NYU

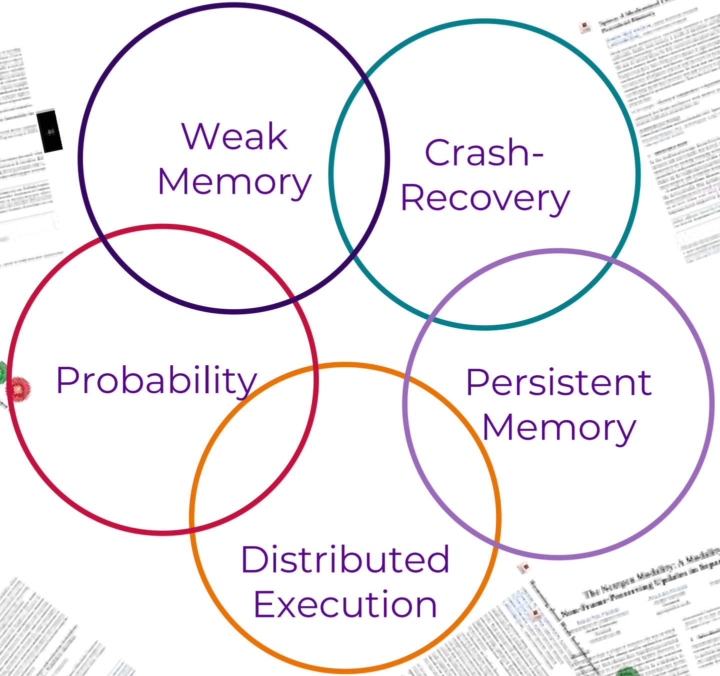
Building Extensible Program Logics with Effect Handlers

Zichen Zhang; Simon Gregersen; Joseph Tassarotti

New England System Verification Day
October 3, 2025

Program Logics for New Features

$$\{P\}\ e\ \{v.\,Q(v)\}$$



NYU

Traditional Approach

Operational Semantics

Reasoning Rules

$$e ::= x \mid e_1(e_2) \mid \cdots$$

Grammar

$$(e, \ldots) \longrightarrow (e', \ldots)$$

$$\{P\}\ e\ \{v.\, Q(v)\}$$

Soundness Theorem

NYU

Our approach

(0) A pure calculus + logic for it $\lambda_{\varnothing}$

(1) Write interpreter

using Effect Handlers

$\texttt{heap\_run}(e) \triangleq \cdots$

$\texttt{conc\_run}(e) \triangleq \cdots$

$\texttt{distr\_run}(e) \triangleq \cdots$

Logic Developer

(2) Prove specs for interpreters

using Hazel [de Vilhena & Pottier (2021)]

$\{\cdots\}\ \texttt{heap\_run}(e)\ \{\cdots\}$

$\Rightarrow \{\cdots\}\ \mathrm{ref}(v)\ \{\cdots\}$

$\Rightarrow \{\cdots\}\ !l\ \{\cdots\}$

$\Rightarrow \{\cdots\}\ l \leftarrow v\ \{\cdots\}$

(3) Verifying using rules derived from interpreter specs

Program Verifier

Effect Handlers

```
try
    let l = do(ALLOC, 1) in
    let x = do(LOAD, l) in
    assert(x = 1)
with
    | ALLOC(v, k)        ⇒ ··· ; k(···)
    | LOAD(l, k)         ⇒ ··· ; k(···)
    | STORE((l, v), k)   ⇒ ··· ; k(···)
    | return(v)          ⇒ v
end
```

Exception

+ *k* (continuation)

NYU

Our contribution

- Handler-based logics for
  - Concurrency
  - Crash recovery
  - Distributed execution
- Stronger reasoning rules
- Relational logic for refinement reasoning

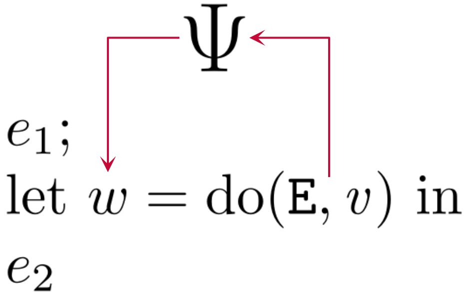
ROCQ Iris

Hazel Logic [de Vilhena & Pottier (2021)]

$$\{P\}\ e\ \langle|\Psi|\rangle\ \{v.\, Q(v)\}$$

Standard Hoare triple, plus

- Effects raised by e are handled by protocol Ψ.



