# Scoped effects as parameterized algebraic theories

Cristina Matache, University of Edinburgh

Joint work with: Sam Lindley, Sean Moss, Sam Staton, Zhixuan Yang, Nick Wu

# Algebraic vs. non-algebraic effects

throw   throws an exception   $\rightsquigarrow$ algebraic

catch$(x, y)$   handles exceptions in $x$ with $y$   $\rightsquigarrow$ non-algebraic

Catch is not algebraic:

$$\text{catch}(x, y) \gg\!= k \quad \neq \quad \text{catch}(x \gg\!= k,\; y \gg\!= k)$$

catch is a <u>handler</u> for throw [Plotkin & Pretnar '09, '13]

Question: how do we treat catch as an <u>operation</u>?

# Outline

1. Algebraic effects
2. Scoped effects
3. Parameterized algebraic theories
4. Scoped effects as parameterized theories (contribution)

# Algebraic Effects: Explicit nondeterminism (backtracking)

[Plotkin & Pretnar '09, '13]

Operations:

$or(x, y)$ choice

$fail$ failure

Equations:

$or(or(x, y), z) = or(x, or(y, z))$

$or(x, fail) = or(fail, x) = x$

Generic effects:

$\underline{or} : unit \to bool$

$\underline{fail} : unit \to 0$

$$\begin{cases} or(x, y) = \text{if } \underline{or}() \text{ then } x \text{ else } y \\ \underline{or}() = or(true, false) \end{cases}$$

# Algebraic Effects: Explicit nondeterminism (backtracking)

Intended model, for a return type $A$:

Carrier $\quad \mathrm{List}(A)$

Operations $\quad [\![\mathrm{or}]\!] : \mathrm{List}(A)^2 \longrightarrow \mathrm{List}(A)$, $\quad [\![\mathrm{or}]\!](x, y) = x +\!\!+ y$

$\quad [\![\mathrm{fail}]\!] : 1 \longrightarrow \mathrm{List}(A)$, $\quad [\![\mathrm{fail}]\!]() = [\,]$

- $\mathrm{List}(A)$ is a free model on $A$
- List extends to a strong monad
  - implementation
  - denotational semantics

# Algebraic effects have:

- An equational reasoning system i.e. algebraic theories
- with semantic models
- s.t. equality in the theory is sound and complete
- Correspondence between theories and monads on Set

Question:

Equational reasoning for non-algebraic effects, like scoped effects?

E.g. $\mathsf{catch}(x, y) \gg= k \neq \mathsf{catch}(x \gg= k, y \gg= k)$

# Outline

1. Algebraic effects
2. Scoped effects
3. Parameterized algebraic theories
4. Scoped effects as parameterized theories (Contribution)

