Algebraic Theories and Control Effects, Back and Forth

Marcelo Fiore

Department of Computer Science and Technology
University of Cambridge

2014

an example of
the unreasonable effectiveness of mathematics
in computer science and logic

LOLA 2018
Oxford, UK
7.VII.2018

Universal Algebra and Computational Effects

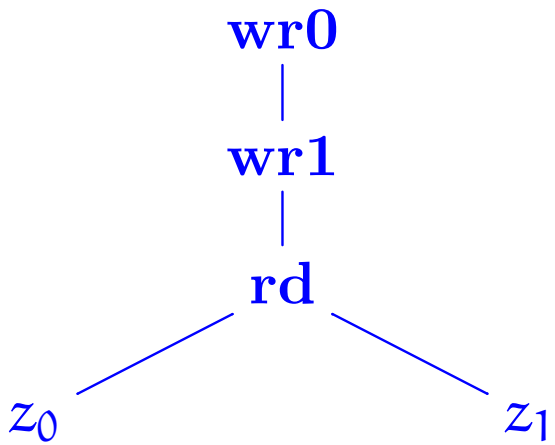
Universal Algebra

[Birkhoff (1935)]

- Signatures

  $\mathbf{wr0} : 1\ ,\ \mathbf{wr1} : 1\ ,\ \mathbf{rd} : 2$

- Free algebras



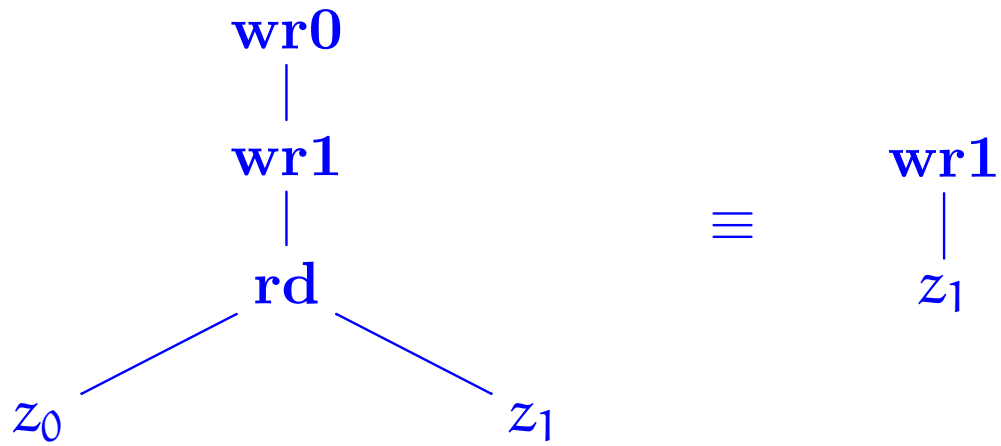
Universal Algebra

[Birkhoff (1935)]

- Signatures

$$\mathbf{wr0} : 1 \ , \ \mathbf{wr1} : 1 \ , \ \mathbf{rd} : 2$$

- Free algebras



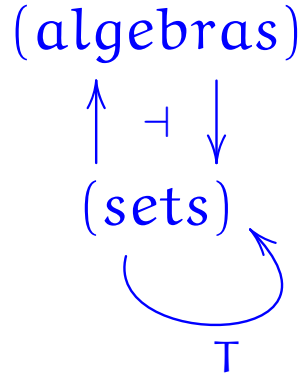
- Equational theories

$$\mathbf{wr0}\big(\mathbf{wr1}(z)\big) \equiv \mathbf{wr1}(z) \quad , \quad \mathbf{wr1}\big(\mathbf{wr0}(z)\big) \equiv \mathbf{wr0}(z)$$
$$\mathbf{wr0}\big(\mathbf{rd}(z_0, z_1)\big) \equiv \mathbf{wr0}(z_0) \quad , \quad \mathbf{wr1}\big(\mathbf{rd}(z_0, z_1)\big) \equiv \mathbf{wr1}(z_1)$$

Notions of Computation

[Moggi (1990); Plotkin, Power (2002)]

- Free-algebra monads



- Computational metalanguage

$$\frac{\Gamma \vdash_c t : \tau}{\Gamma \vdash_c \mathbf{wr0}_\tau(t) : \tau} \qquad \frac{\Gamma \vdash_c t : \tau}{\Gamma \vdash_c \mathbf{wr1}_\tau(t) : \tau}$$

$$\frac{\Gamma \vdash_c t_0 : \tau \qquad \Gamma \vdash_c t_1 : \tau}{\Gamma \vdash_c \mathbf{rd}_\tau(t_0, t_1) : \tau}$$

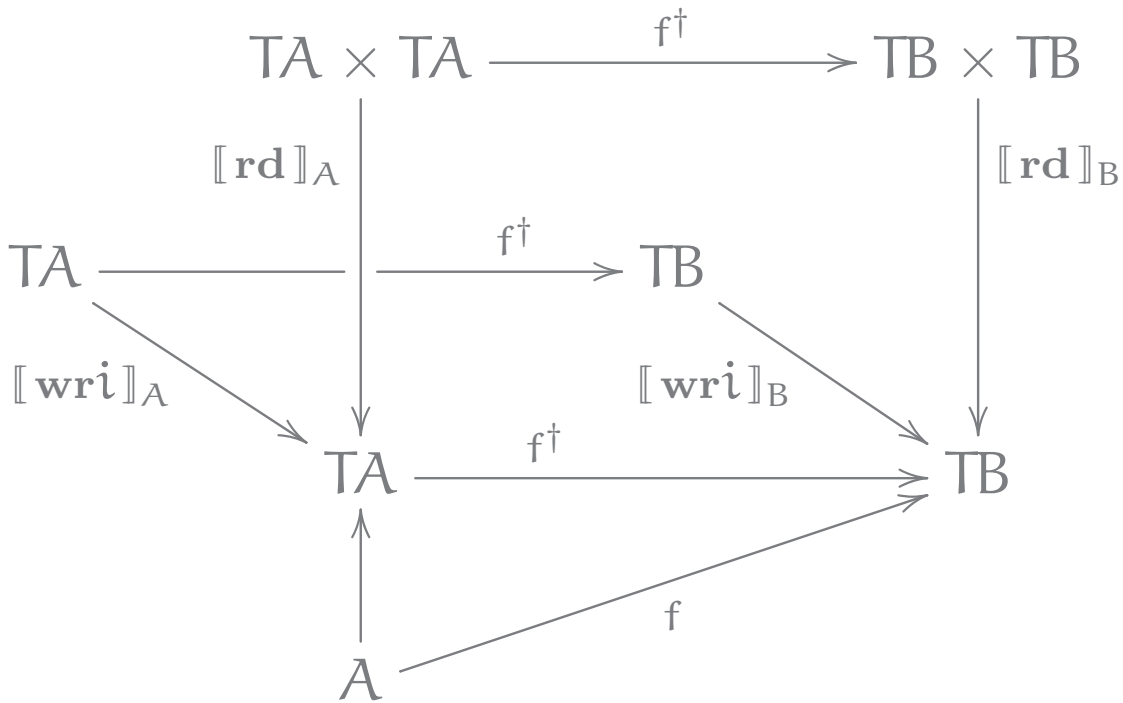
- Denotational semantics

$$[\![\, \Gamma \vdash_c t : \tau \,]\!] : [\![\, \Gamma \,]\!] \to T[\![\, \tau \,]\!]$$

- $[\![\, \mathbf{wr0}_\tau(t) \,]\!] = [\![\, \Gamma \,]\!] \xrightarrow{[\![t]\!]} T[\![\, \tau \,]\!] \xrightarrow{[\![\mathbf{wr0}]\!]} T[\![\, \tau \,]\!]$
- $[\![\, \mathbf{wr1}_\tau(t) \,]\!] = [\![\, \Gamma \,]\!] \xrightarrow{[\![t]\!]} T[\![\, \tau \,]\!] \xrightarrow{[\![\mathbf{wr1}]\!]} T[\![\, \tau \,]\!]$
- $[\![\, \mathbf{rd}_\tau(t_0, t_1) \,]\!] = [\![\, \Gamma \,]\!] \xrightarrow{\langle [\![t_0]\!], [\![t_1]\!] \rangle} T[\![\, \tau \,]\!] \times T[\![\, \tau \,]\!] \xrightarrow{[\![\mathbf{rd}]\!]} T[\![\, \tau \,]\!]$

- Equational theory
  - $\mathbf{let}\ x = \mathbf{wr0}_\sigma(s)\ \mathbf{in}\ t[x] \equiv \mathbf{wr0}_\tau\big(\mathbf{let}\ x = s\ \mathbf{in}\ t[x]\big)$
  - $\mathbf{let}\ x = \mathbf{wr1}_\sigma(s)\ \mathbf{in}\ t[x] \equiv \mathbf{wr1}_\tau\big(\mathbf{let}\ x = s\ \mathbf{in}\ t[x]\big)$
  - $\mathbf{let}\ x = \mathbf{rd}_\sigma(s_0, s_1)\ \mathbf{in}\ t[x]$
    $\equiv \mathbf{rd}_\tau\big(\mathbf{let}\ x = s_0\ \mathbf{in}\ t[x]\ ,\ \mathbf{let}\ x = s_1\ \mathbf{in}\ t[x]\big)$

- Equational theory
  - $\mathbf{let}\ x = \mathbf{wr0}_\sigma(s)\ \mathbf{in}\ t[x] \equiv \mathbf{wr0}_\tau\big(\mathbf{let}\ x = s\ \mathbf{in}\ t[x]\big)$
  - $\mathbf{let}\ x = \mathbf{wr1}_\sigma(s)\ \mathbf{in}\ t[x] \equiv \mathbf{wr1}_\tau\big(\mathbf{let}\ x = s\ \mathbf{in}\ t[x]\big)$
  - $\mathbf{let}\ x = \mathbf{rd}_\sigma(s_0, s_1)\ \mathbf{in}\ t[x]$
    $\equiv \mathbf{rd}_\tau\big(\mathbf{let}\ x = s_0\ \mathbf{in}\ t[x]\ ,\ \mathbf{let}\ x = s_1\ \mathbf{in}\ t[x]\big)$



Second-Order Algebraic Theories

Second-Order Algebraic Theories

- Binding signatures [Aczel (1978)]