Hazel Protocol

$$\{P\}\ e\ \langle\!|\Psi|\!\rangle\ \{v.\,Q(v)\}$$

$$\Psi ::= [P]\ (\mathtt{E}, v)\ [w.\,Q(w)] \mid \cdots$$

If raising an effect E with value *v* satisfying input condition *P*, the handler will return a value *w* satisfying output condition *Q*.

Ψ

Output satisfies Q

Input satisfies P

$$e_1;$$
$$\text{let } w = \text{do}(\mathtt{E}, v) \text{ in}$$
$$e_2$$

NYU

Reasoning in Hazel

- Effect raising rule

$$\frac{P \twoheadrightarrow P' \qquad \forall w.\, Q'(w) \twoheadrightarrow Q(w)}{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\ \Psi\ |\!\rangle\ \{w.\, Q(w)\}}$$

where $\Psi = [P']\ (\mathtt{E}, v)\ [w.\, Q'(w)]$

NYU

Reasoning in Hazel

$$\frac{P \mathrel{-\!\!*} P' \qquad \forall w.\, Q'(w) \mathrel{-\!\!*} Q(w)}{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!| \ [P']\ (\mathtt{E}, v)\ [w.\, Q'(w)]\ |\!\rangle\ \{w.\, Q(w)\}}$$

$$\mathsf{load} \triangleq [l \mapsto x]\ (\mathtt{LOAD}, l)\ [w.\, w = x * l \mapsto x]$$

$$!l \triangleq \mathrm{do}(\mathtt{LOAD}, l)$$

$$\{l \mapsto x\}\ !l\ \langle\!|\mathsf{load}|\!\rangle\ \{v.\, v = x * l \mapsto x\}$$

Sum Protocol

$$\{P\}\ e\ \langle\!|\Psi|\!\rangle\ \{v.\, Q(v)\}$$

$$\Psi \mathrel{+}::= \Psi_1 + \Psi_2$$

$$\frac{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_1|\!\rangle\ \{w.\, Q(w)\} \lor \{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_2|\!\rangle\ \{w.\, Q(w)\}}{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_1 + \Psi_2|\!\rangle\ \{w.\, Q(w)\}}$$

Sum Protocol

$$\frac{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_1|\!\rangle\ \{w.\, Q(w)\} \lor \{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_2|\!\rangle\ \{w.\, Q(w)\}}{\{P\}\ \mathrm{do}(\mathtt{E}, v)\ \langle\!|\Psi_1 + \Psi_2|\!\rangle\ \{w.\, Q(w)\}}$$

$$\mathsf{alloc} \triangleq [\mathsf{True}]\ (\mathtt{ALLOC}, x)\ [l.\, l \mapsto x]$$

$$\mathsf{load} \triangleq [l \mapsto x]\ (\mathtt{LOAD}, l)\ [w.\, w = x * l \mapsto x]$$

$$\mathsf{store} \triangleq [l \mapsto x]\ (\mathtt{STORE}, (l, y))\ [w.\, w = () * l \mapsto y]$$

$$!l \triangleq \mathrm{do}(\mathtt{LOAD}, l) \qquad \mathsf{heap} \triangleq \mathsf{alloc} + \mathsf{load} + \mathsf{store}$$

$$\{l \mapsto x\}\ !l\ \langle\!|\mathsf{heap}|\!\rangle\ \{v.\, v = x * l \mapsto x\}$$

