Normalization by evaluation, algebraic theories, computational effects

Danel Ahman

Hughes Hall
University of Cambridge

An impure higher-order program

```
function foo f =
 let x = read() in
  let y = f (proj₀ (x , one)) in
   let z = return y in
    write zero ; return z
```

An impure higher-order program

reading from a memory cell

```
function foo f =
 let x = read() in
  let y = f (proj₀ (x , one)) in
   let z = return y in
    write zero ; return z
```

writing to a memory cell

An impure higher-order program

higher-order argument

reading from a memory cell

```
function foo f =
 let x = read() in
  let y = f (proj0 (x , one)) in
   let z = return y in
    write zero ; return z
```

writing to a memory cell

How to reason about these effects?

Computational effects

- Examples: global state, input/output, choice, ...

Computational effects

- Examples: global state, input/output, choice, ...

Plotkin, Power '02

- We model them using <u>algebraic theories</u> $T = (\Sigma, E)$

<u>operations</u> Σ + <u>equations</u> E

Computational effects

- Examples: global state, input/output, choice, ...

Plotkin, Power '02

- We model them using algebraic theories $T = (\Sigma , E)$

operations Σ	+	equations E
read x y		$\mathrm{write}_{\mathrm{zero}}\ (\mathrm{write}_{\mathrm{one}}\ x) \equiv \mathrm{write}_{\mathrm{one}}\ x$
$\mathrm{write}_{\mathrm{zero}}\ x$		$\mathrm{write}_{\mathrm{one}}\ (\mathrm{write}_{\mathrm{zero}}\ x) \equiv \mathrm{write}_{\mathrm{zero}}\ x$
$\mathrm{write}_{\mathrm{one}}\ x$		$\mathrm{write}_{\mathrm{zero}}\ (\mathrm{read}\ x\ y) \equiv \mathrm{write}_{\mathrm{zero}}\ x$
		...

How to reason about impure programs based on these algebraic effect theories?

A fine-grain call-by-value intermediate language

Levy, Power, Thielecke '03

- Type signature $\sigma ::= \alpha \,|\, 1 \,|\, \sigma \times \sigma \,|\, \sigma \rightharpoonup \sigma \,|\, \ldots$

- Value terms

$$\frac{}{\Gamma, x : \sigma, \Gamma' \vdash_v x : \sigma} \qquad \frac{\Gamma \vdash_v V_1 : \sigma_1 \quad \Gamma \vdash_v V_2 : \sigma_2}{\Gamma \vdash_v \langle V_1, V_2 \rangle : \sigma_1 \times \sigma_2} \qquad \frac{\Gamma \vdash_v V : \sigma_1 \times \sigma_2}{\Gamma \vdash_v \pi_i(V) : \sigma_i}$$

$$\frac{\Gamma, x : \sigma \vdash_p N : \tau}{\Gamma \vdash_v \lambda x : \sigma . N : \sigma \rightharpoonup \tau}$$

- Producer terms

$$\frac{\Gamma \vdash_p M : \sigma \quad \Gamma, x : \sigma \vdash_p N : \tau}{\Gamma \vdash_p M \,\mathbf{to}\, x.N : \tau} \qquad \frac{\Gamma \vdash_v V : \sigma}{\Gamma \vdash_p \mathbf{return}\, V : \sigma}$$

$$\frac{\Gamma \vdash_v V : \sigma \rightharpoonup \tau \quad \Gamma \vdash_v W : \sigma}{\Gamma \vdash_p VW : \tau}$$

Extending algebraic theories to the intermediate language

- Every operation in Σ defines a producer term

$$\boxed{\frac{\Gamma \vdash_p M_0 : \sigma \quad \Gamma \vdash_p M_1 : \sigma}{\Gamma \vdash_p \mathrm{read}_\sigma(M_0, M_1) : \sigma}} \quad \boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}(M) : \sigma}} \quad \boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(one)\sigma}(M) : \sigma}}$$