  $\mathbf{app} : (0, 0)\ ,\ \mathbf{abs} : (1)$

Second-Order Algebraic Theories

- Binding signatures [Aczel (1978)]

  $\mathbf{app} : (0, 0)\ ,\ \mathbf{abs} : (1)$

- Free algebras [Fiore, Plotkin, Turi (1999); Hamana (2004); Fiore (2008)]

  $t \ ::= \ \underline{x} \mid M[t_1, \ldots, t_n]$ $\rightsquigarrow$ substitution structure

  $\quad \mid \ \mathbf{app}(t_1, t_2) \mid \mathbf{abs}(x.t')$

[Fiore, Hur (2010); Fiore, Mahmoud (2010)]

- Second-order equational theories

$$(\beta) \quad \mathbf{app}\big(\,\mathbf{abs}(x.\,\mathsf{M}[\underline{x}])\,,\,\mathsf{N}[]\,\big) \;\equiv\; \mathsf{M}\big[\mathsf{N}[]\big]$$

$$(\eta) \quad \mathbf{abs}\big(\,x.\,\mathbf{app}(\,\mathsf{F}[]\,,\,\underline{x}\,)\,\big) \;\equiv\; \mathsf{F}[]$$

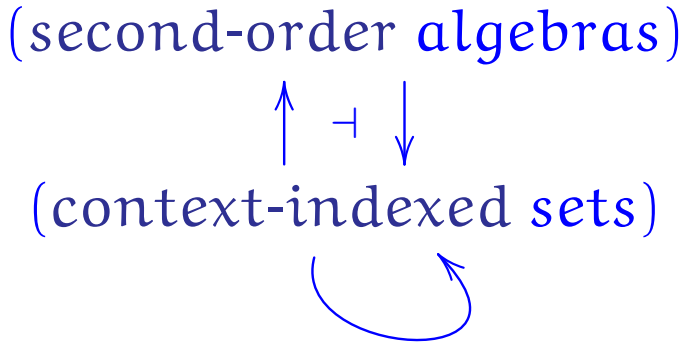
[Fiore, Hur (2010); Fiore, Mahmoud (2010)]

- Second-order equational theories

$$(\beta)\quad \mathbf{app}\big(\,\mathbf{abs}(x.\,\mathsf{M}[\underline{x}])\,,\,\mathsf{N}[]\,\big) \;\equiv\; \mathsf{M}\big[\mathsf{N}[]\big]$$

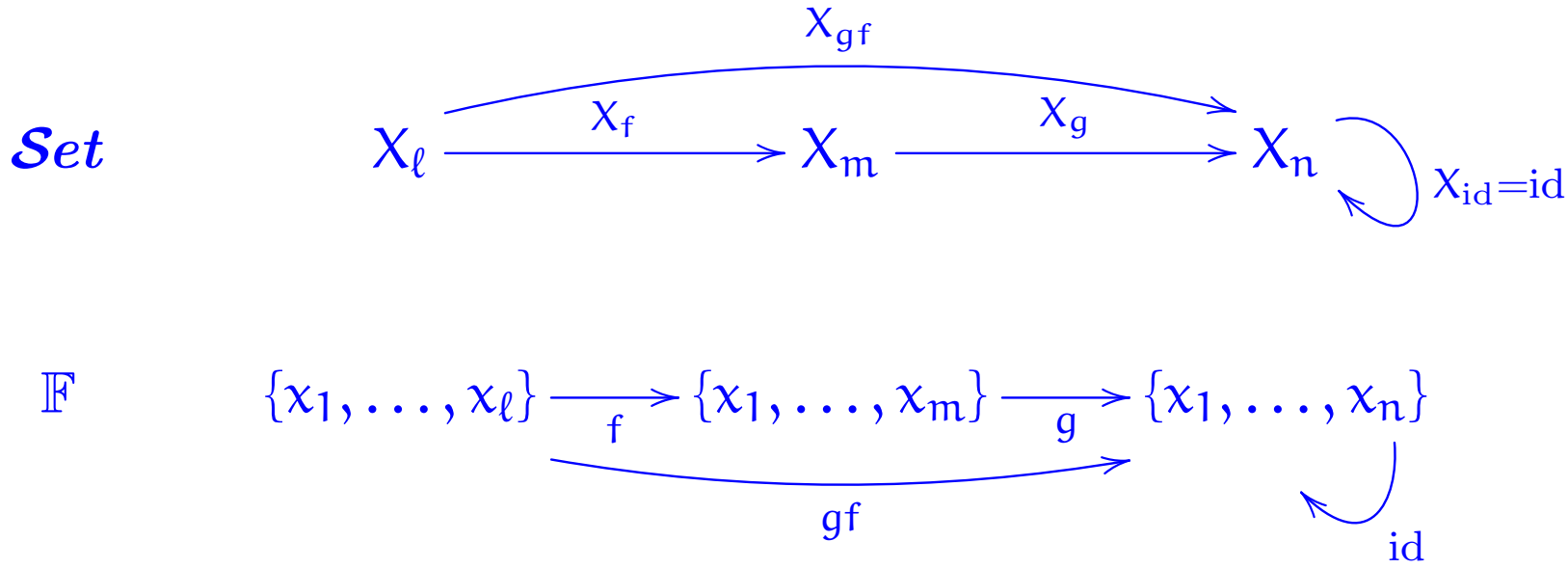
$$(\eta)\quad \mathbf{abs}\big(\,x.\,\mathbf{app}(\,\mathsf{F}[]\,,\,\underline{x}\,)\,\big) \;\equiv\; \mathsf{F}[]$$

- Free-algebra monads



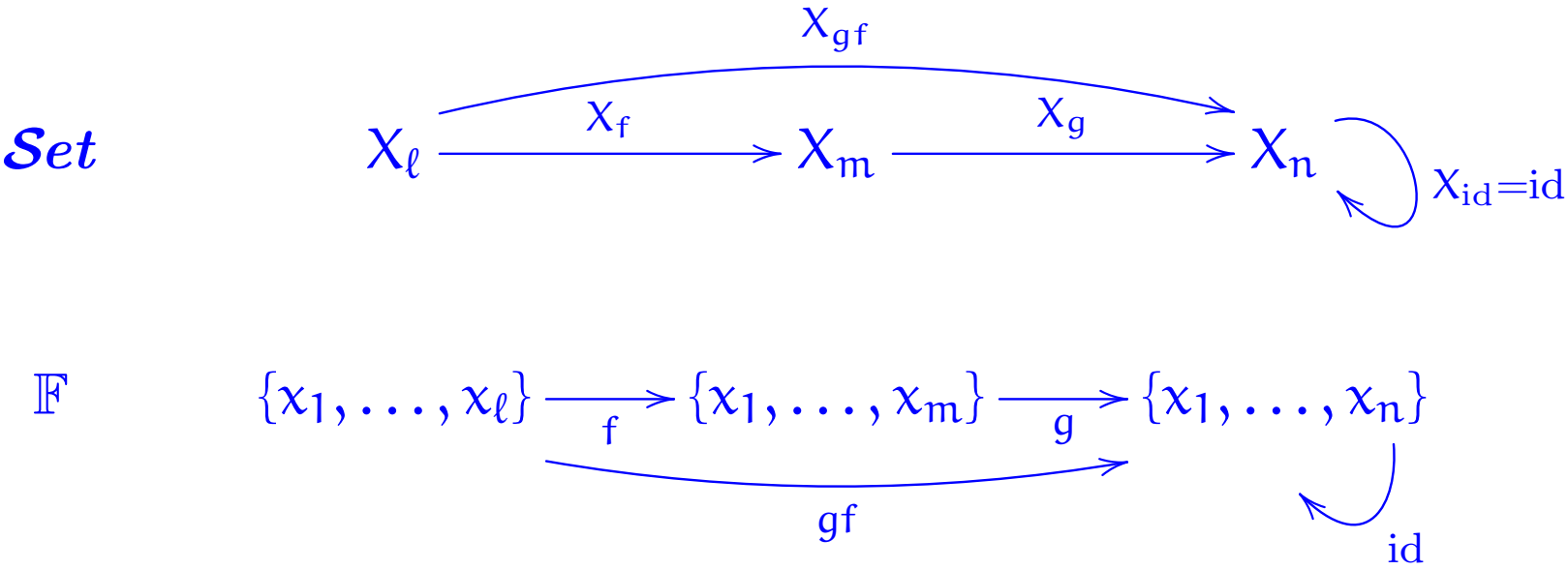
$\mathcal{S}et^{\mathbb{F}}$

- Context-indexed sets



$\mathcal{S}et^{\mathbb{F}}$

- Context-indexed sets

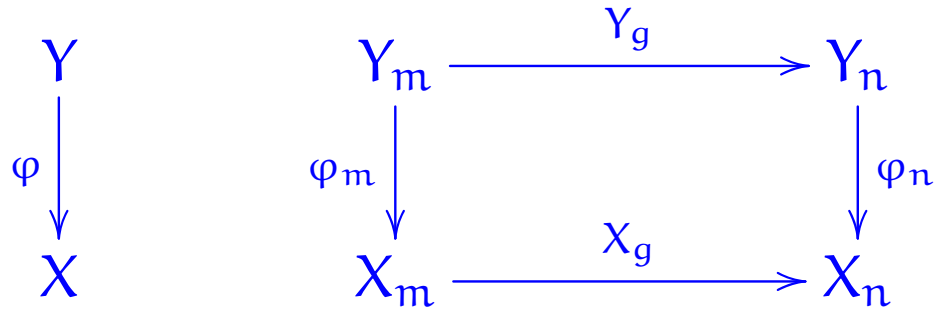


★ **Examples**

$\vec{S}$ $\quad S \rightrightarrows S \rightrightarrows \cdots \qquad S \qquad \cdots$

V $\quad \emptyset \rightarrow \{x_1\} \rightrightarrows \cdots \qquad \{x_1,\ldots,x_n\} \qquad \cdots$