Installing Handlers

$$\begin{array}{l}
\text{try try try} \\
\qquad\quad e \\
\qquad \text{with} \\
\qquad \mid \texttt{FORK}(f, h) \Rightarrow \cdots \mid \cdots \\
\qquad \text{end} \\
\quad \text{with} \\
\quad \mid \texttt{ALLOC}(v, h) \Rightarrow \cdots \mid \cdots \\
\quad \text{end} \\
\text{with} \\
\mid \texttt{READ}(_, h) \Rightarrow \cdots \mid \cdots \\
\text{end}
\end{array}$$

$$\{P\}\ e\ \langle|\mathsf{state} + \mathsf{heap} + \mathsf{conc}|\rangle\ \{v.\, Q(v)\}$$

conc_run

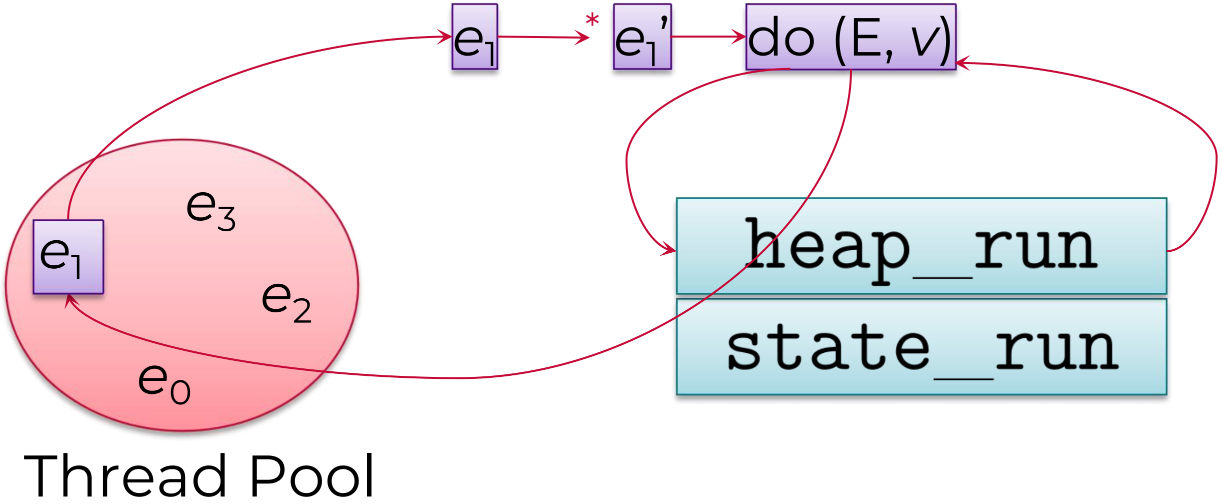
heap_run

state_run

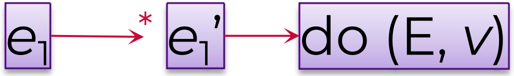
Concurrency

`conc_run`

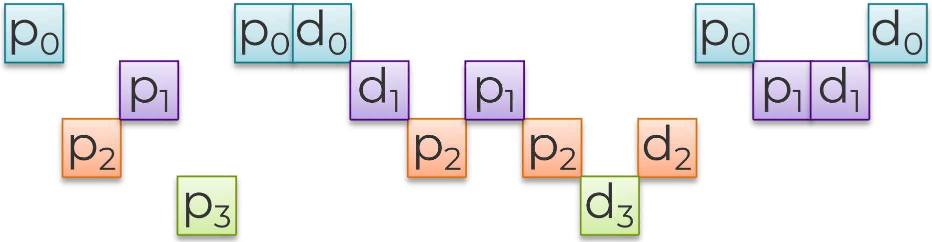
$\{P\}\ e\ \langle|\text{state} + \text{heap} + \text{conc}|\rangle\ \{v.\, Q(v)\}$



