Equational theories for scoped effects

Cristina Matache

MSFP Workshop, 18 July 2026

University of Birmingham

Joint work with Sam Lindley, Sean Moss, Sam Staton, Nicolas Wu, and Zhixuan Yang

Introduction

Effects represent impure computation in a functional programming language, for example:

- nondeterminism,
- manipulating memory,
- taking input from a keyboard.

Moggi gave denotational semantics to effects using **strong monads** on the category of sets [Moggi'89,'91] $\Longrightarrow$ Monads for programming in Haskell.

Introduction: Plotkin's and Power's framework of algebraic effects

Idea: effects are realized by operations and equations that **generate** a monad, but are not identified with the monad.

- Operations produce the effects (e.g. reading and writing to memory, nondeterministic choice).
- Equations specify the behaviour of these operations.

Theorem [Lawvere, Linton]

To give a finitary monad on the category of sets is equivalent to giving an algebraic theory (in the sense of universal algebra, or Lawvere theories).

Introduction: Plotkin's and Power's framework of algebraic effects

Idea: effects are realized by operations and equations that **generate** a monad, but are not identified with the monad.

- Operations produce the effects (e.g. reading and writing to memory, nondeterministic choice).
- Equations specify the behaviour of these operations.

Example: the List monad is presented by two operations:

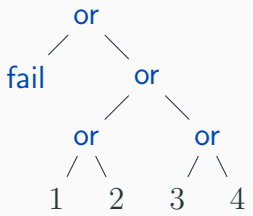
- $+\!\!+$, list concatenation, which is associative, and
- $[]$ the empty list, unit element for $+\!\!+$.

Introduction: algebraic vs scoped operations

Algebraic operations commute with sequencing (monadic bind).

Example: nondeterministic choice and failure.
The following two terms perform the same computation:

$$\mathsf{or}\big(\mathsf{fail},\ \mathsf{or}(1 \gg\!= \lambda x.\, \mathsf{or}(x, x+1),\ 3 \gg\!= \lambda x.\, \mathsf{or}(x, x+1))\big) \qquad \mathsf{or}\big(\mathsf{fail},\ \mathsf{or}(1,3)\big) \gg\!= \lambda x.\, \mathsf{or}(x, x+1)$$

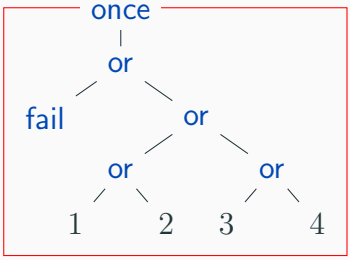


Introduction: algebraic vs scoped operations

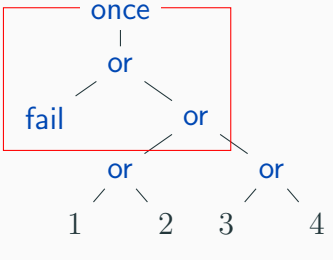
Scoped operations do not commute with sequencing [Wu et al'14].

Example: once chooses the first branch that does not fail.
The scope of once is delimited.

$\mathsf{once}(\mathsf{or}(\mathsf{fail},\ \mathsf{or}(\mathsf{or}(1,2),\mathsf{or}(3,4)))) \rightsquigarrow 1$



$\mathsf{once}(\mathsf{or}(\mathsf{fail},\ \mathsf{or}(1,3))) \gg\!= \lambda x.\ \mathsf{or}(x, x+1) \rightsquigarrow \mathsf{or}(1,2)$



Introduction: algebraic vs scoped operations

Scoped operations do not commute with sequencing [Wu et al'14].

Example: catching exceptions

$$\mathsf{catch}(x, y) \gg\!= k \neq \mathsf{catch}(x \gg\!= k, y \gg\!= k)$$

Scoped effects don't fit in the framework of algebraic effects.