- Renaming-invariant functions



$$\{x_1, \ldots, x_m\} \xrightarrow[g]{} \{x_1, \ldots, x_n\}$$

- Renaming-invariant functions

$$\begin{array}{ccc} Y & & Y_m \xrightarrow{Y_g} Y_n \\ \varphi \downarrow & & \varphi_m \downarrow \quad\quad \downarrow \varphi_n \\ X & & X_m \xrightarrow{X_g} X_n \end{array}$$

$$\{x_1, \ldots, x_m\} \xrightarrow[g]{} \{x_1, \ldots, x_n\}$$

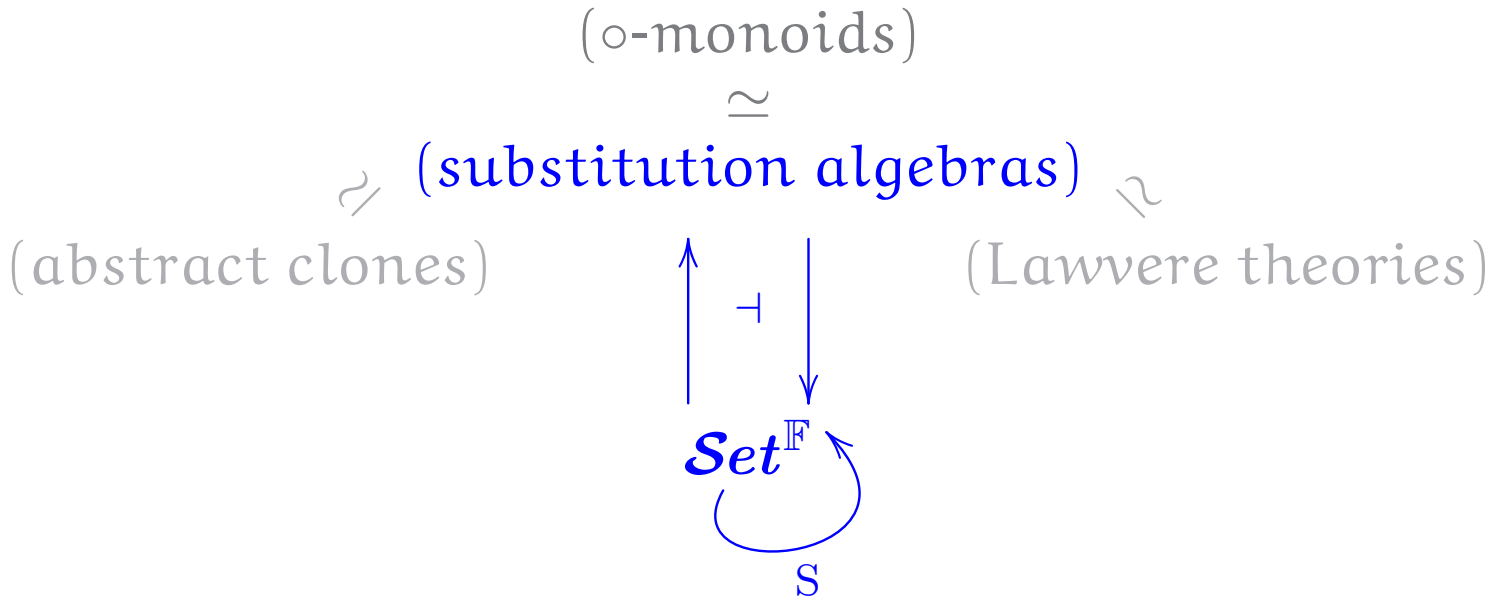
- **NB**

$$\begin{array}{ccccccc} 1 & \{*\} & = & \{*\} & = & \cdots & \{*\} & \cdots \\ \not\downarrow & & & & & & & \\ V & \emptyset & \longrightarrow & \{x_1\} & \rightrightarrows & \cdots & \{x_1, \ldots, x_n\} & \cdots \\ \not\downarrow & & & & & & & \\ 0 & \emptyset & = & \emptyset & = & \cdots & \emptyset & \cdots \end{array}$$

Substitution Algebras

Substitution Algebras [Fiore, Plotkin, Turi (1999)]

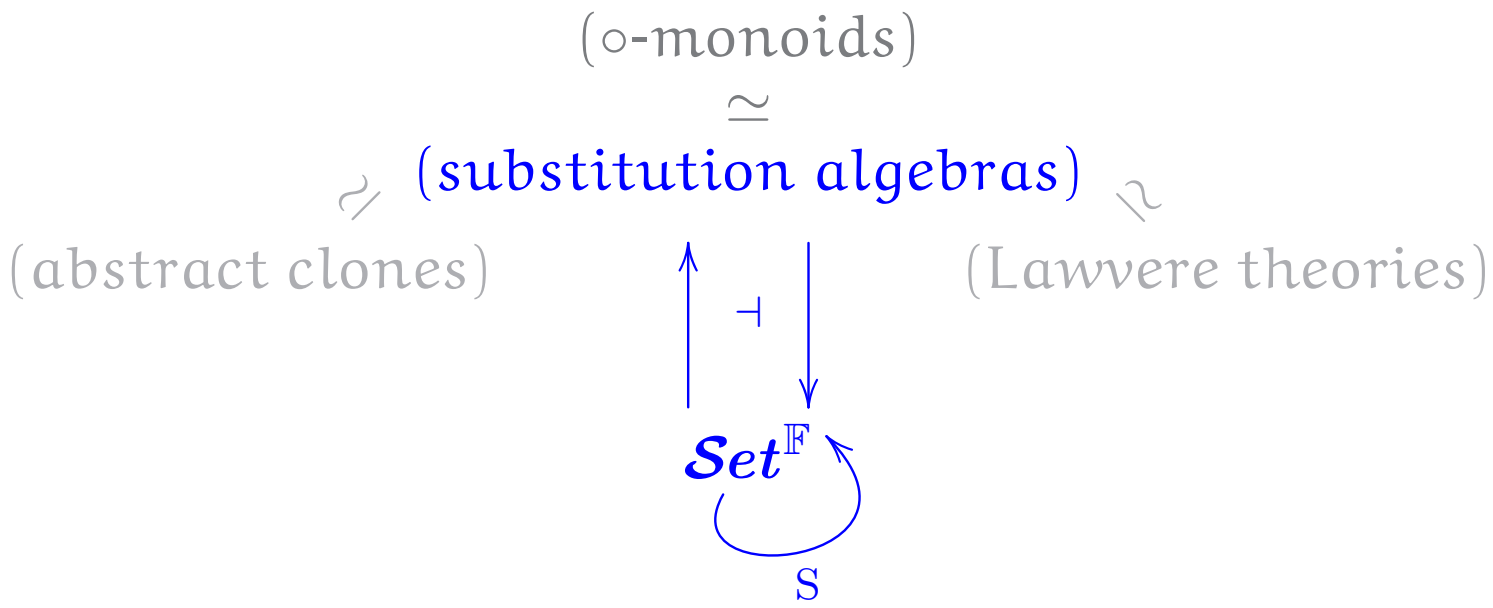
- Free substitution-algebra monad



$$SA = \mu X.\, V + A \circ X$$

Substitution Algebras [Fiore, Plotkin, Turi (1999)]

- Free substitution-algebra monad



$$SA = \mu X.\, V + A \circ X$$

★ **Definition**

A *substitution algebra* is a structure

$$V \xrightarrow{\textbf{var}} A \xleftarrow{\textbf{sub}} A^V \times A \quad \text{in } \mathcal{S}et^{\mathbb{F}}$$

subject to four axioms: substitution, weakening, extensionality, and associativity.

▸ Substitution Algebra Axioms

Computational Interpretation of Substitution Algebras

Computational Interpretation

[Fiore, Staton (2014)]

- Computational metalanguage

$$\frac{\Gamma \vdash_{\mathsf{v}} e : \mathbf{V}}{\Gamma \vdash_{\mathsf{c}} \mathbf{var}_{\tau}(e) : \tau} \qquad \frac{\Gamma, a : \mathbf{V} \vdash_{\mathsf{c}} t : \tau \quad \Gamma \vdash_{\mathsf{c}} u : \tau}{\Gamma \vdash_{\mathsf{c}} \mathbf{sub}(a.\, t, u) : \tau}$$

Computational Interpretation

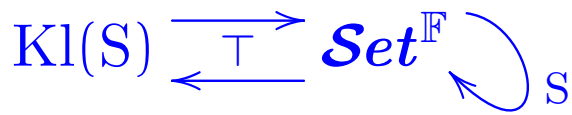
[Fiore, Staton (2014)]

- Computational metalanguage

$$\frac{\Gamma \vdash_{\mathsf{v}} e : \mathbf{V}}{\Gamma \vdash_{\mathsf{c}} \mathbf{var}_{\tau}(e) : \tau} \qquad \frac{\Gamma, \mathfrak{a} : \mathbf{V} \vdash_{\mathsf{c}} \mathfrak{t} : \tau \quad \Gamma \vdash_{\mathsf{c}} \mathfrak{u} : \tau}{\Gamma \vdash_{\mathsf{c}} \mathbf{sub}(\mathfrak{a}.\mathfrak{t}, \mathfrak{u}) : \tau}$$

- Denotational semantics

computations values



- Equational theory (I)
  - $\mathbf{let}\ x = \mathbf{sub}\big(\, a.\, \mathsf{M}[a]\,,\ \mathsf{N}[\,]\,\big)\ \mathbf{in}\ \mathsf{P}[x]$

    $\equiv\ \mathbf{sub}\big(\, a.\, \mathbf{let}\ x = \mathsf{M}[a]\ \mathbf{in}\ \mathsf{P}[x]\ \ ,\ \mathbf{let}\ x = \mathsf{N}[\,]\ \mathbf{in}\ \mathsf{P}[x]\ \,\big)$

- Equational theory (I)
  - $\mathbf{let}\ x = \mathbf{sub}\big(\, a.\, \mathsf{M}[a]\,,\ \mathsf{N}[\,]\,\big)\ \mathbf{in}\ \mathsf{P}[x]$

    $\equiv\ \mathbf{sub}\big(\, a.\, \mathbf{let}\ x = \mathsf{M}[a]\ \mathbf{in}\ \mathsf{P}[x]\ \ ,\ \mathbf{let}\ x = \mathsf{N}[\,]\ \mathbf{in}\ \mathsf{P}[x]\ \,\big)$
  - Weakening axiom

    $\mathbf{sub}\big(\, a.\, \mathsf{M}[\,]\,,\ \mathsf{N}[\,]\,\big) \equiv \mathsf{M}[\,]$

- Equational theory (I)
  - $\mathbf{let}\ \mathrm{x} = \mathbf{sub}\big(\ \mathrm{a}.\,\mathsf{M}[\mathrm{a}]\ ,\ \mathsf{N}[\,]\ \big)\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]$