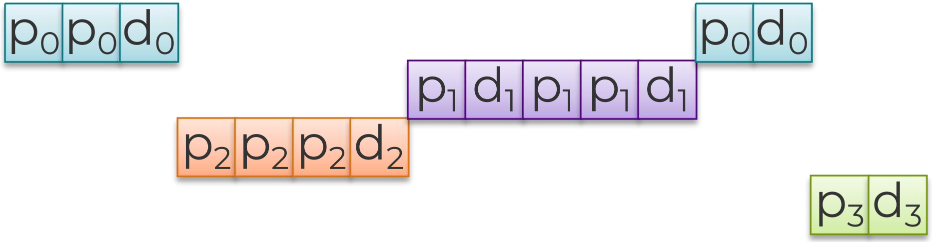
Execution Trace



Standard semantics
(preemptive)



Our semantics

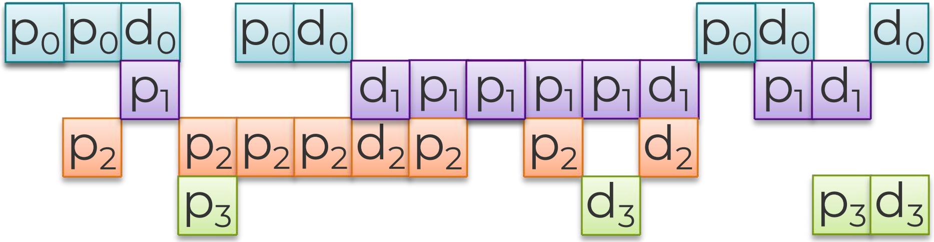


p = pure step d = effect step “do(...)”

Invariants

Standard semantics

Our semantics



Iris invariant access rule

Our invariant access rule

e = pure*, do(...)

$$\frac{\{P\}\ e\ \langle|\Psi|\rangle\ \{v.\, P * Q(v)\} \qquad \text{atomic}(e)}{\boxed{P} \vdash \{\text{True}\}\ e\ \langle|\Psi|\rangle\ \{v.\, Q(v)\}}$$

Is this rule sound w.r.t. standard semantics?

Relational Logic

Is this rule sound w.r.t. standard semantics?

$$\{P\}\ e_1 \precsim e_2\ \langle\!|\Psi_1; \Psi_2|\!\rangle\ \{v.\, Q(v)\}$$

$$\Downarrow$$

$$e_1 \lesssim e_2$$ (contextual refinement)

Informally, $\forall e_1 \longrightarrow^* v_1.\, \exists e_2 \longrightarrow^* v_2.\, v_1 \approx v_2$

e under standard semantics $\lesssim$ e under our semantics

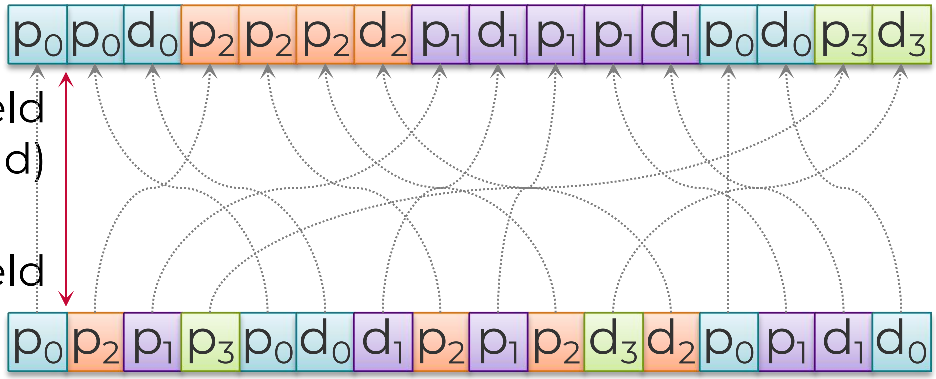
Refinement Proof

our semantics

cannot yield
(switch thread)