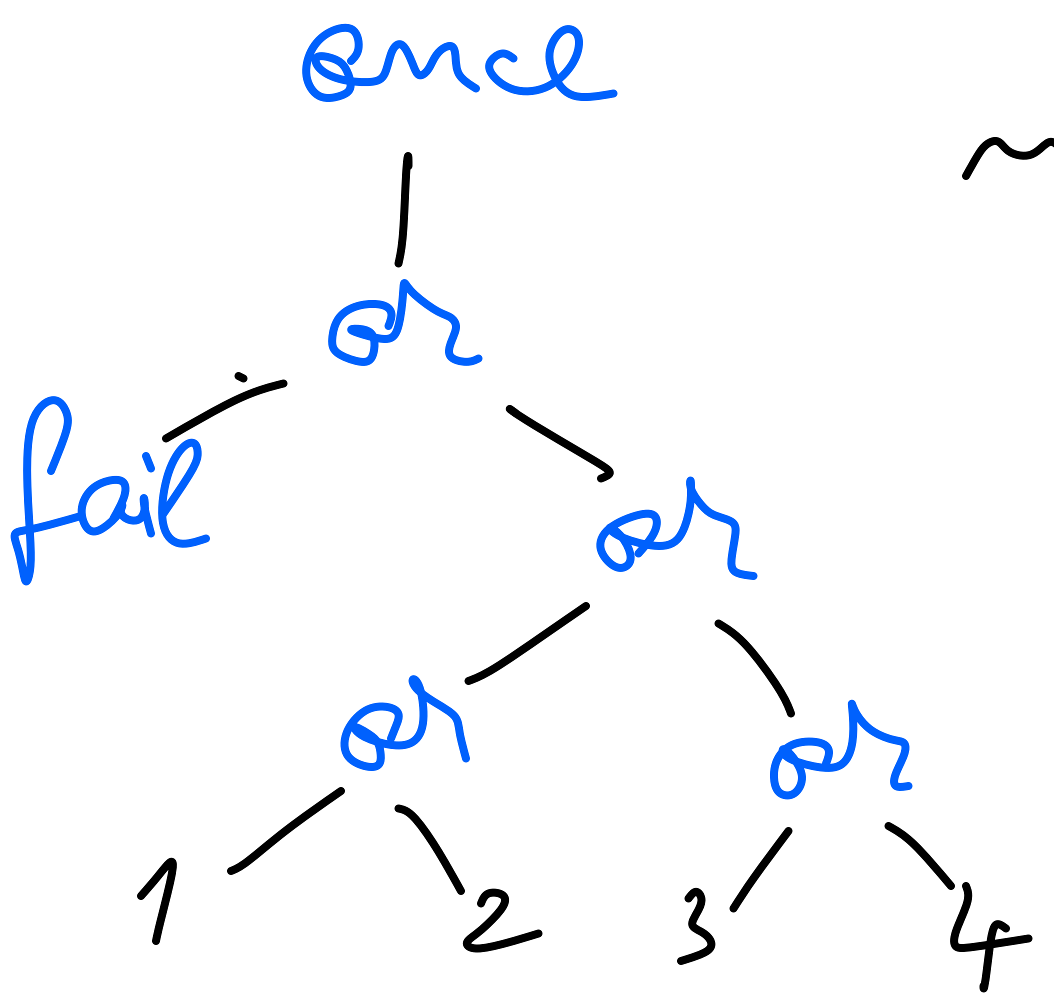
# Scoped effects: Nondeterminism with once [Wu et al. '14]

Operations: $or(x, y)$, $fail$

$once(x)$ chooses first non-failing branch of $x$

Example:

$once(or(fail, or(or(1,2), or(3,4))))$



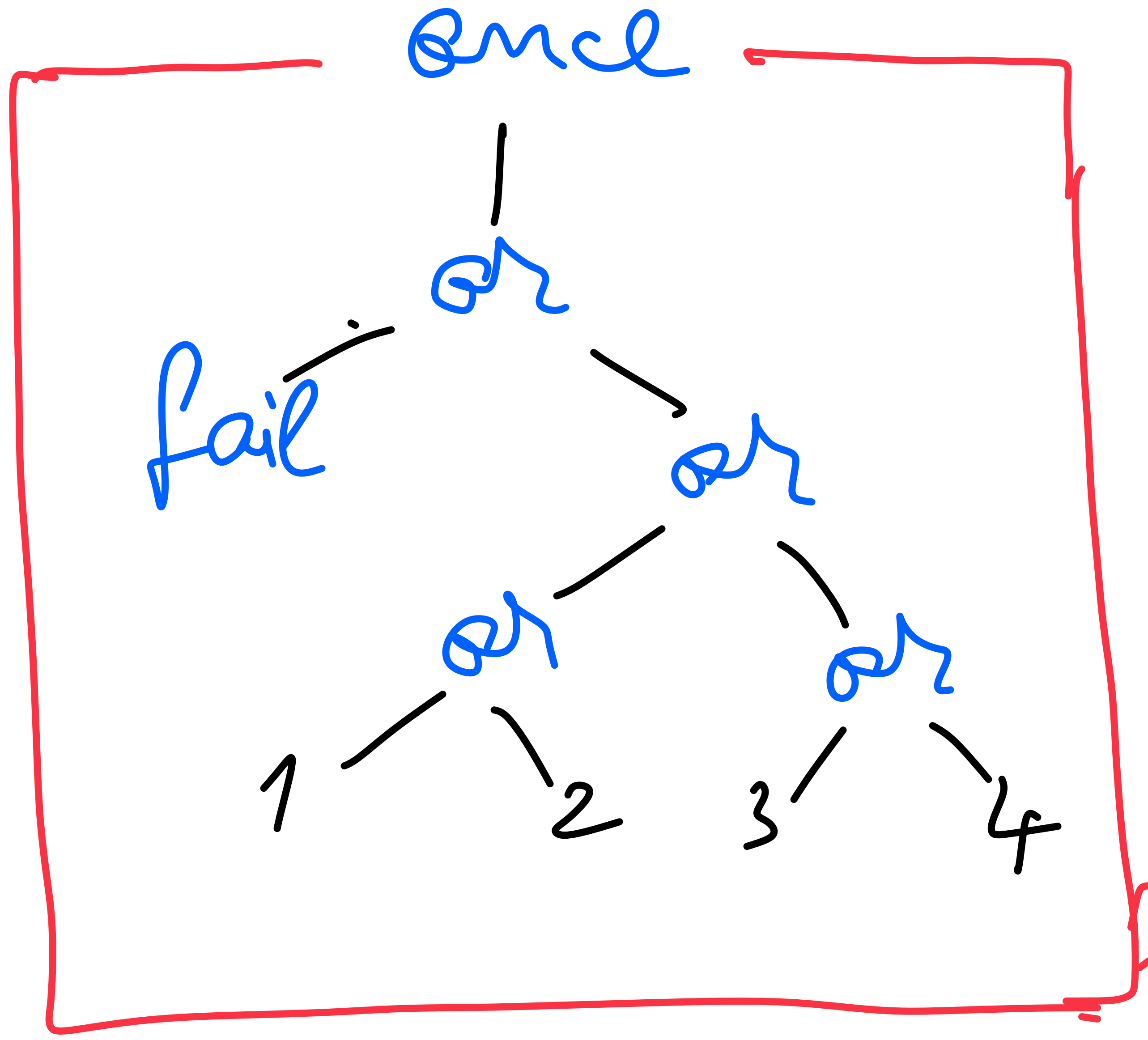

$\rightsquigarrow 1$

Another scoped operation: $catch(x, y)$

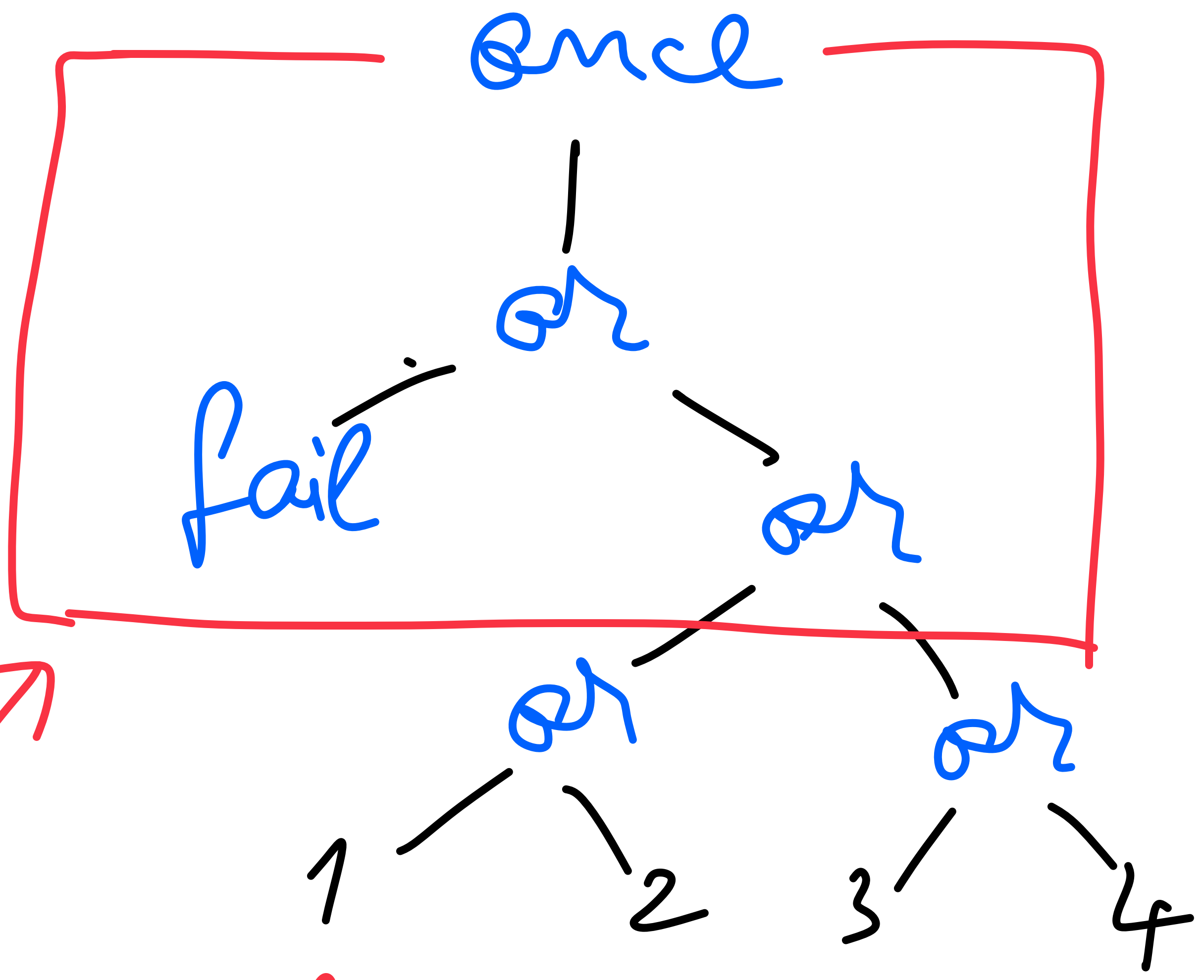
# Scoped effects: Nondeterminism with once

Once is not algebraic:

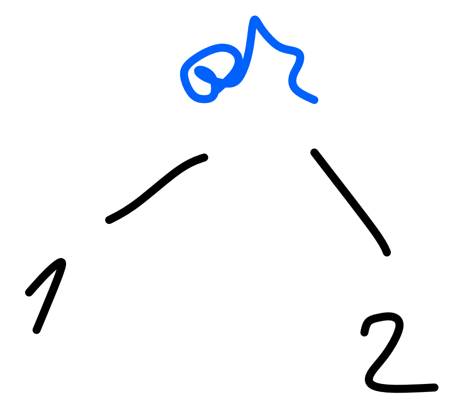
$$\mathrm{once}(\mathrm{or}(\mathrm{fail}, \mathrm{or}(1,3)) \gg\!= \lambda x.\, \mathrm{or}(x, x+1)) \neq \mathrm{once}(\mathrm{or}(\mathrm{fail}, \mathrm{or}(1,3))) \gg\!= \lambda x.\, \mathrm{or}(x, x+1)$$




$\rightsquigarrow 1$




$\rightsquigarrow$




scope of once is delimited

# Scoped effects : background

- Scoped effects look like handlers of algebraic effects

- Handling scoped effects ⇝ free monads from a signature

E.g. [Wu et.al.'14], [Piróg et.al LICS'18], [Yang et.al. ESOP'22, ICFP'23]

## Our contribution:

Equational theories for scoped effects, that generate monads.

