- **Weakened invariant $\boxed{\triangleright P}$: The situation $\triangleright P$ always holds**
  - **Later** modality $\triangleright$: “Holds one index later” [Nakano ’20]
    - For **soundness**, Notorious **obstacle** of verification in **Iris**

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

Naive but **unsound**

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

Sound but **weakened**

Iris

Problem about later

- **Laters ▷ 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 ▷ blocks access** to esp. inside of **nested refs**

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

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

Blocks access!

Existing workaround & Its limitation

- **Step-indexing: Tie program steps with laters ▷**

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

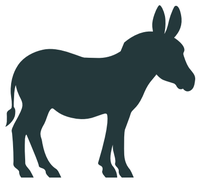
Wait for one program step,
then you can strip off one later

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

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

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

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



Recent approaches to termination & liveness

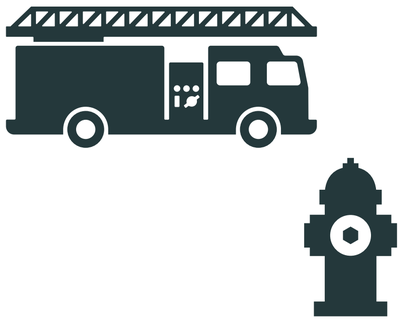
1. **Give up** invariants on non-timeless propositions
   - Many recent SLs for **advanced liveness properties**: Simuliris [Gäher+ '22], CCR [Song+ '23], Fairness Logic [Lee+ '23], etc.
2. **Finitely** bound program steps [Mevel+ '19]
   - **Bounded** termination $\$n$, not genuine liveness

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



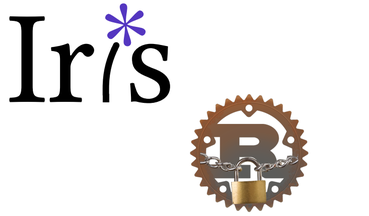
Is that even possible?

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** ▷
    - 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**



github.com/hopv/nola

On my GitHub pages

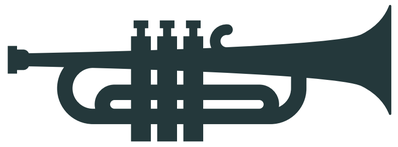
Key idea: Custom syntax for SL formulas

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

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

Iris

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



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

$P \in \mathit{Fml}$

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

Power of SL formulas

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

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

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

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

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

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

Any **semantic** SL props under later!

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

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

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

- **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**

$$\texttt{r} : \texttt{ref bool} \triangleq \boxed{\texttt{r} \mapsto \text{true} \vee \texttt{r} \mapsto \text{false}}$$

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

$$\left[\, [\![\texttt{r} : \texttt{ref (ref bool)}]\!] \,\right] *\texttt{r} \left[\, \lambda \texttt{s}.\ [\![\texttt{s} : \texttt{ref bool}]\!] \,\right]^{\text{Winv}\,[\![\ ]\!]}$$

Later-free access!

Verification example: Infinite list

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

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

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

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

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

Iris

Iteration

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

Termination!

By meta-logic induction

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

Custom view shifts & Hoare triples

- ✦ Enable **customizing** the **world satisfaction**
  - ▸ Or the **“mother invariant”** for **higher-order ghost state**

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

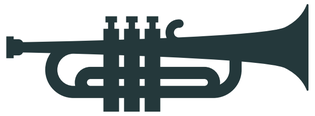
World satisfactions can be combined

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

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

Model of Nola's invariant

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



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

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

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

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

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

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

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

$$[\![\texttt{[}\top\texttt{]}\ \texttt{e}\ \texttt{[}\top\texttt{]}]\!] \triangleq^{?} \left[\top\right]\ \texttt{e}\ \left[\top\right]^{\text{Winv}\,[\![\ ]\!]}$$

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

- **Allows flexible construction for extra expressivity**
  - E.g., **Stratification** $[\![\ ]\!]_i : Fml_i \to iProp$

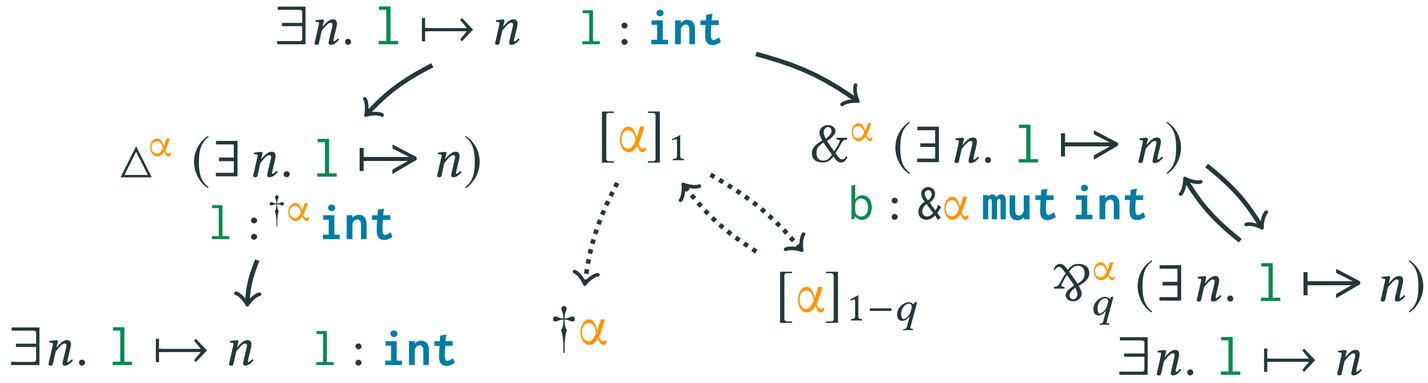
$$[\![\texttt{[}P\texttt{]}\ \texttt{e}\ \texttt{[}\Phi\texttt{]}]\!]_1 \triangleq \left[[\![P]\!]_1\right]\ \texttt{e}\ \left[[\![\Phi]\!]_1\right]^{\text{Winv}\,[\![\ ]\!]_0}$$

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);
```



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

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

Last challenge: Semantic alteration of syntax

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

Goal $\dfrac{\mathsf{T} \leq \mathsf{U} \quad \mathsf{U} \leq \mathsf{T}}{\mathtt{ref}\ \mathsf{T} \leq \mathtt{ref}\ \mathsf{U}}$

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

- **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**

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

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

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

$$\mathrm{der} \in Deriv$$

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′) } } }
```

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

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

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

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

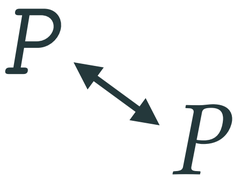
refines

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

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., **stratification**)



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

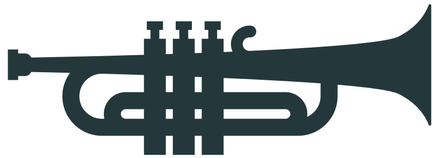
Summary of Nola

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