may yield

standard semantics



e under standard semantics $\lesssim$ e under our semantics

Refinement Proof

our semantics

emulated
standard semantics

standard semantics

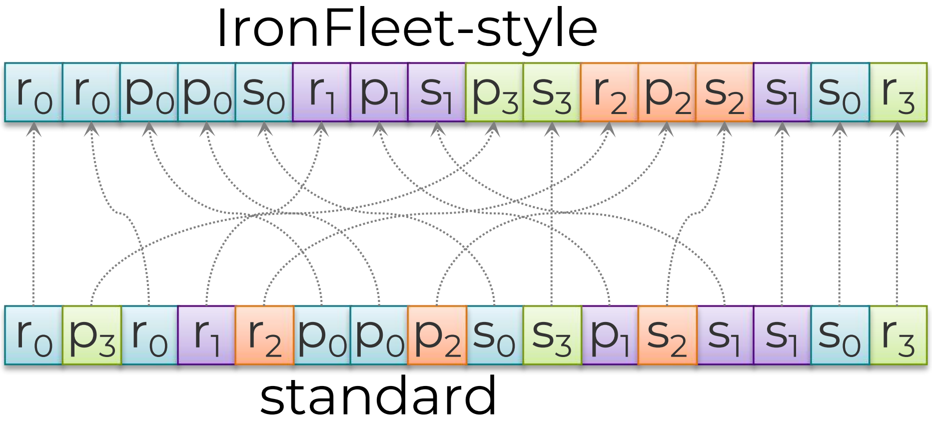
e under standard semantics $\lesssim$ e under our semantics

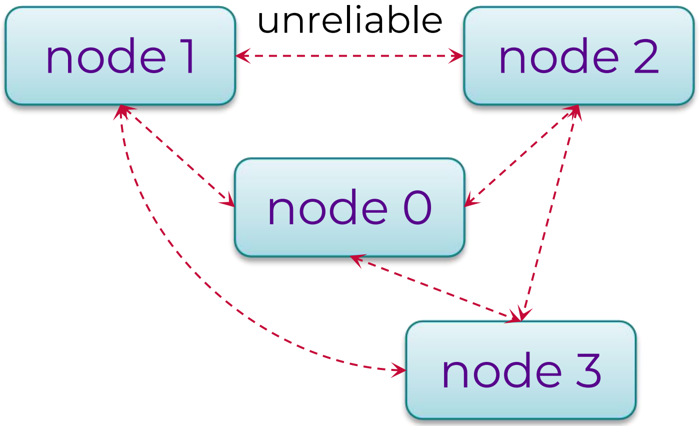
$\Uparrow$

yield $\lesssim$ nop (evidence accumulation)

Distributed System with IronFleet-style Atomic Block [Hawblitzel et al. (2015)]

r = receive   p = pure step   s = send





NYU

Distributed System with IronFleet-style Atomic Block [Hawblitzel et al. (2015)]

$$\frac{\{P\}\ e\ \langle|\Psi|\rangle\ \{v.\, P * Q(v)\} \qquad e = (\mathsf{recv} \mid \mathsf{pure})^*, \mathsf{send}}{\boxed{P} \vdash \{\mathsf{True}\}\ e\ \langle|\Psi|\rangle\ \{v.\, Q(v)\}}$$

$r_0\ r_0\ p_0\ p_0\ s_0\ r_1\ p_1\ s_1\ p_3\ s_3\ r_2\ p_2\ s_2\ s_1\ s_0\ r_3$

Summary

- Handler-based logics for

$$\{P\}\ e\ \langle\!|\Psi|\!\rangle\ \{v.\, Q(v)\}$$

  - Concurrency with stronger invariant rule
  - Distributed execution with IronFleet-style atomic blocks
  - *Crash recovery with Perennial-style crash invariants*
  - *Asynchronous disk based on crash-aware prophecy variables*
- Relational logic for refinement reasoning

$$\{P\}\ e_1 \precsim e_2\ \langle\!|\Psi_1; \Psi_2|\!\rangle\ \{v.\, Q(v)\}$$

Thank you for your attention!

NYU