Algebraic Theories and Control Effects, Back and Forth

Marcelo Fiore

COMPUTER LABORATORY
UNIVERSITY OF CAMBRIDGE

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

ALCOP
16.V.2014

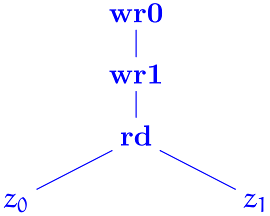
Universal Algebra

[Birkhoff (1935)]

- Signatures

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

- Free algebras



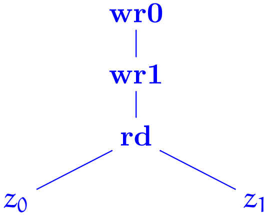
Universal Algebra

[Birkhoff (1935)]

- Signatures

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

- Free algebras



- Equational theories

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

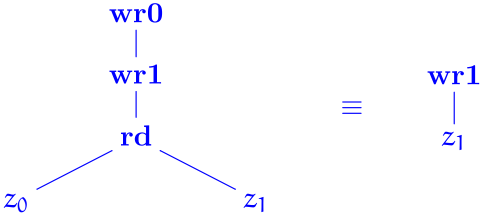
Universal Algebra

[Birkhoff (1935)]

- Signatures

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

- Free algebras



- Equational theories

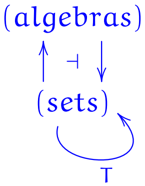
$$\mathbf{wr0}\big(\mathbf{wr1}(z)\big) \equiv \mathbf{wr1}(z) \quad , \quad \mathbf{wr1}\big(\mathbf{wr0}(z)\big) \equiv \mathbf{wr0}(z)$$

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

Notions of Computation

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

- Free-algebra monads



- Computational metalanguage

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

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

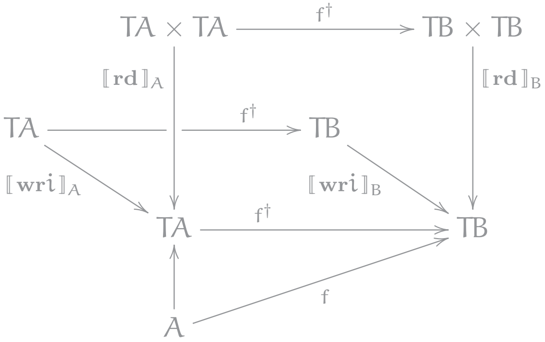
- Denotational semantics

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

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

- 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

- 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)]

  $t \quad ::= \quad \mathbf{var}(x) \ \mid\ \mathsf{M}[t_1, \ldots, t_n] \ \mid\ \mathbf{app}(t_1, t_2) \ \mid\ \mathbf{abs}(x.t')$

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

- Second-order equational theories

$$\mathbf{app}\big(\mathbf{abs}(x.\,\mathsf{M}[\,\mathbf{var}(x)\,]),\,\mathsf{N}[\,]\,\big) \equiv \mathsf{M}\big[\mathsf{N}[\,]\big]$$

$$\mathbf{abs}\big(\,x.\,\mathbf{app}(\,\mathsf{F}[\,],\,\mathbf{var}(x)\,)\,\big) \equiv \mathsf{F}[\,]$$

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

- Second-order equational theories

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

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

- Free-algebra monads

$$\begin{array}{c} (\text{second-order algebras}) \\ \uparrow \;\dashv\; \downarrow \\ (\text{context-indexed sets}) \end{array}$$