    $\equiv\ \mathbf{sub}\big(\ \mathrm{a}.\,\mathbf{let}\ \mathrm{x} = \mathsf{M}[\mathrm{a}]\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]\ \ ,\ \mathbf{let}\ \mathrm{x} = \mathsf{N}[\,]\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]\ \ \big)$
  - Weakening axiom

    $\mathbf{sub}\big(\ \mathrm{a}.\,\mathsf{M}[\,]\ ,\ \mathsf{N}[\,]\ \big) \equiv \mathsf{M}[\,]$
  - Substitution axiom

    $\mathbf{sub}\big(\ \mathrm{a}.\,\mathbf{var}(\mathrm{a})\ ,\ \mathsf{N}[\,]\ \big) \equiv \mathsf{N}[\,]$

- Equational theory (I)
  - $\mathbf{let}\ x = \mathbf{sub}\big(\, a.\, \mathsf{M}[a]\,,\ \mathsf{N}[\,]\,\big)\ \mathbf{in}\ \mathsf{P}[x]$

    $\equiv\ \mathbf{sub}\big(\, a.\, \mathbf{let}\ x = \mathsf{M}[a]\ \mathbf{in}\ \mathsf{P}[x]\ \ ,\ \mathbf{let}\ x = \mathsf{N}[\,]\ \mathbf{in}\ \mathsf{P}[x]\ \,\big)$
  - Weakening axiom

    $\mathbf{sub}\big(\, a.\, \mathsf{M}[\,]\,,\ \mathsf{N}[\,]\,\big) \equiv \mathsf{M}[\,]$
  - Substitution axiom

    $\mathbf{sub}\big(\, a.\, \mathbf{var}(a)\,,\ \mathsf{N}[\,]\,\big) \equiv \mathsf{N}[\,]$
  - Abort law

    $\big(\,\mathbf{let}\ x = \mathbf{var}(a)\ \mathbf{in}\ \mathsf{P}[x]\,\big)\ \equiv\ \mathbf{var}(a)$

- Equational theory (I)
  - $\mathbf{let}\ \mathrm{x} = \mathbf{sub}\big(\,\mathrm{a}.\,\mathsf{M}[\mathrm{a}]\,,\,\mathsf{N}[\,]\,\big)\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]$
    $\equiv\ \mathbf{sub}\big(\,\mathrm{a}.\,\mathbf{let}\ \mathrm{x} = \mathsf{M}[\mathrm{a}]\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]\ \,,\,\mathbf{let}\ \mathrm{x} = \mathsf{N}[\,]\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]\ \,\big)$
  - Weakening axiom
    $\mathbf{sub}\big(\,\mathrm{a}.\,\mathsf{M}[\,]\,,\,\mathsf{N}[\,]\,\big) \equiv \mathsf{M}[\,]$
  - Substitution axiom
    $\mathbf{sub}\big(\,\mathrm{a}.\,\mathbf{var}(\mathrm{a})\,,\,\mathsf{N}[\,]\,\big) \equiv \mathsf{N}[\,]$
  - Abort law
    $\big(\,\mathbf{let}\ \mathrm{x} = \mathbf{var}(\mathrm{a})\ \mathbf{in}\ \mathsf{P}[\mathrm{x}]\,\big)\ \equiv\ \mathbf{var}(\mathrm{a})$
- Operational intuition

  In $\mathbf{sub}(\mathrm{a}.\mathrm{t}[\mathrm{a}], \mathrm{u})$ the *jump point* a is declared and the computation proceeds as in $\mathrm{t}[\mathrm{a}]$ leading to a value if this produces one, or *aborting and restarting* with the computation of u if $\mathbf{var}(\mathrm{a})$ is invoked.

- Equational theory (II)
  - Extensionality axiom

    $$\mathbf{sub}\big(\,\mathrm{a}.\,\mathsf{M}[\mathrm{a}]\,,\,\mathbf{var}(\mathrm{b})\,\big)\;\equiv\;\mathsf{M}[\mathrm{b}]$$

  - Associativity axiom

    $$\mathbf{sub}\big(\,\mathrm{a}.\,\mathbf{sub}\big(\,\mathrm{b}.\,\mathsf{L}[\mathrm{a},\mathrm{b}]\,,\,\mathsf{M}[\mathrm{a}]\,\big)\,,\,\mathsf{N}[\,]\,\big)$$
    $$\equiv\;\mathbf{sub}\big(\,\mathrm{b}.\,\mathbf{sub}(\,\mathrm{a}.\,\mathsf{L}[\mathrm{a},\mathrm{b}]\,,\,\mathsf{N}[\,]\,)\,,\,\mathbf{sub}(\,\mathrm{a}.\,\mathsf{M}[\mathrm{a}]\,,\,\mathsf{N}[\,]\,)\,\big)$$

★ Substitution, Jumps, and Algebraic Effects

[Fiore, Staton (2014)]

- Computational interpretation of substitution algebras as a code-jumping mechanism.
- Adequate denotational semantics, equational theory, and abstract machine.
- Representation of first-order virtual effects.
- Extension with effect handlers (by structural recursion).

CPS Interpretation of Substitution Algebras

Algebraic CPS Semantics

★ **Theorem** [Lawvere (1969), Kock (1970)]

In the context of strong monads,

$$\frac{\mathrm{T}\text{-algebra structures on objects } A}{\text{monad morphisms } \mathrm{T} \to \mathrm{K}_A}$$

for K_A the double-dualization or continuation monad relative to A.

Algebraic CPS Semantics

★ **Theorem** [Lawvere (1969), Kock (1970)]

In the context of strong monads,

$$\frac{\mathrm{T}\text{-algebra structures on objects } A}{\text{monad morphisms } \mathrm{T} \to \mathrm{K}_A}$$

for K_A the double-dualization or continuation monad relative to A.

- Algebraic CPS semantics:

$$\frac{\mathrm{V} \text{ is the initial substitution algebra}}{\text{monad morphism } \mathrm{S} \to \mathrm{K}_\mathrm{V}}$$

CPS Interpretation

$$\mathbf{var}_X \longmapsto \lambda a : V.\, \lambda k : V^X.\, a$$

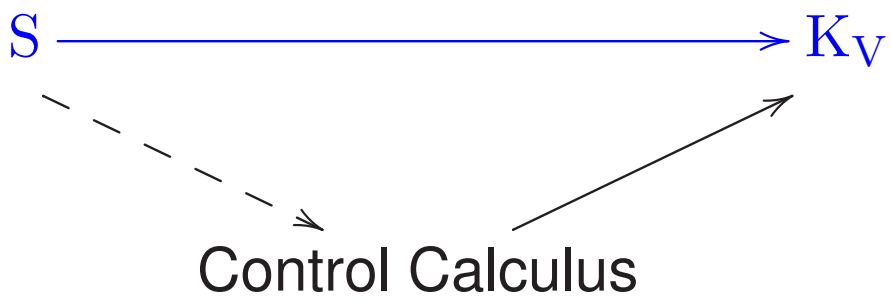
$$\mathbf{sub}_X \longmapsto \lambda\langle M : (K_V X)^V, N : K_V X\rangle.\, \lambda k : V^X.\, M\,(N\,k)\,k$$

$$S \longrightarrow K_V$$

CPS Interpretations

$$\mathbf{var}_X \longmapsto \lambda a : V.\, \lambda k : V^X.\, a$$

$$\mathbf{sub}_X \longmapsto \lambda \langle M : (K_V X)^V, N : K_V X \rangle.\, \lambda k : V^X.\, M\,(N\,k)\,k$$



CPS Interpretation in ML

```
signature SubstitutionAlgebra
= sig
    type V
    val var : V -> 'a
    val sub : ( V -> 'a ) * 'a -> 'a
end
```

CPS Interpretation in ML

```
signature SubstitutionAlgebra
= sig
    type V
    val var : V -> 'a
    val sub : ( V -> 'a ) * 'a -> 'a
end

structure sa :> SubstitutionAlgebra
= struct
    type R = unit cont
    type V = unit -> R
    fun var( a ) = throw (a()) ()
    fun sub( M , N )
      = callcc( fn k =>  M( fn() => throw k N ) )
  end
```

```
signature ParameterisedSubstitutionAlgebra
= sig
    type 'p V
    val var : 'p V * 'p -> 'a
    val sub : ( 'p V -> 'a ) * ('p -> 'a )  -> 'a
end

structure psa :> ParameterisedSubstitutionAlgebra
= struct
    type R = unit cont
    type 'p V = 'p -> R
    fun var( a , p ) = throw (a p) ()
    fun sub( M , N )
      = callcc( fn k =>  M( fn p => throw k (N p) ) )
  end
```

Inception Algebras

Inception Algebras

★ **Definition**

An (L, P)-*inception algebra* is a structure

$$L \times P \xrightarrow{\textbf{recall}} A \xleftarrow{\textbf{incept}} A^L \times A^P$$

Inception Algebras

★ **Definition**

An (L, P)-*inception algebra* is a structure

$$\mathsf{L} \times \mathsf{P} \xrightarrow{\textbf{recall}} A \xleftarrow{\textbf{incept}} A^{\mathsf{L}} \times A^{\mathsf{P}}$$

subject to:

- Substitution axiom

  $$\mathbf{inc}\big(\, \ell.\, \mathbf{rec}(\ell, \mathsf{P}[\,])\,,\; x.\, \mathsf{L}[x] \,\big) \;\equiv\; \mathsf{L}\big[\mathsf{P}[\,]\big]$$

- Extensionality axiom

  $$\mathbf{inc}\big(\, \ell.\, \mathsf{P}[\ell]\,,\; x.\, \mathbf{rec}(k, x) \,\big) \;\equiv\; \mathsf{P}[k]$$

- Weakening axiom

  $$\mathbf{inc}\big(\, \ell.\, \mathsf{M}[\,]\,,\; x.\, \mathsf{L}[x] \,\big) \;\equiv\; \mathsf{M}[\,]$$

- Associativity axiom

  $$\mathbf{inc}\big(\, \ell.\, \mathbf{inc}\big(\, k.\, \mathsf{P}[\ell, k]\,,\; x.\, \mathsf{K}[\ell, x] \,\big)\,,\; y.\, \mathsf{L}[y] \,\big)$$
  $$\equiv\; \mathbf{inc}\big(\, k.\, \mathbf{inc}\big(\, \ell.\, \mathsf{P}[\ell, k]\,,\; y.\, \mathsf{L}[y] \,\big)\,,\; x.\, \mathbf{inc}\big(\, \ell.\, \mathsf{K}[\ell, x]\,,\; y.\, \mathsf{L}[y] \,\big) \,\big)$$

