Effects with Variable Binding

Jaro Reinders [1] Casper Bach [2] Benedikt Ahrens [1] Nicolas Wu [3]

[1]Delft University of Technology

[2]University of Southern Denmark

[3]Imperial College London

July 18, 2026

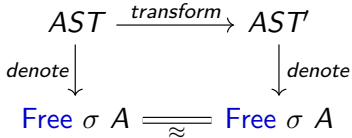
Overview

- Algebraic effects
- Semantics of computational effects
- Continuations are host-language functions
- Prevents analysis and transformation
- Solution: explicit variable binding
- To be used as compiler IR

	multi-pass	modular	verified
Nanopass	+	+	−
CompCert	+	−	+
ITrees	−	+	+
Ours	+	+	WIP

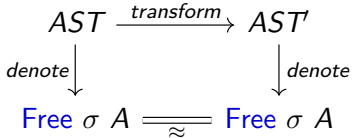
Verified program transformations (sneak peek)

ITrees:



Verified program transformations (sneak peek)

ITrees:



Ours:

Algebraic Effects Concretely

- 'Counter' effect with operations: $\mathit{increment} : 1 \to 1$, $\mathit{count} : 1 \to \mathbb{N}$
- Generates this free monad:

```
data Counter (A : Set) : Set where
  increment : Counter A → Counter A
  count : (Nat → Counter A) → Counter A
  return : A → Counter A
```

- count (λ c → return c)

Algebraic Effects Concretely

- 'Counter' effect with operations: $\mathit{increment} : 1 \to 1$, $\mathit{count} : 1 \to \mathbb{N}$
- Generates this free monad:

```
data Counter (A : Set) : Set where
  increment : Counter A → Counter A
  count : (Nat → Counter A) → Counter A
  return : A → Counter A
```

- count (λ c → return c)
- count (λ c → return 0) (⟹ return 0)
- Problem: How to tell the difference?

Contexts and indices

Contexts:

$$\begin{aligned}&\mathsf{Context} : \mathsf{Set}_1\\&\_,\_ : \mathsf{Context} \to \mathsf{Set} \to \mathsf{Context}\\&[] : \mathsf{Context}\end{aligned}$$

De Bruijn indices:

$$\begin{aligned}&\mathsf{Ix} : \mathsf{Set} \to \mathsf{Context} \to \mathsf{Set}\\&\#0 : \mathsf{Ix}\ A\ (\Gamma, A)\\&\#1 : \mathsf{Ix}\ B\ (\Gamma, B, A)\\&\ldots\end{aligned}$$

A category for explicit variable binding

$$\text{Set} \quad \Longrightarrow \quad \text{Context} \to \text{Set}$$

(Categorically a presheaf over renamings)

Solution: Effects with Variable Binding

```
data Counter' (A : Context → Set) (Γ : Context) : Set where
  increment : Counter' A Γ → Counter' A Γ
  count : Counter' A (Γ , Nat) → Counter' A Γ
  return : A Γ → Counter' A Γ
```

- `count (return #0) : Counter' (Ix Nat) Γ`

Solution: Effects with Variable Binding

```
data Counter' (A : Context → Set) (Γ : Context) : Set where
  increment : Counter' A Γ → Counter' A Γ
  count : Counter' A (Γ , Nat) → Counter' A Γ
  return : A Γ → Counter' A Γ
```

- count (return #0) : Counter' (Ix Nat) Γ
- But what if we want to return a constant?

Numeric effect operations

```
data M (A : Context → Set) (Γ : Context) : Set where
  increment :                                  M A Γ       → M A Γ
  count     :                                  M A (Γ , Nat) → M A Γ
  nat       : Nat                        →    M A (Γ , Nat) → M A Γ
  plus      : Ix Nat Γ → Ix Nat Γ →           M A (Γ , Nat) → M A Γ
  return    : A Γ                                          → M A Γ
```

- count (return #0) : M (Ix Nat) Γ
- count (nat 0 (return #0)) : M (Ix Nat) Γ

Further developments

In the paper:

- First-order signatures
- Monad structure
- Pretty printing
- Compiling conditionals
- Dead code elimination

Further developments

In the paper:

- First-order signatures
- Monad structure
- Pretty printing
- Compiling conditionals
- Dead code elimination

Not in the paper:

- Scoped operations, such as loops
- Equations and correctness proofs
- Formalization of monad structure
- Register allocation

Conclusion

- Algebraic Effects embeddings use functions as continuations
- Hinders analysis and transformation
- We propose to use explicit variable binding
- Moving from Set to Context $\to$ Set (presheaf over renamings)
- Many properties including the free monad construction are easily generalized
- Treat every language construct as an effect operation
- A basis for a verified compiler

Thank you!