Pure Borrow: Linear Haskell Meets Rust-Style Borrowing

15th **Yusuke Matsushita** (Kyoto University) **Hiromi Ishii** (JIJ Inc.)

Published at ACM PLDI 2026

Big picture

Humans — Functional languages — Same results regardless of when & where computed — **Purity**

Computer — Imperative languages — Crucial for performance — **Mutation**

Ideal

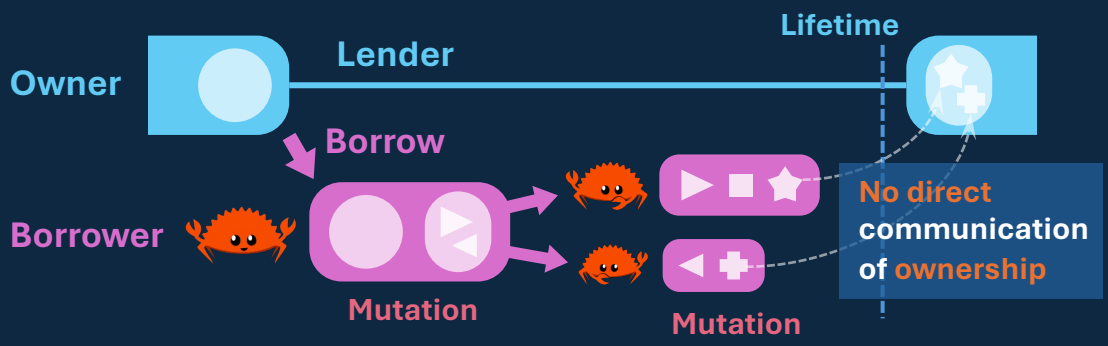
λ-calculus Church 1936 → **Haskell** '90 → **Linear Haskell** Spiwack+ '18 → **Pure Borrow**

C '72 → **Rust** 2015 → **Pure Borrow**

Ownership & borrowing controls mutation

Our work: **Rust**-style **borrowing** in **Linear Haskell** with **purity**!

Rust-style borrowing



Our Pure Borrow API

```
runBO :: Lin ⊸ (∀α. BO^α (End^α ⇒ a)) ⊸ a
parBO :: BO^α a ⊸ BO^α b ⊸ BO^α (a, b)

borrow :: Lin ⊸ a ⊸ (Mut^α a, Lend^α a)
drop :: Mut^α a ⊸ ()
reclaim :: End^α ⇒ Lend^α a ⊸ a
```

Monad BO^α = Computation possible only during lifetime α

cf. **ST monad** Launchbury+ '94

```
runST :: (∀s. ST^s a) → a
-- parST does not exist
```

```
modifyAt :: Int → (a ⊸ a) ⊸ Mut^α (Vector a) ⊸ BO^α (Mut^α (Vector a))
splitAt :: Int → Mut^α (Vector a) ⊸ (Mut^α (Vector a), Mut^α (Vector a))
```

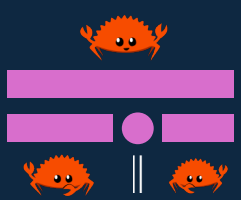
cf. traditionally `modifyAt_ :: Int → (a ⊸ a) ⊸ Vector a ⊸ Vector a`

Direct ownership threading

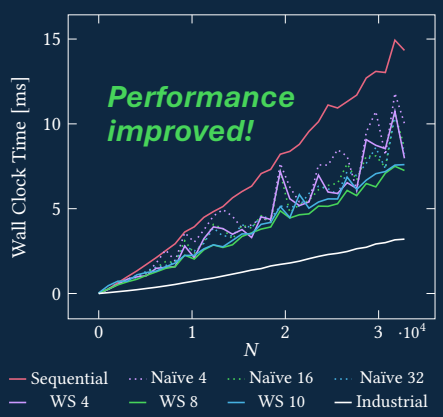
Case study

Parallel in-place quicksort

```
qsort :: Mut^α (Vector Int) ⊸ BO^α ()
qsort v = let (Ur n, v) = size v in
  if n ≤ 1 then pure (drop v) else do
    (Ur a, v) ← readAt (n // 2) v
    (lo, hi) ← divide a v 0 n
    drop <$> parBO (qsort lo) (qsort hi)
```



+ Advanced parallelism by **work stealing**



Metatheory

Design denotational operational semantics, which is pure

Conjecture (WIP): It matches **standard** operational semantics under **well-typedness**

Standard

$ref = \mathrm{Ref}\ \ell,\ now = \emptyset,\ E;\ \ell \mapsto x;\ main$

$\longrightarrow^* ref' = \mathrm{Ref}\ \ell,\ now' = \emptyset,\ E';\ \ell \mapsto y;\ main$

$E \triangleq x = 3,\ main = \mathrm{execBO}\ now\ (\mathrm{modify}\ (\lambda x \to x + 4)\ ref)$

$E' \triangleq x = 3,\ main = (now', ref')$

Denotational

$ref = \mathrm{Mut}^\delta\ (\mathrm{Ref}\ x),\ now = \{\},\ E;\ main$

$\longrightarrow^* ref' = \mathrm{Mut}^\delta\ (\mathrm{Ref}\ 7),\ now' = \{\delta \leftarrow 7\},\ E';\ main$

History of mutations is recorded to locally model lender reclaim