★ Examples

- $(\mathrm{V}, 1)$-inception algebras
  = substitution algebras ($\simeq \circ$-monoids)

★ Examples

- $(\mathrm{V}, 1)$-inception algebras
  = substitution algebras ($\simeq \circ$-monoids)

- For a set D, $(\mathrm{V}, \vec{\mathrm{D}})$-inception algebras model a *code-jumping with data-passing* mechanism:

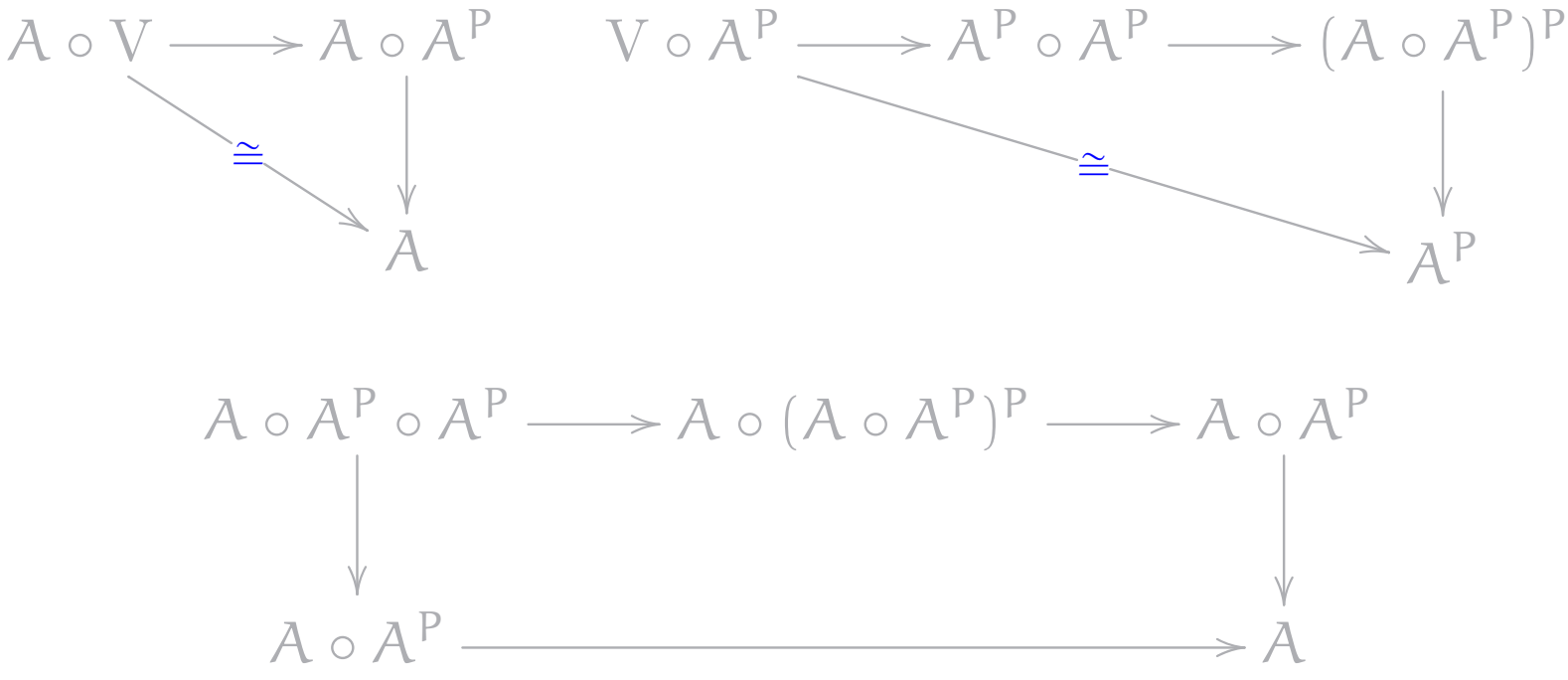
$$\frac{\Gamma \vdash_{\mathsf{v}} e : \mathbf{V} \qquad \Gamma \vdash_{\mathsf{v}} \mathrm{d} : \mathbf{D}}{\Gamma \vdash_{\mathsf{c}} \mathbf{rec}_{\tau}(e, \mathrm{d}) : \tau}$$

$$\frac{\Gamma, \ell : \mathbf{V} \vdash_{\mathsf{c}} \mathrm{t} : \tau \qquad \Gamma, \mathrm{x} : \mathbf{D} \vdash_{\mathsf{c}} \mathrm{u} : \tau}{\Gamma \vdash_{\mathsf{c}} \mathbf{inc}_{\tau}\big(\ell.\,\mathrm{t}\,, \mathrm{x}.\,\mathrm{u}\big) : \tau}$$

★ (V, P)-inception algebras

$\simeq$ unitary Menger algebras of rank P

$$V \to A^P \quad , \quad A \circ A^P \to A$$



★ For a set D, the free $(\mathrm{V}, \vec{\mathrm{D}})$-inception algebra on A is

$$\mu \mathrm{X}.\ (\mathrm{V} \times \vec{\mathrm{D}}) + A \circ \mathrm{X}^{\vec{\mathrm{D}}} \ .$$

($\rightsquigarrow$ effect handlers)

★ For a set D, the free $(\mathrm{V}, \vec{\mathrm{D}})$-inception algebra on A is

$$\mu X.\ (\mathrm{V} \times \vec{\mathrm{D}}) + A \circ X^{\vec{\mathrm{D}}} \ .$$

($\rightsquigarrow$ effect handlers)

- **NB**

  The analysis of free (V, P)-inception algebras suggests new mathematical models and programming primitives.

$$X \bullet Y \ = \ X \circ (\mathrm{V} \times Y)$$

$$\mathbf{inc}_{\mathrm{k}}\big(\ \ell.\,\mathbf{rec}(\ell, \mathsf{P}[\ell])\ ,\ x.\,\mathsf{L}[x]\ \big) \ \equiv \ \mathsf{L}\big[\mathsf{P}[\mathrm{k}]\big]$$

$$\mu X.\ (\mathrm{V} \times \mathrm{P}) + A \bullet X^{\mathrm{P}}$$

Inception Algebras in Logic

- $\mathrm{L} = \neg\,\mathrm{P}$

Negation Elimination

$$\frac{\neg\,\mathrm{P} \qquad \mathrm{P}}{A}$$

Excluded Middle

$$\frac{\begin{array}{cc}[\neg\,\mathrm{P}] & [\mathrm{P}] \\ \vdots & \vdots \\ A & A\end{array}}{A}$$

[de Groote (1995)]

NB Logical inceptions are not ML exceptions in scope.

▸ Counterexample

- Algebraic CPS semantics
  - $(\mathrm{R}^{\mathrm{P}}, \mathrm{P})$-inception algebra structure on R:

$$\begin{cases} \mathbf{rec} & \mapsto & \lambda\langle \ell : \mathrm{R}^{\mathrm{P}}, \mathrm{p} : \mathrm{P} \rangle . \, \ell \, \mathrm{p} \\ \mathbf{inc} & \mapsto & \lambda\langle \mathrm{M} : \mathrm{R}^{(\mathrm{R}^{\mathrm{P}})}, \mathrm{N} : \mathrm{R}^{\mathrm{P}} \rangle . \, \mathrm{M} \, \mathrm{N} \end{cases}$$

- Algebraic CPS semantics
  - (R^P, P)-inception algebra structure on R:

$$\begin{cases} \mathbf{rec} & \mapsto & \lambda\langle \ell : R^P, p : P \rangle.\, \ell\, p \\ \mathbf{inc} & \mapsto & \lambda\langle M : R^{(R^P)}, N : R^P \rangle.\, M\, N \end{cases}$$

  - Induced (R^P, P)-inception algebra structure on $K_R X$:

$$\begin{cases} \mathbf{rec}_X & \mapsto & \lambda\langle \ell : R^P, p : P \rangle.\, \lambda k : R^X.\, \ell\, p \\ \mathbf{inc}_X & \mapsto & \lambda\langle M : (K_R X)^{(R^P)}, N : (K_R X)^P \rangle.\, \lambda k : R^X. \\ & & \quad M\, (\lambda p : P.\, N\, p\, k)\, k \end{cases}$$

- Programming idiom

```
signature LogicalInceptionAlgebra
= sig
    type 'p linc
    val rec : 'p linc * 'p -> 'a
    val inc : ( 'p linc -> 'a ) * ( 'p -> 'a )  -> 'a
  end
```

‣ Logical Inception Algebra Structure

- Programming idiom

```
signature LogicalInceptionAlgebra
= sig
    type 'p linc
    val rec : 'p linc * 'p -> 'a
    val inc : ( 'p linc -> 'a ) * ( 'p -> 'a )  -> 'a
  end
```

- **NB** `inc` generalises and introduces a level of abstraction over `callcc`:

$$\texttt{callcc}(\texttt{f}) \;=\; \texttt{inc}(\texttt{f}, \texttt{id})$$

$$\texttt{inc}(\texttt{f}, \texttt{h}) \;=\; \texttt{callcc}(\texttt{f} \circ \neg\, \texttt{h})$$

- Programming idiom

```
signature LogicalInceptionAlgebra
= sig
    type 'p linc
    val rec : 'p linc * 'p -> 'a
    val inc : ( 'p linc -> 'a ) * ( 'p -> 'a )  -> 'a
  end
```

- **Applications**
  - Classical logic ▸ De Morgan
  - Encoding of local and global exception mechanisms
  - Safe exception handling in program modules
  - Coroutines

Algebraically,

$$\textbf{CPS} = \textbf{NE} + \textbf{EM}$$

Untyped Inception Algebras

- $(\mathrm{V}, \mathrm{V}^{\mathrm{n}})$-inception algebras

$$\mathrm{V} \times \mathrm{V}^{\mathrm{n}} \xrightarrow{\textbf{rec}} A \xleftarrow{\textbf{inc}} A^{\mathrm{V}} \times A^{\mathrm{V}^{\mathrm{n}}}$$

Untyped Inception Algebras

- $(\mathrm{V}, \mathrm{V}^{\mathrm{n}})$-inception algebras

$$\mathrm{V} \times \mathrm{V}^{\mathrm{n}} \xrightarrow{\mathbf{rec}} A \xleftarrow{\mathbf{inc}} A^{\mathrm{V}} \times A^{\mathrm{V}^{\mathrm{n}}}$$

  model the *untyped* CPS calculus[Appel (1992); Thielecke (1997)].