- Extend the usual beta-eta equations

$$\frac{\Gamma, x : \sigma \vdash_p M : \tau \quad \Gamma \vdash_v V : \sigma}{\Gamma \vdash_p (\lambda x : \sigma.M)V \equiv M[V/x] : \tau} \qquad \frac{\Gamma \vdash_v V \sigma \rightharpoonup \tau}{\Gamma \vdash_v V \equiv \lambda x : \sigma.(Vx) : \sigma \rightharpoonup \tau}$$

with all the equations in E

$$\boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}\,(\mathrm{write}_{(one)\sigma}\, M) \equiv \mathrm{write}_{(one)\sigma}\, M : \sigma}}$$

Extending algebraic theories to the intermediate language

- Every operation in Σ defines a producer term

$$\boxed{\frac{\Gamma \vdash_p M_0 : \sigma \quad \Gamma \vdash_p M_1 : \sigma}{\Gamma \vdash_p \mathrm{read}_\sigma(M_0, M_1) : \sigma}} \quad \boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}(M) : \sigma}} \quad \boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(one)\sigma}(M) : \sigma}}$$

- Extend the usual beta-eta equations

$$\frac{\Gamma, x : \sigma \vdash_p M : \tau \quad \Gamma \vdash_v V : \sigma}{\Gamma \vdash_p (\lambda x : \sigma.M)V \equiv M[V/x] : \tau} \qquad \frac{\Gamma \vdash_v V \sigma \rightharpoonup \tau}{\Gamma \vdash_v V \equiv \lambda x : \sigma.(V x) : \sigma \rightharpoonup \tau}$$

with all the equations in E

$$\boxed{\frac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}\,(\mathrm{write}_{(one)\sigma}\ M) \equiv \mathrm{write}_{(one)\sigma}\ M : \sigma}}$$

Extending algebraic theories to the intermediate language

- Every operation in Σ defines a producer term

$$\boxed{\dfrac{\Gamma \vdash_p M_0 : \sigma \quad \Gamma \vdash_p M_1 : \sigma}{\Gamma \vdash_p \mathrm{read}_\sigma(M_0, M_1) : \sigma}} \quad \boxed{\dfrac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}(M) : \sigma}} \quad \boxed{\dfrac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(one)\sigma}(M) : \sigma}}$$

- Extend the usual beta-eta equations

$$\dfrac{\Gamma, x : \sigma \vdash_p M : \tau \quad \Gamma \vdash_v V : \sigma}{\Gamma \vdash_p (\lambda x : \sigma.M)V \equiv M[V/x] : \tau} \qquad \dfrac{\Gamma \vdash_v V \sigma \rightharpoonup \tau}{\Gamma \vdash_v V \equiv \lambda x : \sigma.(V x) : \sigma \rightharpoonup \tau}$$

with all the equations in E

$$\boxed{\dfrac{\Gamma \vdash_p M : \sigma}{\Gamma \vdash_p \mathrm{write}_{(zero)\sigma}\,(\mathrm{write}_{(one)\sigma}\, M) \equiv \mathrm{write}_{(one)\sigma}\, M : \sigma}}$$

Is this representation of algebraic theories correct?

Theorem:

Given two terms in the algebraic effect theory,

they are provably equal in the algebraic theory

iff

they are provably equal in the extended language

Theorem:

Given two terms in the algebraic effect theory,

they are provably equal in the algebraic theory

✓ iff

they are provably equal in the extended language

Theorem:
Given two terms in the algebraic effect theory,

they are provably equal in the algebraic theory

✓ iff $\boxed{\text{Tricky!}}$

they are provably equal in the extended language

Provable equality

```
function foo f =
 let x = read() in
  let y = f (proj0 (x , one)) in
   let z = return y in
    write zero ; return z
```

is provably equal to

```
function foo f =
 let x = read() in
  let z = f x in
    write zero ; return z
```

How to decide provable equality?