In this talk

[Lindley et al ESOP'24, Matache et al TOPLAS'25]

- Clarify what scoped effects are using monads and algebraic theories.
- Recover complete equational reasoning for scoped effects.

Outline

Algebraic theories

An **algebraic theory** $\mathcal{T} = (\Sigma, E)$ consists of a signature $\Sigma : \mathcal{O} \to \mathbb{N}$ of operations, and a set of equations E.

Example: explicit nondeterminism (backtracking) [Plotkin & Pretnar'09, '13]

$\mathsf{or} : 2$ choice

$\mathsf{fail} : 0$ failure

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

$$x \vdash \mathsf{or}(\mathsf{fail}, x) = \mathsf{or}(x, \mathsf{fail}) = x$$

In direct style:

$$\underline{\mathsf{or}} : \mathsf{unit} \to \mathsf{bool} \qquad \underline{\mathsf{fail}} : \mathsf{unit} \to \mathsf{empty}$$

Translation between direct style and algebraic operations:

$$\mathsf{or}(x, y) = \mathsf{if}\ \underline{\mathsf{or}}()\ \mathsf{then}\ x\ \mathsf{else}\ y \qquad \underline{\mathsf{or}}() = \mathsf{or}(\mathsf{true}, \mathsf{false})$$

Algebraic theories

Example: explicit nondeterminism (backtracking) [Plotkin & Pretnar'09, '13]

$\mathsf{or} : 2$	choice	$x, y, z \vdash \mathsf{or}(x, \mathsf{or}(y, z)) = \mathsf{or}(\mathsf{or}(x, y), z)$
$\mathsf{fail} : 0$	failure	$x \vdash \mathsf{or}(\mathsf{fail}, x) = \mathsf{or}(x, \mathsf{fail}) = x$

Model of an algebraic theory = a set (carrier) + interpretations for the operations, satisfying the equations.

Fix a set A. The intended model for the theory of nondeterminism is:

- Carrier: the set $\mathsf{List}(A)$
- Operations:

$$[\![\mathsf{or}]\!] : \mathsf{List}(A)^2 \to \mathsf{List}(A), \qquad [\![\mathsf{or}]\!](l_1, l_2) = l_1 +\!\!+ l_2$$
$$[\![\mathsf{fail}]\!] : 1 \to \mathsf{List}(A), \qquad [\![\mathsf{fail}]\!]() = []$$

Algebraic theories

- $\mathsf{List}(A)$ is the **free model** on A of the theory of nondeterminism.
- $\mathsf{List}(-)$ extends to a monad on Set (part of the finitary monad - algebraic theory correspondence).
- Consequence of freeness: the equations in the theory are **sound and complete** with respect to the $\mathsf{List}(A)$ model.

Algebraic theories

- $\mathsf{List}(A)$ is the **free model** on A of the theory of nondeterminism.
- $\mathsf{List}(-)$ extends to a monad on Set (part of the finitary monad - algebraic theory correspondence).
- The List monad **supports** the operations from the theory, meaning e.g.:

$$[\![\mathsf{or}]\!] : \mathsf{List}(A) \times \mathsf{List}(A) \to \mathsf{List}(A)$$

is natural in A and commutes with $\gg\!=$ of the List monad, e.g.

$$\big([\![\mathsf{or}]\!]([1],[3]) \gg\!= k\big) = [\![\mathsf{or}]\!]([1] \gg\!= k,\ [3] \gg\!= k) = [1,2,3,4]$$

where $k = \lambda x.[x, x+1]$.

Outline

What is a scoped effect? Four different views

(1) A **scoped operation** is a handler for an algebraic effect. Semantically, an algebra for a monad on Set. [Plotkin & Pretnar'09, '13]

(2) Given a monad T on Set, a **scoped operation** sc : k is a natural transformation $T^k \to T$. [Wu et al'14, Yang & Wu ICFP'23].