- Conversion table:

$\mathbf{rec}(\ell, \vec{x})$	$\ell\langle\vec{x}\rangle$
$\mathbf{inc}(\ell.\,\mathsf{P}[\ell]\,,\ \vec{x}.\,\mathsf{L}[\vec{x}]\,)$	$\mathsf{P}[\ell]\,\{\,\ell\langle\vec{x}\rangle = \mathsf{L}[\vec{x}]\,\}$

Sorted Inception Algebras

- Sorted sets [Milner (1991)]

$$|-| : S \rightharpoonup S^*$$

- Structures

$$V_\sigma \times \prod_i V_{|\sigma|_i} \qquad A^{V_\sigma} \times A^{\prod_i V_{|\sigma|_i}}$$

$$\searrow \qquad\qquad \swarrow$$

$$A$$

in $\mathcal{S}et^{\mathbb{F}[S]}$

subject to the inception algebra axioms. ▸ Inception Algebra Axioms

★ **Example**

For a set of sorts S, the sorted set

$$T_S \rightharpoonup T_S{}^*, \text{ where } T_S = \mu X.\, S + X^*$$

yields the typed CPS calculus, and hence sorted $\otimes\neg$-categories[Thielecke (1997)].

Back to Control Effects

Some Developments

- Untyped inception algebras
  - Recursion.
  - Algebraic interpretation of the lambda calculus in the untyped CPS calculus.
- Right Lambda Algebras
  - Computational interpretation as a mechanism for stack manipulation of code pointers. [Fiore, Staton (2014)]
  - Stack abstract machine for the untyped CPS calculus, and hence also for the lambda calculus.
- Left Lambda Algebras
  - Computational interpretation as a synchronous coroutine mechanism.
  - Producer-Consumer programming pattern.

Untyped Inception Algebras
Recursion
CPS Lambda Structure
Right Lambda Algebras
Stack Abstract Machine
Left Lambda Algebras
Coroutine Mechanism
Producer-Consumer

Final Remarks

Final Remarks

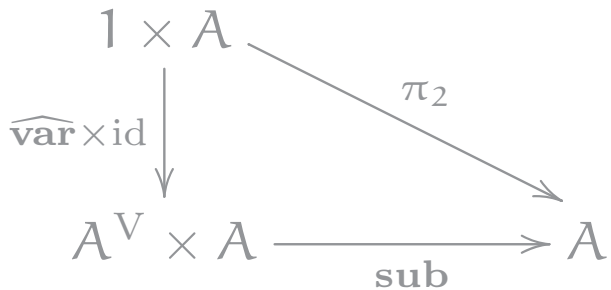
- Conclusions
  - The first algebraic axiomatisation of a variety of control effects.
  - Foundational analysis of the principles of programming with algebraic effects.
  - Inception algebras
- Directions
  - Semantic models
  - Classical/intermediate logics
  - Algebraic theories
  - Programming

Appendix

Substitution Algebra Axioms

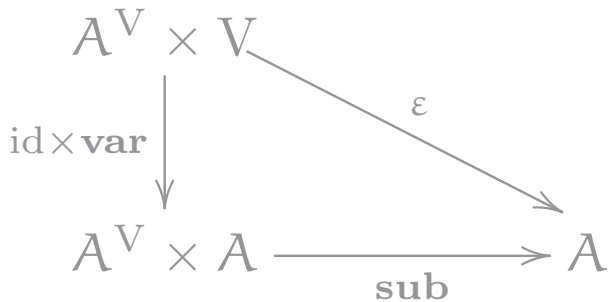
- Substitution

$$\mathbf{sub}\big(\, a.\,\mathbf{var}(a)\,,\, M[\,]\,\big) \;\equiv\; M[\,]$$



- Extensionality

$$\mathbf{sub}\big(\, a.\,M[a]\,,\, \mathbf{var}(b)\,\big) \;\equiv\; M[b]$$

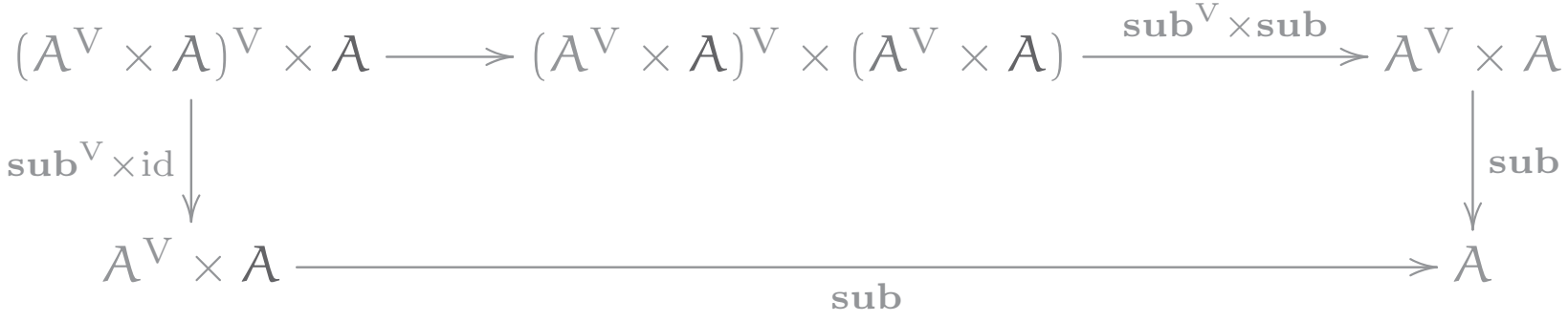


- Weakening

$$\mathbf{sub}\big(\,a.\,M[\,]\,,\,N[\,]\,\big)\;\equiv\;M[\,]$$

- Associativity

$$\mathbf{sub}\big(\,a.\,\mathbf{sub}\big(\,b.\,L[a,b]\,,\,M[a]\,\big)\,,\,N[\,]\,\big)$$
$$\equiv\;\mathbf{sub}\big(\,b.\,\mathbf{sub}(\,a.\,L[a,b]\,,\,N[\,]\,)\,,\,\mathbf{sub}(\,a.\,M[a]\,,\,N[\,]\,)\,\big)$$



Contraction Laws

- $\mathbf{sub}\big(\, a.\, M[a,b]\,,\, \mathbf{var}(b)\,\big) \;\equiv\; M[b,b]$

- $\mathbf{sub}\big(\, a.\, \mathbf{sub}(\, b.\, M[a,b]\,,\, N[\,]\,)\,,\, N[\,]\,\big) \;\equiv\; \mathbf{sub}\big(\, c.\, M[c,c]\,,\, N[\,]\,\big)$

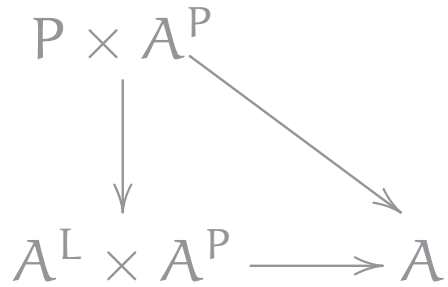
- $\mathbf{sub}\big(\, c.\, \mathbf{let}\; x = M[c]\; \mathbf{in}\; N[x,c]\;\;,\, L[\,]\,\big)$

  $\equiv\;\; \mathbf{sub}\big(\, a.\, \mathbf{let}\; x = M[a]\; \mathbf{in}\; \mathbf{sub}\big(b.\, N[x,b], L[\,]\big)\;\;,\, L[\,]\,\big)$

Inception Algebra Axioms

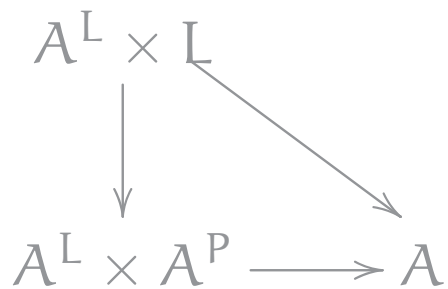
- Substitution

$$\mathbf{inc}\big(\,\ell.\,\mathbf{rec}(\ell, \mathsf{P}[\,])\,,\; x.\,\mathsf{L}[x]\,\big) \;\equiv\; \mathsf{L}[\mathsf{P}]$$



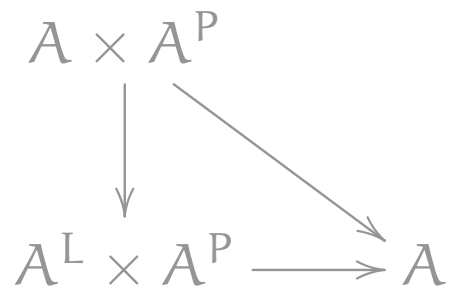
- Extensionality

$$\mathbf{inc}\big(\,\ell.\,\mathsf{P}[\ell]\,,\; x.\,\mathbf{rec}(k, x)\,\big) \;\equiv\; \mathsf{P}[k]$$



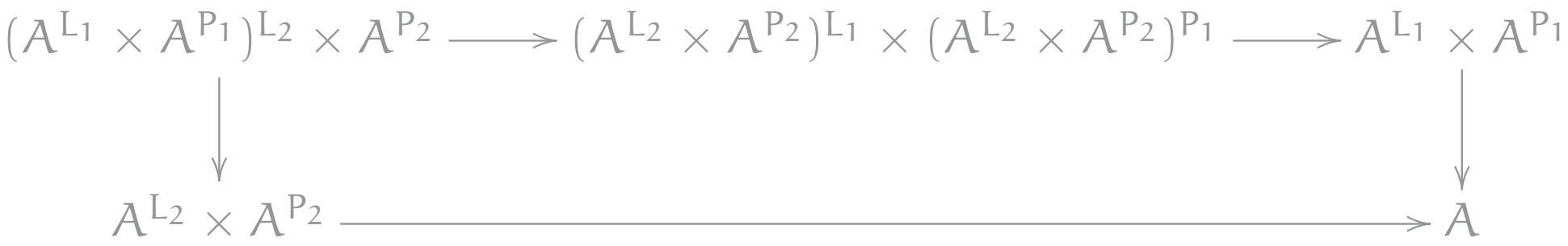
- Weakening

$$\mathbf{inc}\big(\,\ell.\,\mathsf{M}[\,]\,,\,x.\,\mathsf{L}[x]\,\big)\;\equiv\;\mathsf{M}[\,]$$



- Associativity

$$\mathbf{inc}\big(\,\ell.\,\mathbf{inc}\big(\,k.\,\mathsf{P}[\ell,k]\,,\,x.\,\mathsf{K}[\ell,x]\,\big)\,,\,y.\,\mathsf{L}[y]\,\big)$$
$$\equiv\;\mathbf{inc}\big(\,k.\,\mathbf{inc}\big(\,\ell.\,\mathsf{P}[\ell,k]\,,\,y.\,\mathsf{L}[y]\,\big)\,,\,x.\,\mathbf{inc}\big(\,\ell.\,\mathsf{K}[\ell,x]\,,\,y.\,\mathsf{L}[y]\,\big)\,\big)$$



▸ Sorted Inception Algebras

Inceptions Are Not Local Exceptions

- **NB**

$$\mathbf{sub}\big(\, e_1 . \mathbf{let}\ \ell = \mathbf{sub}(\, e_2 . \mathbf{ret}\, e_2 \,,\, \mathbf{ret}\, e_1 \,)\ \mathbf{in}\ \mathbf{var}\, \ell \ ,\, \langle\,\rangle \,\big) \ \equiv\ \langle\,\rangle$$

however

```
let exception e1
in ( let val l
          = let exception e2
            in e2
            handle e2 => e1 end
     in
       raise l
     end )
handle e1 => () end
```

outputs `uncaught exception e2`.