# Outline

1. Algebraic effects
2. Scoped effects
3. Parameterized algebraic theories
4. Scoped effects as parameterized theories (Contribution)

# Parameterized algebraic theories

[Staton FoSSaCS'13, LiCS'13, POPL'15]

- Extend algebraic theories with binding of abstract parameters
- Uniform syntax for axiomatizing e.g.:

| Example | Parameters |
| --- | --- |
| local state | location names |

$\mathsf{new}_0(a.x(a))$  create new location a, containing 0

↑ fresh parameter, bound

$\mathsf{read}(a, x, y)$  read the bit stored in a

↑ free parameter

+ other operations and equations

# Parameterized algebraic theories [Staton FoSSACS'13, LICS'13, POPL'15]

- Extend algebraic theories with binding of abstract parameters
- Uniform syntax for axiomatizing e.g.:

| Example | Parameters |
|---|---|
| local state | location names |
| $\pi$-calculus | channels |
| quantum computation | qubits |

- Have canonical semantic status, similar to algebraic theories

# Outline

1. Algebraic effects
2. Scoped effects
3. Parameterized algebraic theories
4. Scoped effects as parameterized theories (Contribution)

# Scoped effects as parameterized algebraic theories

Contribution: Equational theories for scoped effects

Idea:

- A scoped effect = a parameterized theory
  where parameters are names of scopes