(3) A **scoped effect** is determined by an endofunctor on $\mathsf{Set}^{|\mathbb{N}|}$, ($|\mathbb{N}|$ is a discrete category), and taking the free monad on this endofunctor. [Piróg et al LICS'18]

(4) A **scoped effect** is determined by an endofunctor on $\mathsf{Set}^{\mathsf{FinSet}}$, and taking the free monad on this endofunctor. [Wu et al'14, Yang et al ESOP'22]

Nondeterminism with once as a scoped effect [Wu et. al.'14]

(2) Given a monad T on Set, a **scoped operation** sc : k is a natural transformation $T^k \to T$. [Wu et al'14, Yang & Wu ICFP'23].

Recall that once chooses the first non-failing branch of a computation.
It fits the definition of a scoped operation.

The List monad supports the natural transformation:

$$[\![\text{once}]\!] : \mathsf{List}(A) \to \mathsf{List}(A) \qquad [\![\text{once}]\!]([a, \dots]) = [a] \qquad [\![\text{once}]\!]([]) = []$$

which does not commute with $\gg\!=$

$$\big([\![\text{once}]\!]([\![\text{or}]\!]([1],[3]))\big) \gg\!= k = \big([1] \gg\!= k\big) = [1,2]$$

$$[\![\text{once}]\!]\big([\![\text{or}]\!]([1],[3]) \gg\!= k\big) = [\![\text{once}]\!]([1,2,3,4]) = [1]$$

where $k = \lambda x.\, [x, x+1]$.

Our proposal: a scoped effect is a theory

(5) A **scoped effect** is an algebraic theory of a suitable kind.

This definition provides **equational reasoning** for scoped effects.

- This definition agrees with (2) and (3) for examples of scoped effects:
  - nondeterminism (backtracking) with once,
  - exception catching,
  - state with local values
  - nondeterminism (backtracking) with cut.
- Theorems connecting definition (5) with (2) and (3) in general.

Outline

Scoped effects as parameterized algebraic theories

Parameters = names of scopes

A scoped effect = a theory with **ordered, linear** parameters

Example: nondeterminism with once

$\mathsf{or}(x, y)$	choice
fail	failure
$\mathsf{once}(a.x(a))$	open a new scope named a, continue as $x(a)$; a is bound
$\mathsf{close}(a;\ y)$	close the scope a, y cannot use a anymore; a is free

A term has two contexts. The context of parameters is ordered and linear.

Example:

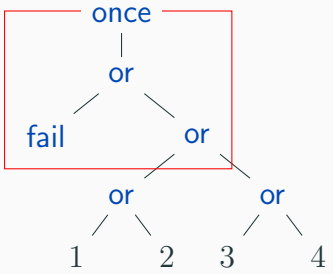
$$x : 1 \mid b \vdash \mathsf{close}(b;\ \mathsf{once}(a.x(a)))$$

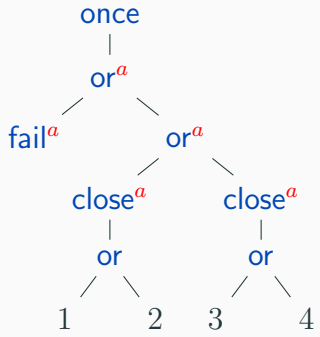
Nondeterminism with once as a parameterized theory

Closing a scope becomes an explicit operation.

$$\mathsf{once}(\mathsf{or}(\mathsf{fail},\ \mathsf{or}(1,3))) \gg\!= \lambda x.\, \mathsf{or}(x, x+1) \rightsquigarrow \mathsf{or}(1,2)$$

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





Equations for nondeterminism with once

Explicit nondeterminism

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

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

Once/close

$$\mathsf{once}(a.\mathsf{fail}) = \mathsf{fail} \quad (1)$$

$$\mathsf{once}(a.\mathsf{or}(x(a),\ x(a))) = \mathsf{once}(a.x(a)) \quad (2)$$

$$\mathsf{once}\big(a.\mathsf{or}(\mathsf{close}(a;\ x),\ y(a))\big) = x \quad (3)$$