Logical Inception Algebra Structure

```
signature LogicalInceptionAlgebra
= sig
    type 'p linc
    val rec : 'p linc * 'p -> 'a
    val inc : ( 'p linc -> 'a ) * ( 'p -> 'a )  -> 'a
  end

structure lia :> LogicalInceptionAlgebra
= struct
    type R = unit cont
    type 'p linc = 'p -> R
    fun rec( a , p ) = throw (a p) ()
    fun inc( M , N )
      = callcc( fn k => M( fn p => throw k (N p) ) )
  end
```

De Morgan

```
val DeMorgan : ( 'a * 'b ) linc -> ('a linc,'b linc) sum
  = fn c =>
    inc( fn a => left a ,
         fn x => inc( fn b => right b ,
                      fn y => rec( c , (x,y) ) ) )
```

Nullary/Unary Untyped Inception Algebras

- Self recall

$$\mathbf{inc}\big(\,\ell.\,\mathbf{rec}(\ell,\ell)\,,\,x.\,\mathsf{M}[x]\,\big)\;\equiv\;\mathbf{inc}\big(\,\ell.\,\mathsf{M}[\ell]\,,\,x.\,\mathsf{M}[x]\,\big)$$

Nullary/Unary Untyped Inception Algebras

- Self recall

$$\mathbf{inc}\big(\,\ell.\,\mathbf{rec}(\ell,\ell)\,,\,x.\,M[x]\,\big)\;\equiv\;\mathbf{inc}\big(\,\ell.\,M[\ell]\,,\,x.\,M[x]\,\big)$$

- Recursion

  For

$$Y_t\;=\;\mathbf{inc}\big(\,\ell.\,\mathbf{rec}(\ell,\ell)\,,\,\ell.\,\mathbf{sub}\big(\,a.\,t[a]\,,\,\mathbf{rec}(\ell,\ell)\big)\,\big)$$

  we have

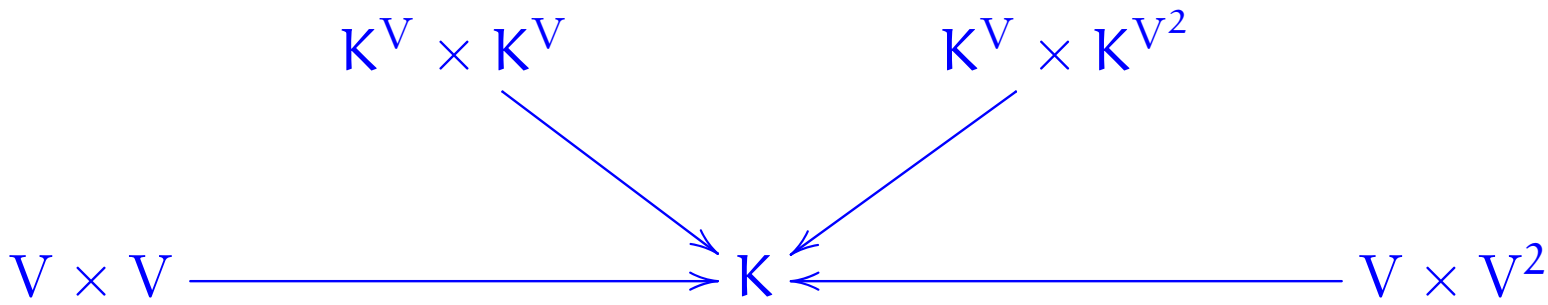
$$Y_t\;\equiv\;\mathbf{sub}\big(\,a.\,t[a]\,,\,Y_t\,\big)$$

▸ Developments

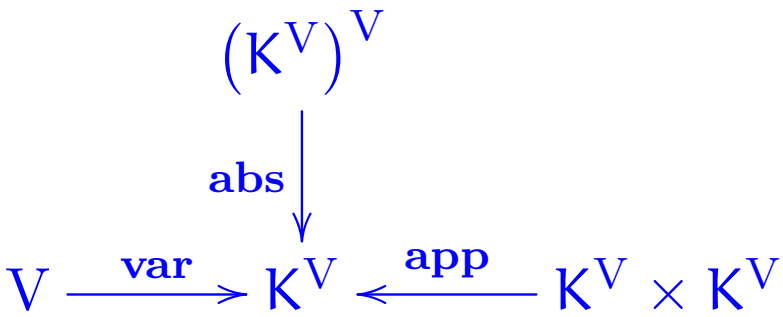
Unary/Binary Untyped Inception Algebras

- CPS lambda structure

The initial unary/binary untyped inception algebra



provides a CPS lambda structure



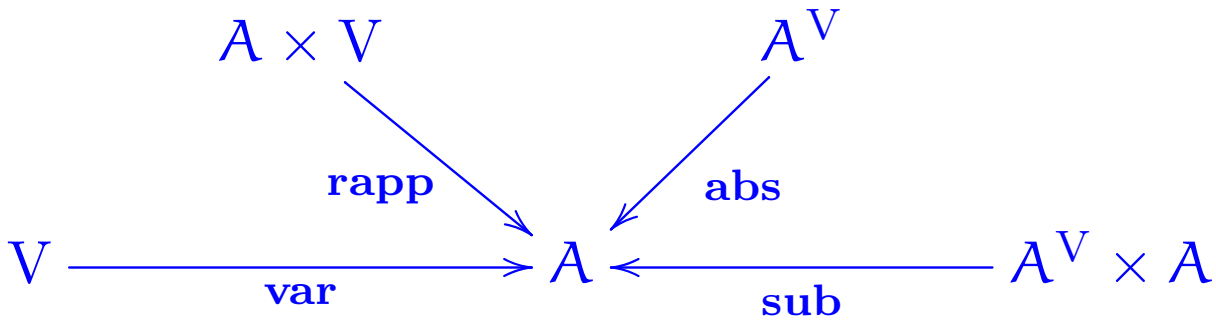
thereby inducing an homomorphic CPS interpretation

$$\Lambda \to \mathrm{K}^{\mathrm{V}}$$

of lambda terms.

- CPS interpretation
  - $\mathbf{var} : \mathrm{x}[e] \mapsto \mathbf{rec}(\, e\,,\, \langle \mathrm{x} \rangle\,)$
  - $\mathbf{abs} : \mathsf{M}[\mathrm{x}][e] \mapsto \mathbf{inc}(\;\; \mathrm{f}.\mathbf{rec}(\, e\,,\, \langle \mathrm{f} \rangle\,)\;\;,\;\; \langle \mathrm{a}, e \rangle.\, \mathsf{M}[\mathrm{a}][e]\;\;)$
  - $\mathbf{app} : \mathsf{M}[e], \mathsf{N}[e] \mapsto \mathbf{inc}(\;\; \mathrm{m}.\mathsf{M}[\mathrm{m}]\;\;,\;\; \langle \mathrm{f} \rangle.\, \mathbf{inc}(\;\; \mathrm{n}.\mathsf{N}[\mathrm{n}]\;\;,\;\; \langle \mathrm{a} \rangle.\, \mathbf{rec}(\mathrm{f}, \langle \mathrm{a}, e \rangle)\;\;)\;\;)$

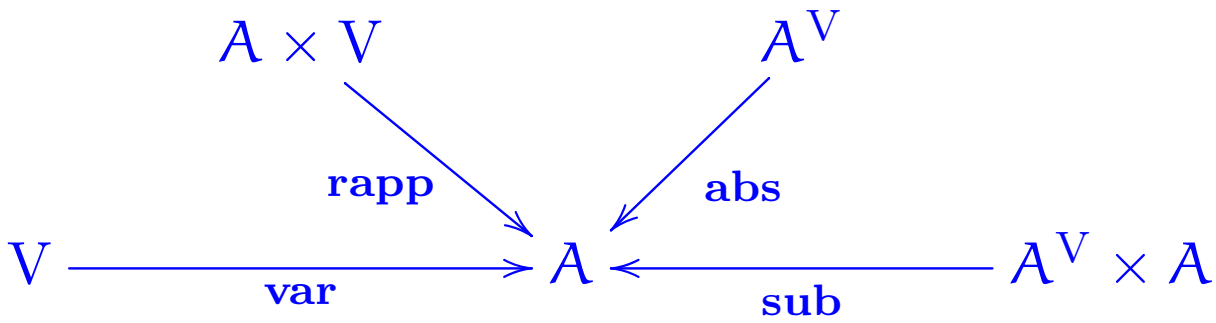
Right Lambda Algebras



$$(\beta_r) \quad \mathbf{rapp}\big(\,\mathbf{abs}(x.\,M[x])\,,\,a\,\big) \;\equiv\; M[a]$$

[Hirschowitz, Maggesi (2010); Hyland (2013)]

Right Lambda Algebras



push a

$$\underbrace{(\beta_r)}_{\text{summon}} \qquad \overbrace{\mathbf{rapp}\big(\underbrace{\mathbf{abs}(x.\,M[x])}_{\text{pop in } x},\, a\big)}^{\text{push } a} \;\equiv\; M[a]$$

- Computational interpretation: [Fiore, Staton (2014)]

  A mechanism for stack manipulation of code pointers.