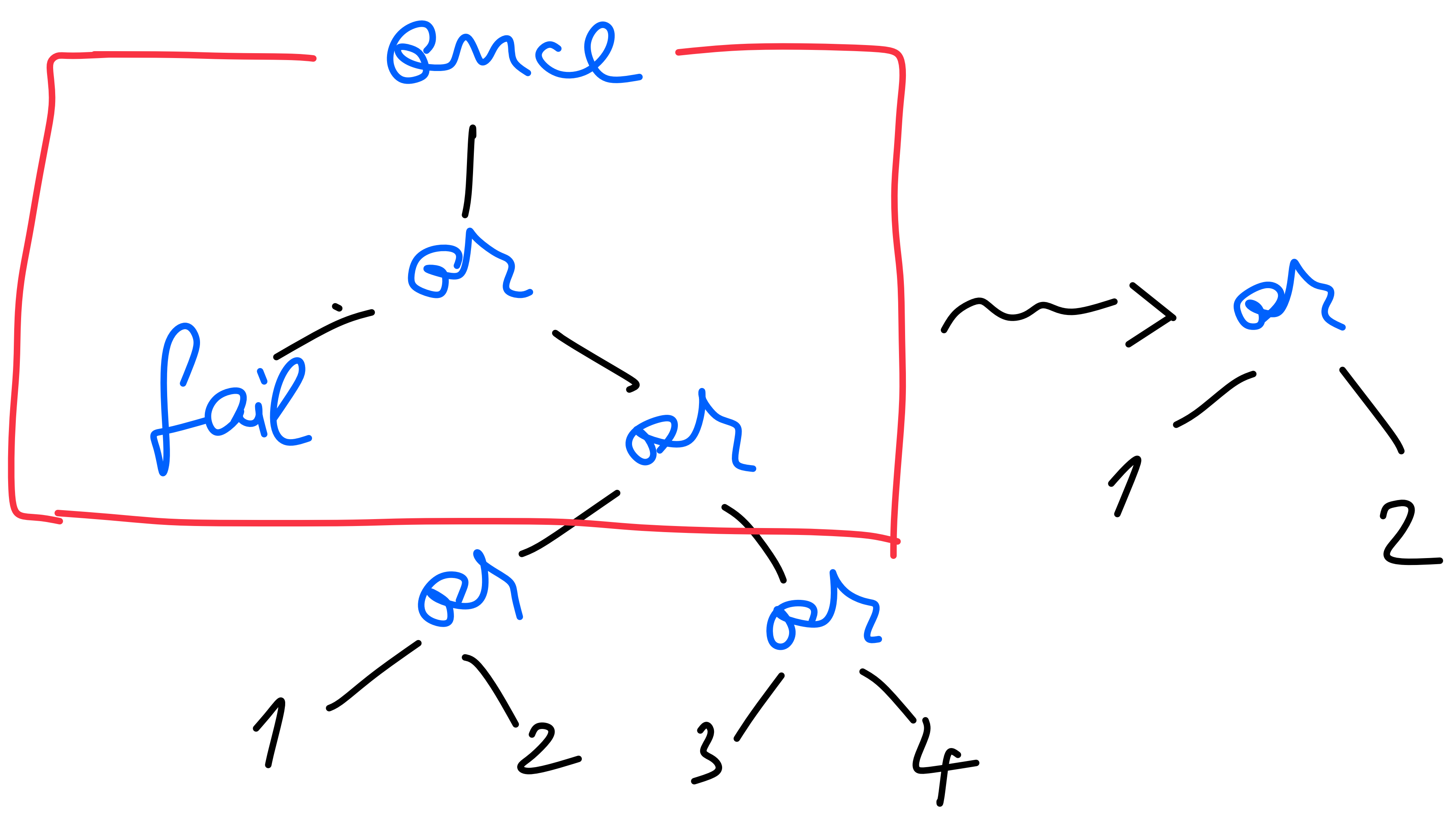
- Opening/closing scopes is explicit

# Nondeterminism with once as a parameterized theory

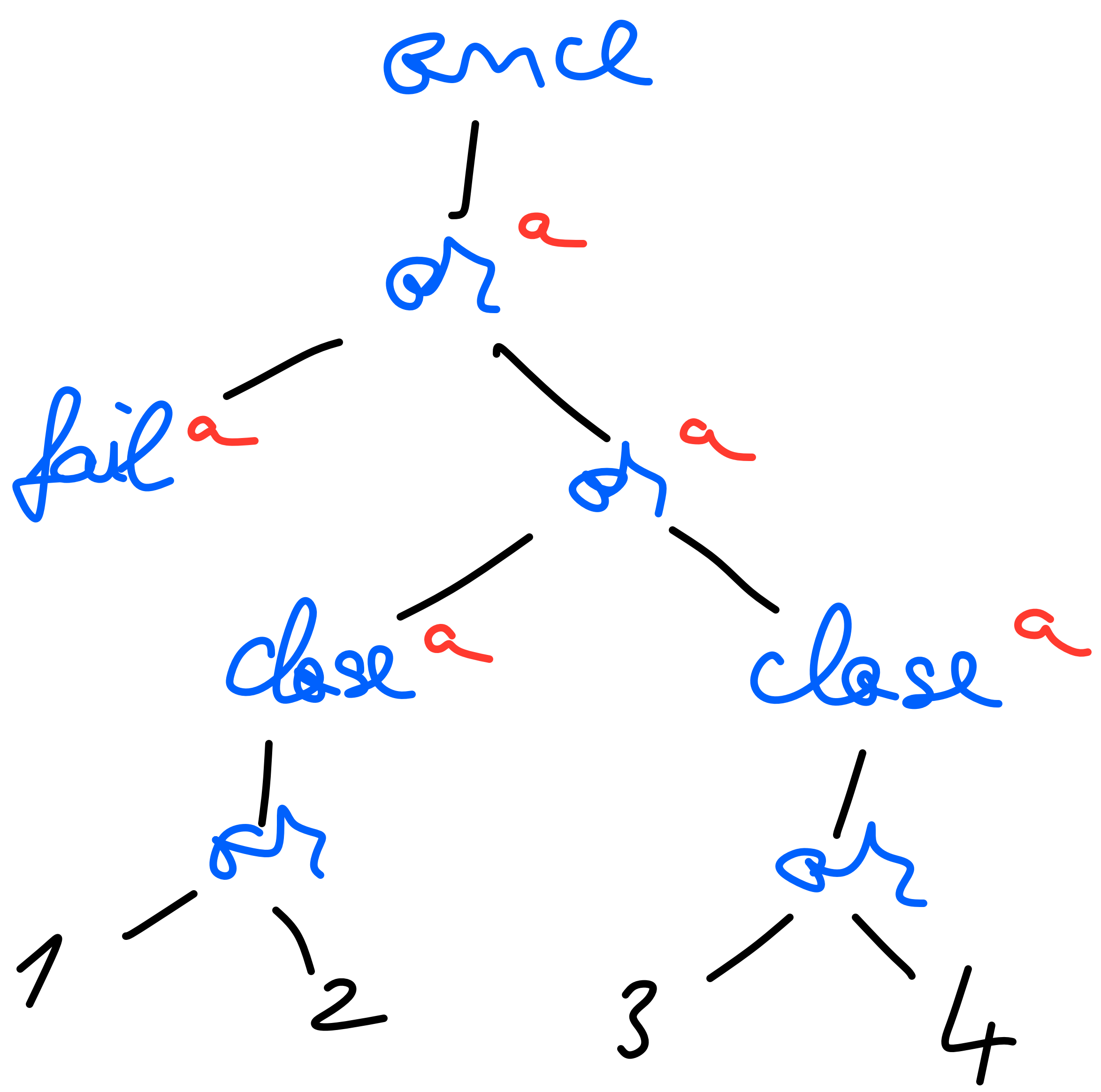
Operations: $\mathit{or}(x, y)$, $\mathit{fail}$, $\mathit{once}(a.\, x(a))$, $\mathit{close}(a, y)$