Normalization

- So we want do <u>decide when terms are provably equal</u>
- We do this by computing their <u>normal forms</u>

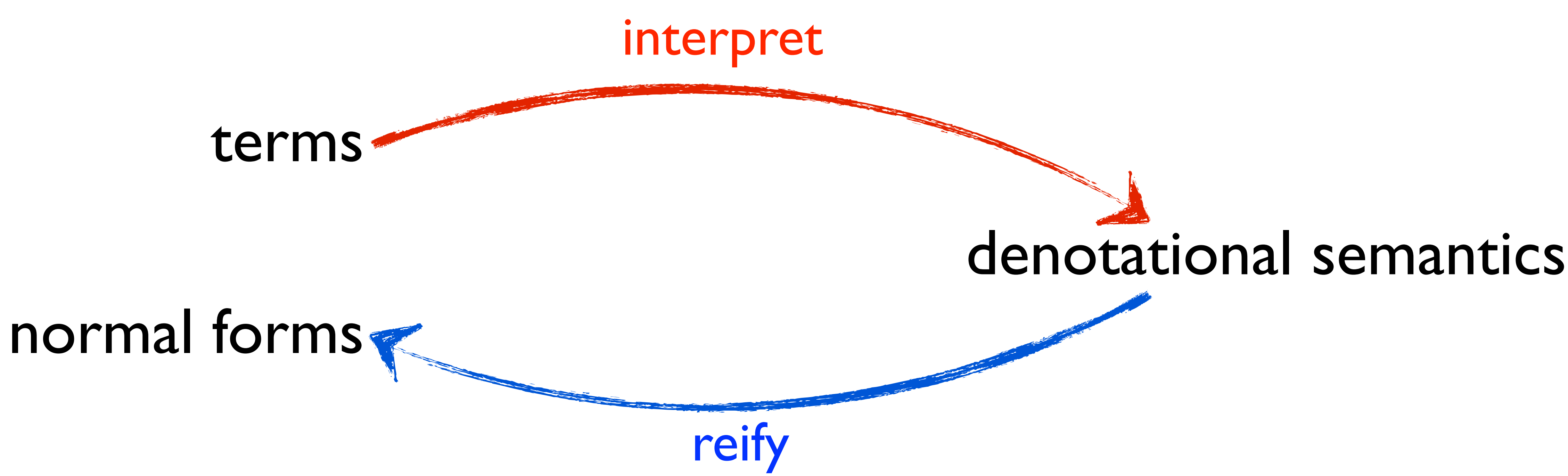
  satisfying:

Theorem:
Given two provably equal terms in the language,

they have canonical normal forms

Normalization by evaluation

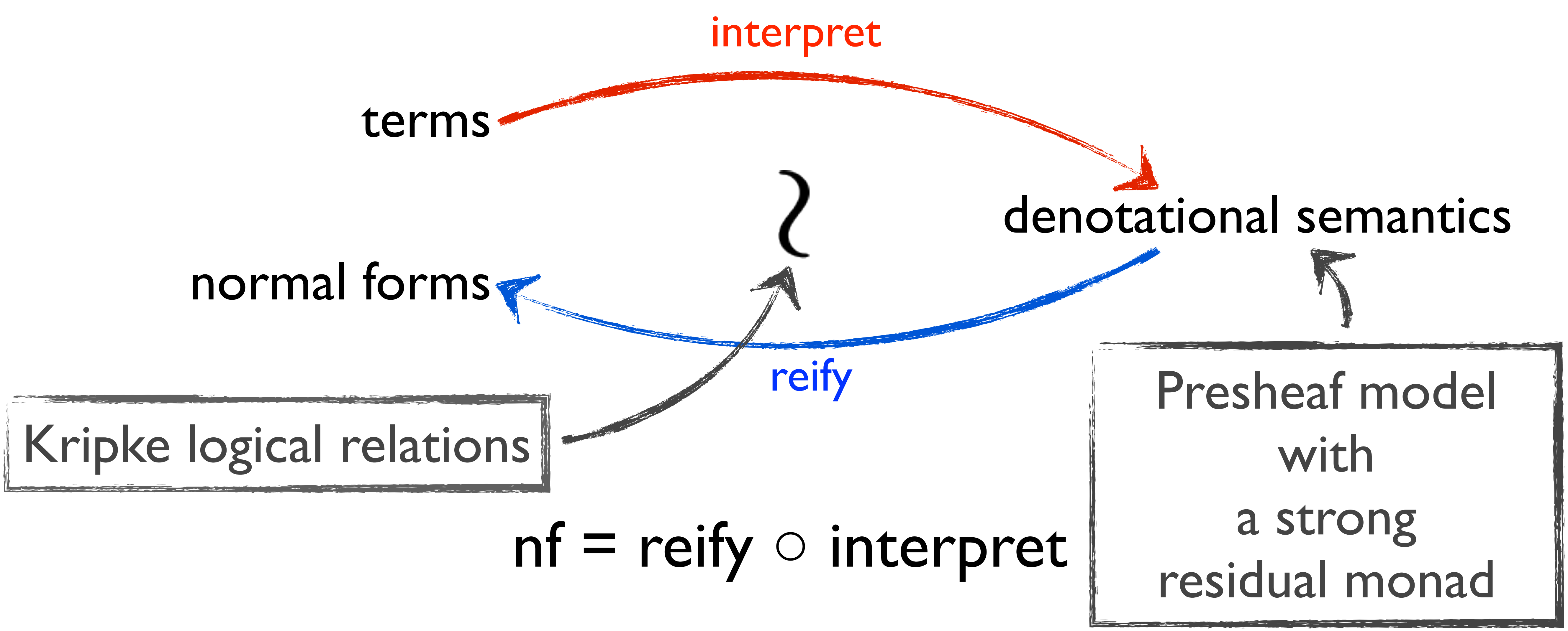
- A semantic notion of normalization
  - *Berger & Schwichtenberg '91, Filinski '01, Fiore et. al. '02, Abel et. al. '07*
- We define an inverse of interpretation called reification



nf = reify ○ interpret

Normalization by evaluation

- A semantic notion of normalization
  - *Berger & Schwichtenberg '91, Filinski '01, Fiore et. al. '02, Abel et. al. '07*
- We define an inverse of interpretation called reification



$$\mathrm{nf} = \mathrm{reify} \circ \mathrm{interpret}$$

Why a residualizing interpretation?

- We need to preserve the order of (possible) effects!

1. the reading effect

2. possible unknown effects

```
function foo f =
 let x = read() in
  let y = f (proj0 (x , one)) in
   let z = return y in
    write zero ; return z
```

3. the writing effect

Why a residualizing interpretation?

- We need to preserve the order of (possible) effects!

1. the reading effect

2. possible unknown effects

```
function foo f =
 let x = read() in
  let y = f (proj₀ (x , one)) in
   let z = return y in
    write zero ; return z
```

3. the writing effect

The main normalization results

Provably equal normal forms

$$\mathrm{nf} = \mathrm{reify} \circ \mathrm{interpret}$$

Theorem:
Given a term t in the language,

nf t is provably equal to t in the language

Canonical normal forms

nf = reify ○ interpret

Theorem:
Given two provably equal terms t and u in the language,

nf t and nf u are equivalent up to the algebraic theory

(equal if E is empty)

This representation is correct!

Theorem:
Given two terms in the algebraic effect theory,

they are provably equal in the algebraic theory

✓ iff ✓

they are provably equal in the extended language

Conclusions and future work

- We have justified the correctness of extending algebraic theories to a call-by-value intermediate language
- The normalization algorithm and proofs have been rigorously formalized in Agda
  - $\approx$ 6000 lines of formal proofs
- Future investigations
  - sum types and natural numbers
  - parametrized and second-order algebraic theories