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

- Context-indexed sets

$$X_m$$

$$\{x_1, \ldots, x_m\}$$

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

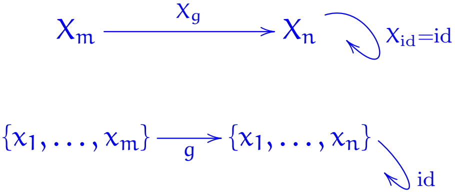
- Context-indexed sets

$$X_m \xrightarrow{X_g} X_n$$

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

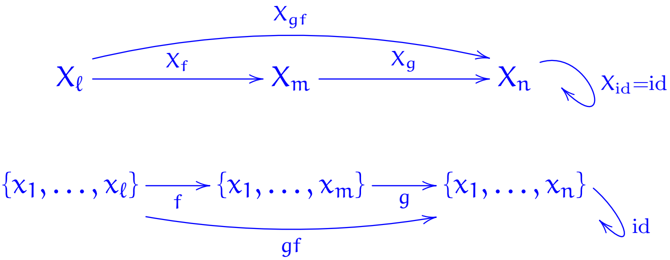
$\mathcal{S}et^{\mathbb{F}}$

- Context-indexed sets



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

- Context-indexed sets



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

- Renaming-invariant functions

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

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

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

- Renaming-invariant functions

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

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

★ **Examples**

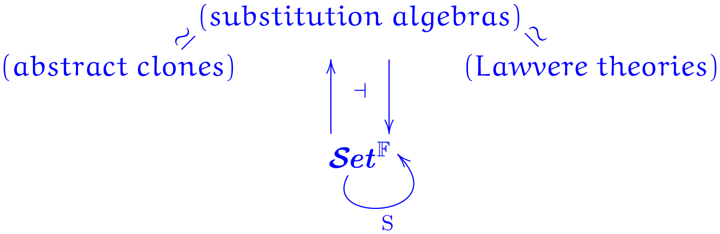
$$1 \qquad \{*\} = \{*\} = \cdots \qquad \{*\} \qquad \cdots$$

$$V \qquad \{\} \longrightarrow \{x_1\} \rightrightarrows \cdots \qquad \{x_1, \ldots, x_n\} \qquad \cdots$$

Substitution Algebras

[Fiore, Plotkin, Turi (1999)]

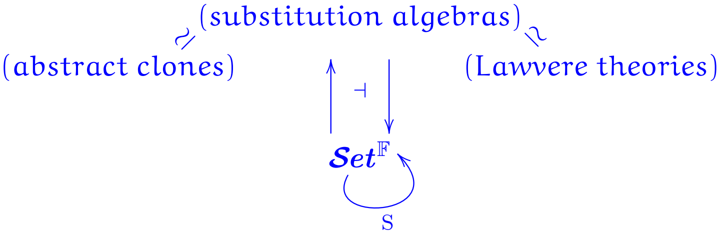
- Free substitution-algebra monad



Substitution Algebras

[Fiore, Plotkin, Turi (1999)]

- Free substitution-algebra monad



★ **Definition**

A *substitution algebra* is a structure

$$\mathrm{V} \xrightarrow{\ \mathbf{var}\ } A \xleftarrow{\ \mathbf{sub}\ } A^{\mathrm{V}} \times A$$

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

▸ Substitution Algebra Axioms

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

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

- Denotational semantics

computations values

$$\mathrm{Kl}(S) \underset{\longleftarrow}{\overset{\longrightarrow}{\top}} \mathcal{S}\boldsymbol{et}^{\mathbb{F}}$$

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

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

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

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

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

- 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

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

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

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

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

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

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

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

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

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

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

    $\big(\textbf{let } x = \textbf{var}(a) \textbf{ in } \mathsf{P}[x]\big)\ \equiv\ \textbf{var}(a)$
- Operational intuition

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

- Equational theory (II)
  - Extensionality axiom

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

  - Associativity axiom

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

★ 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 incorporating effect handlers.

CPS Semantics

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

In the context of strong monads, T-algebra structures on objects A are in bijective correspondence with monad morphisms $\mathrm{T} \to \mathrm{K}_A$ for K_A the double-dualization or continuation monad relative to A.

- CPS translation: $\mathrm{S} \to \mathrm{K}_\mathrm{V}$

CPS Semantics

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

In the context of strong monads, T-algebra structures on objects A are in bijective correspondence with monad morphisms $T \to K_A$ for K_A the double-dualization or continuation monad relative to A.

- CPS translation: $S \to K_V$
  - $\mathbf{var}_X \mapsto \lambda a : V.\, \lambda k : V^X.\, a$
  - $\mathbf{sub}_X \mapsto \lambda\langle M : (K_V X)^V, N : K_V X\rangle.\, \lambda k : V^X.\, M\,(N\,k)\,k$

Direct-Style CPS Semantics

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

Direct-Style CPS Semantics

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

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

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

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

subject to:

- Substitution axiom

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

- 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

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

★ Examples

- Substitution algebras = $(\mathrm{V}, 1)$-inception algebras

★ Examples

- Substitution algebras = $(V, 1)$-inception algebras

- More generally, for a set D, (V, D)-inception algebras model a *code-jumping with data-passing* mechanism:

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

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

Inception Algebras in Logic

- $L = \neg P$

¬ elimination

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

excluded middle

$$\frac{\begin{matrix}[\neg P] & & [P] \\ \vdots & & \vdots \\ A & & A\end{matrix}}{A}$$

[de Groote (1995)]

- Inceptions are not local exceptions.