$\mathit{once}(\mathit{or}(\mathit{fail}, \mathit{or}(1,3))) \gg\!= \lambda x.\, \mathit{or}(x, x+1)$

$\mathit{once}(a.\, \mathit{or}(\mathit{fail}, \mathit{or}(\mathit{close}(a, \mathit{or}(1,2)), \mathit{close}(a, \mathit{or}(3,4)))))$




becomes



# Equations for nondeterminism with once

**Explicit nondeterminism**

$$or(or(x, y), z) = or(x, or(y, z))$$

$$or(x, fail) = x$$

$$or(fail, x) = x$$

**Once/close**

$$once(a.\,close(a, x)) = x$$

$$once(a.\,fail) = fail$$

$$once(a.\,or(x(a), x(a))) = once(a.\,x(a))$$

$$once(a.\,or(close(a, x), y(a))) = x$$

Example:

$$once(a.\,or(fail, or(close(a, or(1,2)), close(a, or(3,4))))) = or(1, 2)$$

# Equations for nondeterminism with once

**Explicit nondeterminism**

$$\mathrm{or}(\mathrm{or}(x,y),z) = \mathrm{or}(x,\mathrm{or}(y,z))$$

$$\mathrm{or}(x,\mathrm{fail}) = x$$

$$\mathrm{or}(\mathrm{fail},x) = x$$

**Once/close**

$$\mathrm{once}(a.\mathrm{close}(a,x)) = x$$

$$\mathrm{once}(a.\,\mathrm{fail}) = \mathrm{fail}$$

$$\mathrm{once}(a.\mathrm{or}(x(a),x(a))) = \mathrm{once}(a.x(a))$$

$$\mathrm{once}(a.\mathrm{or}(\mathrm{close}(a,x),y(a))) = x$$

**Theorem.** The model of [LICS'18] for nondeterminism with once, where roughly $[\![\mathrm{once}]\!] : \mathrm{List}(\mathrm{List}) \longrightarrow \mathrm{List}$, is a free model for the parameterized theory above.

## Other scoped effects we studied: [Under review]

- Exception catching
- Mutable state with local values

} Equational characterizations using parameterized theories

## Future work

- More examples
- Programming in generic effect style for scoped effects
- Monad-theory correspondence for scoped effects (by restricting the correspondence for parameterized theories)