We can derive:

$$\mathsf{once}(a.\mathsf{close}(a;\ x)) = x$$

We can prove using the equations:

$$\mathsf{once}\big(a.\mathsf{or}(\mathsf{fail},\ \mathsf{or}(\mathsf{close}(a;\ \mathsf{or}(1, 2)),\ \mathsf{close}(a;\ \mathsf{or}(3, 4))))\big) = \mathsf{or}(1, 2)$$

Parameterized algebraic theories [Staton FoSSaCS'13]

- Extend plain algebraic theories with binding.
- Have a correspondence with certain strong monads on a functor category. In our case: $\mathsf{Set}^{|\mathbb{N}|}$, where $|\mathbb{N}|$ is discrete.
  Via the free model construction.
- Provide an equational reasoning system.

Our proposed definition

A **scoped theory** is a parameterized theory with linear, ordered parameters with certain restrictions on the signature (to match examples).

E.g. operations can bind at most one parameter in each argument;
there is a close operation that consumes one parameter.

Models for explicit nondeterminism with once

Model = an object from $\mathsf{Set}^{|\mathbb{N}|}$ (carrier) + interpretations for the operations.

The **free model** on $A = (A_0, \emptyset, \emptyset, \dots) \in \mathsf{Set}^{|\mathbb{N}|}$ has:

- Carrier: the sequence $TA(n) = \mathsf{List}^{n+1}(A_0)$, for $n \in \mathbb{N}$
- Operations:

$$
\begin{array}{lll}
[\![\mathsf{once}]\!]_n : TA(n+1) \to TA(n) & [\![\mathsf{once}]\!]_n([a, \dots]) = a, & [\![\mathsf{once}]\!]_n([]) = [] \\
[\![\mathsf{close}]\!]_n : TA(n) \to TA(n+1) & [\![\mathsf{close}]\!]_n(l) = [l] & \\
[\![\mathsf{or}]\!]_n : TA(n)^2 \to TA(n) & [\![\mathsf{or}]\!]_n(l_1, l_2) = l_1 +\!\!+\, l_2 & \\
[\![\mathsf{fail}]\!]_n : 1 \to TA(n) & [\![\mathsf{fail}]\!]_n() = [] &
\end{array}
$$

Recall: $\mathsf{once}(a.x(a))$ $\mathsf{close}(a;\, y)$ $\mathsf{or}(x, y)$ fail

Relation to previous definitions of scoped effects

(2) Given a monad T on Set, a **scoped operation** sc : k is a natural transformation $T^k \to T$. [Wu et al'14, Yang & Wu ICFP'23].

Theorem

From the monad on $\mathsf{Set}^{|\mathbb{N}|}$ generated by the theory of nondeterminism with once, we can recover the List monad on Set and the scoped operation $[\![\mathsf{once}]\!] : \mathsf{List} \to \mathsf{List}$.

At least two inequivalent scoped theories that recover $(\mathsf{List}, [\![\mathsf{once}]\!])$. We chose an intuitive one.

More generally, from Def (2) we can always construct a (maximal) parameterized theory, then recover the data of Def (2) from it.

Relation to previous definitions of scoped effects

(3) A **scoped effect** is determined by an endofunctor on $\mathsf{Set}^{|\mathbb{N}|}$, ($|\mathbb{N}|$ is a discrete category), and taking the free monad on this endofunctor. [Piróg et al LICS'18]

A model is an algebra for this free monad.

Theorem

The **free model** on $A = (A_0, \emptyset, \emptyset, \dots) \in \mathsf{Set}^{|\mathbb{N}|}$ for the theory of nondeterminism with once, is the monad algebra from [Piróg et al LICS'18].

We gave a framework in which **complete equational axiomatization** of their model is possible.