Counterexample

- CPS semantics

Logical Inception Algebras CPS Semantics

- Programming idiom

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

▸ Logical Inception Algebra Structure

- Programming idiom

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

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

$$\mathtt{callcc}(\mathrm{f}) = \mathtt{inc}\big(\mathrm{f}, \mathrm{id}\big)$$

$$\mathtt{inc}\big(\mathrm{f}, \mathrm{h}\big) = \mathtt{callcc}\big(\mathrm{f} \circ \neg\, \mathrm{h}\big)$$

- Programming idiom

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

★ **Example**

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

- Programming idiom

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

- **Applications**
  - Encoding of local and global exception mechanisms
  - Safe exception handling in program modules
  - Coroutines

Untyped Inception Algebras

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

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

Untyped Inception Algebras

- (V, V^n)-inception algebras

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

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

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

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

- Recursion

  For

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

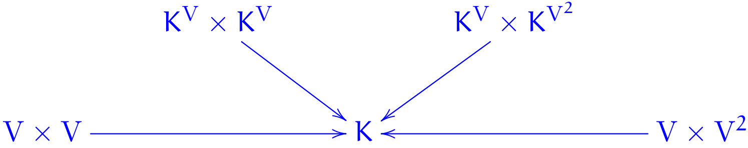
  we have

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

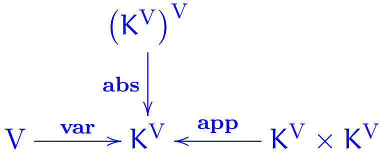
Unary/Binary Untyped Inception Algebras

- CPS lambda structure

  The initial unary/binary untyped inception algebra



provides a CPS lambda structure



thereby inducing an homomorphic CPS interpretation

$$\Lambda \to K^V$$

of lambda terms.

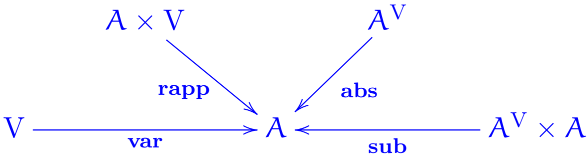
- CPS interpretation

  - $\mathbf{var} : x[e] \mapsto \mathbf{rec}(\, e\, ,\ \langle x \rangle\,)$

  - $$\mathbf{abs} : \mathsf{M}[x][e] \mapsto \mathbf{inc}(\ \ f.\, \mathbf{rec}(\, e\, ,\ \langle f \rangle\,)\ \ ,\ \ \langle a, e \rangle.\, \mathsf{M}[a][e]\ \ )$$

  - $$\mathbf{app} : \mathsf{M}[e], \mathsf{N}[e] \mapsto \mathbf{inc}(\ \ m.\mathsf{M}[m]\ \ ,\ \ \langle f \rangle.\, \mathbf{inc}(\ \ n.\mathsf{N}[n]\ \ ,\ \ \langle a \rangle.\, \mathbf{rec}(f, \langle a, e \rangle)\ \ )\ \ )$$

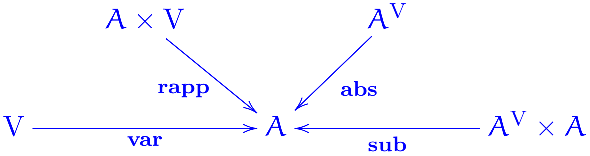
Right Lambda Algebras



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

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

Right Lambda Algebras



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

- Computational interpretation:

  [Fiore, Staton (2014)]

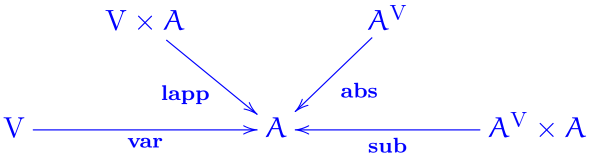
  A mechanism for stack manipulation of code pointers.

- Application:

  Stack abstract machine for CPS calculus.