- Application:

  Stack abstract machine for CPS calculus.

- Stack CPS structure

- $$\begin{array}{rcl} \mathrm{V} \times \mathrm{V}^{n} & \longrightarrow & A \\ a, \langle a_1, \ldots, a_n \rangle & \mapsto & \mathbf{push}(\cdots \mathbf{push}(\mathbf{var}(a), a_1) \cdots, a_n) \end{array}$$

- $$\begin{array}{rcl} A^{\mathrm{V}} \times A^{\mathrm{V}^{n}} & \longrightarrow & A \\ \mathsf{M}[a], \mathsf{N}[a_1, \ldots, a_n] & \mapsto & \mathbf{sub}(\ a.\,\mathsf{M}[a]\ , \\ & & \quad \mathbf{pop}(a_1.\cdots \mathbf{pop}(a_n.\mathsf{N}[a_1, \ldots, a_n]))\ ) \end{array}$$

▸ Developments

- Parameterised fixpoint
  - For

$$\mathsf{T}[\mathrm{f}, \mathrm{x}] = \mathbf{in}(\mathbf{yield}(\mathrm{x}, \mathbf{thunk}(z.\, \mathbf{push}(z, \mathbf{push}(\mathrm{f}, \mathbf{var}\ \mathrm{f})))))$$

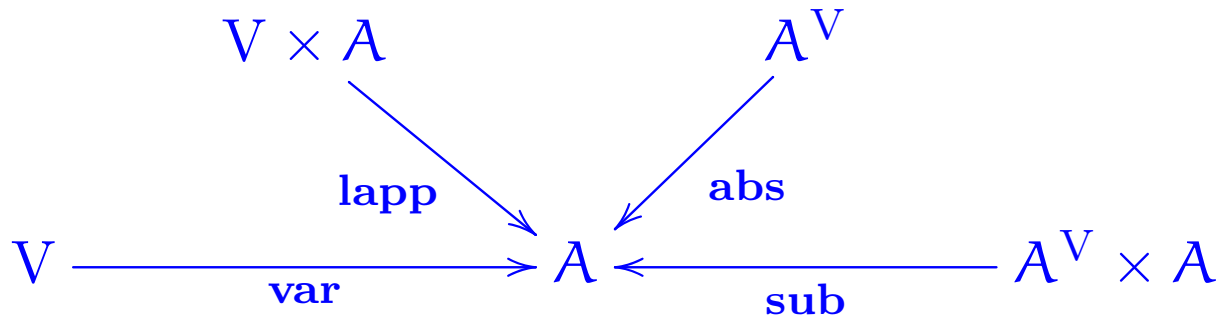
    and

$$\mathbf{In}[\mathrm{x}] = \mathbf{sub}\big(\, \mathrm{f}.\, \mathsf{T}[\mathrm{f}, \mathrm{x}]\,,\ \mathbf{pop}\big(\, \mathrm{f}.\, \mathbf{pop}(\mathrm{x}.\, \mathrm{t}[\mathrm{f}, \mathrm{x}])\,\big)\,\big)$$

    we have

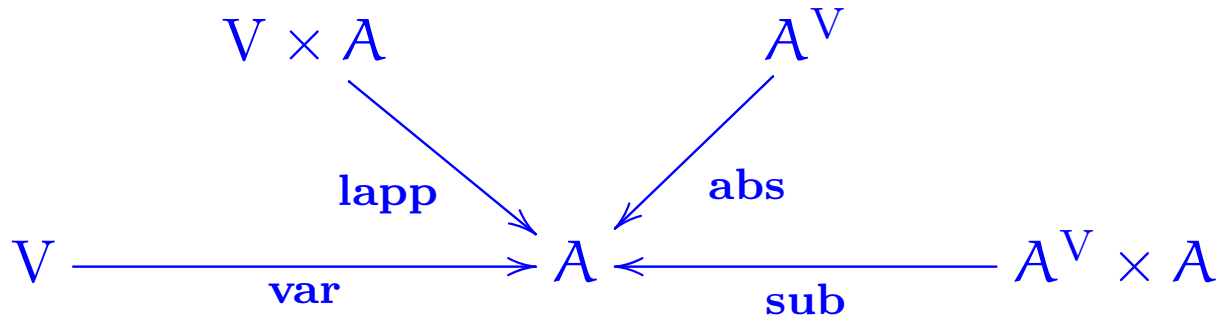
$$\mathbf{In}[\mathrm{x}] \ \equiv\ \mathbf{in}(\mathbf{yield}(\mathrm{x}, \mathbf{thunk}(z.\, \mathbf{In}[z])))$$

Left Lambda Algebras



$$(\beta_l) \quad \mathbf{sub}\big(\,\ell.\,\mathbf{lapp}(\ell, \mathsf{N}[\,])\,,\,\mathbf{abs}(\,\mathsf{k}.\,\mathsf{M}[\mathsf{k}]\,)\,\big) \;\equiv\; \mathbf{sub}\big(\,\mathsf{k}.\,\mathsf{M}[\mathsf{k}]\,,\,\mathsf{N}[\,]\,\big)$$

Left Lambda Algebras



$$\underbrace{(\beta_l)}_{\text{resume}} \quad \mathbf{sub}\big(\,\ell.\,\underbrace{\mathbf{lapp}(\ell, \mathsf{N}[\,])}_{\text{yield N}},\ \underbrace{\mathbf{abs}(\,\mathrm{k}.\,\mathsf{M}[\mathrm{k}]\,)}_{\text{thunk M}}\,\big) \equiv \mathbf{sub}\big(\,\mathrm{k}.\,\mathsf{M}[\mathrm{k}]\,,\,\mathsf{N}[\,]\,\big)$$

- Computational interpretation:

  A synchronous coroutine mechanism

- Application:

  Producer-Consumer

▸ Producer-Consumer

▸ Developments

- Axioms
  - $\mathbf{sub}\big(\,\ell.\,\mathbf{yield}(\ell, \mathsf{N}[\,])\,,\,\mathbf{thunk}(\,\mathsf{k}.\,\mathsf{M}[\mathsf{k}]\,)\,\big) \;\equiv\; \mathbf{sub}\big(\,\mathsf{k}.\,\mathsf{M}[\mathsf{k}]\,,\,\mathsf{N}[\,]\,\big)$
  - $\mathbf{sub}\big(\,\ell.\,\mathbf{thunk}\big(x.\,\mathsf{M}[\ell, x]\big)\,,\,\mathsf{N}[\,]\,\big)$
    $\equiv\;\; \mathbf{thunk}\big(\,x.\,\mathbf{sub}\big(\,\ell.\,\mathsf{M}[\ell, x]\,,\,\mathsf{N}[\,]\,\big)\,\big)$
  - $\mathbf{sub}\big(\,e.\,\mathbf{yield}\big(\mathsf{L}[e], \mathsf{M}[e]\big)\,,\,\mathsf{N}[\,]\,\big)$
    $\equiv\;\; \mathbf{sub}\big(\;\ell.\,\mathbf{yield}\big(\,\ell\,,\,\mathbf{sub}\big(e.\,\mathsf{M}[e], \mathsf{N}[\,]\big)\,\big)\;\;,$
    $\mathbf{sub}\big(\,e.\,\mathbf{var}(\mathsf{L}[e]), \mathsf{N}[\,]\big)\;\;\big)$

- Fixpoints
  - For a computation $\mathtt{t}(\cdot)$,

    $$\mathsf{Y}_\mathtt{t} = \mathbf{sub}\big(\,\mathtt{x}.\,\mathtt{t}\big(\mathbf{yield}(\mathtt{x}, \mathbf{var}\,\mathtt{x})\big)\,,\ \mathbf{thunk}\big(\,\mathtt{x}.\,\mathtt{t}\big(\mathbf{yield}(\mathtt{x}, \mathbf{var}\,\mathtt{x})\big)\,\big)\big)$$

    we have

    $$\mathsf{Y}_\mathtt{t} \equiv \mathtt{t}(\mathsf{Y}_\mathtt{t})$$

  - For a parameterised computation $\mathtt{t}[\mathtt{x}](\mathtt{c}.\mathsf{M}[\mathtt{c}])$, let

    $$\begin{aligned}\mathsf{T} = \ \ & \mathbf{thunk}(\mathtt{f}.\,\mathbf{thunk}(\mathtt{x}.\\ & \quad \mathtt{t}[\mathtt{x}]\big(\,\mathtt{c}.\,\mathbf{sub}(\mathtt{p}.\,\mathbf{yield}(\mathtt{p}, \mathbf{var}\ \mathtt{c}), \mathbf{yield}(\mathtt{f}, \mathbf{var}\ \mathtt{f}))\,\big)))\end{aligned}$$

    For

    $$\mathsf{Y}_\mathtt{t}[\mathtt{x}] = \mathbf{sub}\big(\,\mathtt{f}.\,\mathtt{t}[\mathtt{x}](\mathtt{c}.\,\mathbf{sub}(\mathtt{p}.\mathbf{yield}(\mathtt{p}, \mathbf{var}\ \mathtt{c}), \mathbf{yield}(\mathtt{f}, \mathsf{T}))\,,\ \mathsf{T}\,\big)$$

    we have

    $$\mathsf{Y}_\mathtt{t}[\mathtt{x}] \ \equiv \ \mathtt{t}[\mathtt{x}]\big(\,\mathtt{c}.\,\mathsf{Y}_\mathtt{t}[\mathtt{c}]\,\big)$$

- Producer-Consumer: one in, two out

  Let

$$\begin{aligned}\mathbf{In}[a] &= \mathbf{in}(\mathbf{yield}(a, \mathbf{thunk}(x.\mathbf{In}[x]))) \\ \mathbf{Out} &= \mathbf{thunk}(y.\mathbf{out}(\mathbf{out}(\mathbf{yield}(y, \mathbf{Out}))))\end{aligned}$$

  Then

$$\mathbf{sub}(\, a.\, \mathbf{In}[a]\,,\, \mathbf{Out}\,) \equiv \mathbf{in}(\mathbf{out}(\mathbf{out}(\mathbf{sub}(x.\mathbf{In}[x], \mathbf{Out}))))$$

▸ Developments