More generally, given a scoped theory with no equations, the induced monad on $\mathsf{Set}^{|\mathbb{N}|}$ is isomorphic to the free monad of [Piróg et al LICS'18].

Outline

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

Different applications of parameterized theories, by varying the structural rules of parameters:

Example	Parameters	Models in
name generation	names	$\mathsf{Set}^{\mathsf{Fin}}$
local state	location names	
π-calculus (fragment)	communication channels	
predicate logic	individuals	
quantum computation	qubits (linear)	$\mathsf{Set}^{\mathsf{Bij}}$
scoped effects [ESOP'24, TOPLAS'25]	scopes (ordered, linear)	$\mathsf{Set}^{\lvert\mathbb{N}\rvert}$
POSIX-like fork [POPL'26]	thread IDs	$\mathsf{Set}^{\mathsf{FinRel}}$

A parameterized theory of threads — Next Saturday, ACV workshop

Parameters = thread IDs. (Not used linearly anymore.)

Operations:

fork$(a.x(a),\ y)$	x is the parent thread; might use a y is the child thread; cannot use a a is the ID of child; a is bound
wait$(a;\ x)$	wait for the thread named a to finish, continue as x
stop	end the current thread, unblock all threads waiting for it
act$_\sigma(x)$	perform observable action σ, continue as x

We axiomatize these operations with 9 equations.

A parameterized theory of threads — Example equations

The term $\mathsf{wait}(a;\, \mathsf{stop})$ acts as a unit for fork:

$$\mathsf{fork}(a.\mathsf{wait}(a; \mathsf{stop}), x) = x \qquad \mathsf{fork}(b.x(b), \mathsf{wait}(a; \mathsf{stop})) = x(a)$$

Operations wait and fork commute:

$$\mathsf{wait}(b;\, \mathsf{fork}(a.x(a),\, y)) = \mathsf{fork}(a.\mathsf{wait}(b;\, x(a)),\, \mathsf{wait}(b;\, y))$$

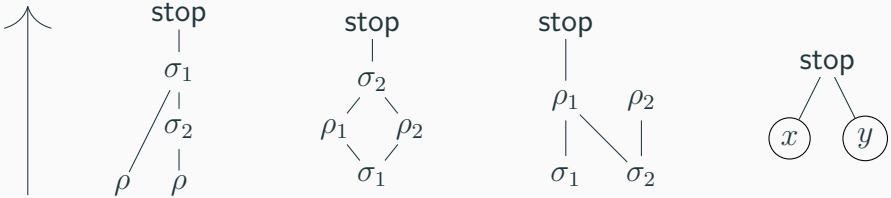
Performing an action can be delegated to a child thread:

$$\mathsf{act}_\sigma(x) = \mathsf{fork}(a.\mathsf{wait}(a, x),\, \mathsf{act}_\sigma(\mathsf{stop}))$$

A parameterized theory of threads

The elements of the monad obtained from the parameterized theory are **partial orders with labels** (pomsets).

Concurrent programs denote pomsets, for example:



We use the monad to give an adequate, fully abstract at first-order, semantics to a concurrent lambda calculus.

Summary

Scoped effects are operations that do not commute with monadic bind.

Our proposed definition

A **scoped effect** is a certain kind of parameterized algebraic theory.

We illustrate this definition with four examples:

- nondeterminism (backtracking) with once,
- exception catching,
- state with local values
- nondeterminism (backtracking) with cut.

Our definition provides **complete equational reasoning** for the examples.

Summary

Scoped effects are operations that do not commute with monadic bind.

Our proposed definition

A **scoped effect** is a certain kind of parameterized algebraic theory.

Theorems connecting to two other proposed definitions of scoped effects.

Future work:

- Axiomatizing more examples of scoped effects.
- Programming with scoped effects in direct style.
- Handlers for parameterized effects: ongoing work on an Idris library [Sigal et al HOPE'24]