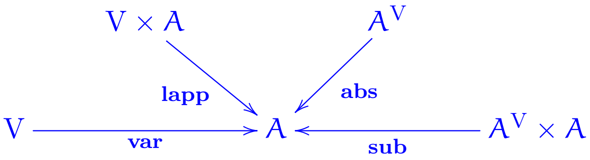
▸ CPS Structure

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



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

- Computational interpretation:

  A synchronous coroutine mechanism

- Application:

  Producer-Consumer

▸ Producer-Consumer

Final Remarks

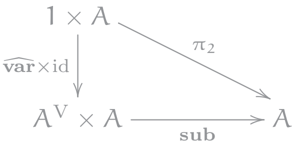
- Conclusions
  - The first algebraic axiomatisation of control effects.
  - Foundational analysis of the principles of programming with algebraic effects.
- Directions
  - Models
  - Classical/intermediate logics
  - Algebraic theories

Appendix

Substitution Algebra Axioms

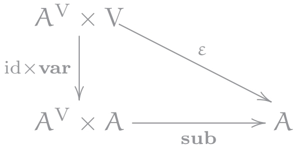
- Substitution

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



- Extensionality

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

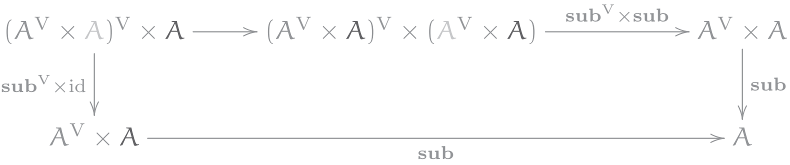


- Weakening

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

- Associativity

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



▸ Substitution Algebras

Contraction Laws

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

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

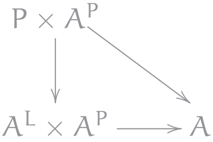
- $\mathbf{sub}\big(\, c.\, \mathbf{let}\ x = \mathsf{M}[c]\ \mathbf{in}\ \mathsf{N}[x, c]\ \ ,\ \mathsf{L}[\,]\,\big)$

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

Inception Algebra Axioms

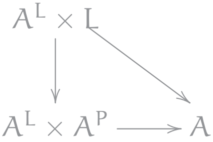
- Substitution

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



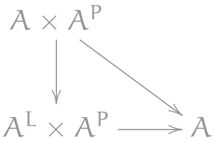
- Extensionality

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



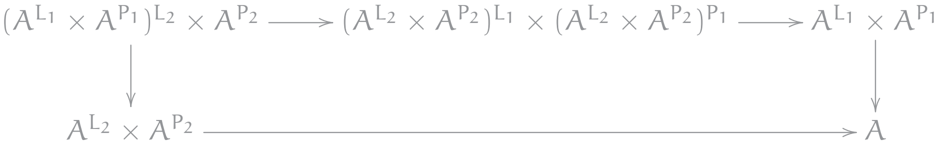
- Weakening

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



- Associativity

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



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 Algebras CPS Semantics

- (R^P, P)-inception algebra structure on R
  - $\mathbf{rec} \mapsto \lambda\langle \ell : R^P, p : P\rangle.\, \ell\, p$
  - $\mathbf{inc} \mapsto \lambda\langle M : R^{(R^P)}, N : R^P\rangle.\, M\, N$
- Induced (R^P, P)-inception algebra structure on $K_R X$
  - $\mathbf{rec}_X \mapsto \lambda\langle \ell : R^P, p : P\rangle.\, \lambda k : R^X.\, \ell\, p$
  - $\mathbf{inc}_X \mapsto \lambda\langle M : (K_R X)^{(R^P)}, N : (K_R X)^P\rangle.\, M\, (\lambda p : P.\, N\, p\, k)\, k$

▸ Inception Algebras in Logic

Logical Inception Algebra Structure

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

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

Sorted Inception Algebras

[Milner (1991)]

- Sorted sets

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

- Structures

$$V_\sigma \times \prod_i V_{|\sigma|_i} \longrightarrow A \longleftarrow A^{V_\sigma} \times A^{\prod_i V_{|\sigma|_i}} \quad \text{in } \mathcal{S}et^{\mathbb{F}[S]}$$

  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)].

Right Lambda Algebras

- Stack CPS structure

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

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

- Parameterised fixpoint
  - For

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

and

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

we have

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

Left Lambda Algebras

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

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

    $$Y_t = \mathbf{sub}\big(\, x.\, t\big(\mathbf{yield}(x, \mathbf{var}\; x)\big)\;,\; \mathbf{thunk}\big(\, x.\, t\big(\mathbf{yield}(x, \mathbf{var}\; x)\big)\,\big)\big)$$

    we have

    $$Y_t \equiv t(Y_t)$$

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

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

    For

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

    we have

    $$Y_t[x] \;\equiv\; t[x]\big(\, c.\, Y_t[c]\,\big)$$

- Producer-Consumer

Let

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

$$\mathbf{Out} \equiv \mathbf{thunk}(y.\mathbf{out}(\mathbf{out}(\mathbf{yield}(y, \mathbf{Out}))))$$

Then

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

▸ Left Lambda Algebras