On the structure of SUBSTITUTION

Abstract

The notion of substitution is ubiquitous in computer science and mathematics. However, there is as yet no theory that treats this commonsense notion in full generality. This talk is a step in this direction. Specifically, through an analysis of the concept and role of substitution in a series of examples from formal languages, algebra, type theory, combinatorics, *etc.* I will synthesise a general unifying framework for defining and studying substitution. Along the way, I will use the theory to extract correct substitution algorithms, provide initial-algebra semantics that respect substitution, establish the admissibility of the cut rule in type theories, *etc.* More speculatively, I will initiate the reduction of type theory to algebra.

Marcelo Fiore
U. of Cambridge

MFPS
26.V.06

SPIRIT OF THE TALK

Advances in mathematics occur in one of two ways. The first occurs by the solution of some outstanding problem, ...
There is a second way by which mathematics advances, ... It happens whenever some commonsense notion that had heretofore been taken for granted is discovered to be wanting, to need clarification or definition. Such foundational advances produce substantial dividens, but not right away.

Gian-Carlo Rota

What is substitution?

Substitution

From Wikipedia, the free encyclopedia

Look up ***substitution*** in Wiktionary, the free dictionary.

Substitution is the replacement of one thing with another. Specific types include:

In **mathematics**:

- Substitution rule, in calculus
- The substitution method of solving simultaneous equations
- Substitution property of equality, in mathematics
- Substitution cipher, in cryptography
- Variable substitution method in lambda calculus and mathematical logic

In **science** and **technology**:

- Substitution (chemistry), in organic chemistry
- Substitution method, in the testing of optical fiber
- Substitution mutation is synonymous with point mutation in genetics
- Virtual Reality as a substitute for Reality

In **economics**:

- Import substitution in trade
- Substitute good, in consumer theory

In **medicine**:

- Substitution therapy for opiates

In **Analytic psychology**:

- It is a defence mechanism where a person replaces one feeling or emotion for another.

Other uses:

- Chord substitution, in music
- Substitution (law), a legal right to change a judge that may be biased.
- Substitution (theatre), an acting methodology
- Substitutes, in sport
- Substitutionary Covenantal Representation in Biblical Theology

Retrieved from "http://en.wikipedia.org/wiki/Substitution"

Categories: Disambiguation

WHAT IS SUBSTITUTION?

- A philosophical question?
- A syntactic or semantic notion?
- Typed or untyped?
- How many kinds of substitution are there?
- What are the laws of substitution?
- What is the structure of substitution?

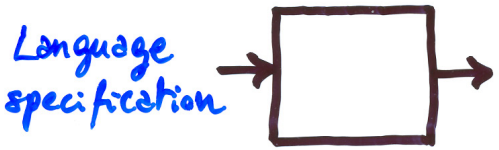
SOME MOTIVATION

- Philosophy
- Mathematics

  structural combinatorics

  higher-dimensional algebra
- Computer science

  An old dream



Example:

syntax $t ::= x \mid t_1(t_2) \mid \lambda x.t$

semantics $(\lambda x.t)\, t' \to t[t'/x]$ > need to be formalised

- Mathematics ↔ Computer science

  combinatorial structures ↔ data structure

  algebra ↔ type theory

AN ANALYSIS OF SUBSTITUTION

- Algebraic languages
- Algebraic languages with variable binding

} variables vs. occurrences

- Type-theoretic aspects
- Dependently-sorted { algebraic languages / type theory
- A general unifying framework

ALGEBRAIC LANGUAGES, DATA TYPES, ETC.

- Signatures of many-sorted operators

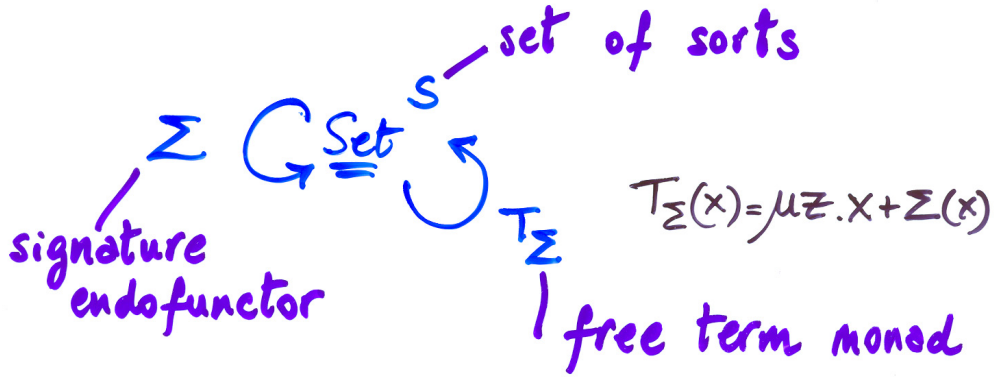
  $o : \sigma_1, \ldots, \sigma_n \to \sigma$ a data type in ML

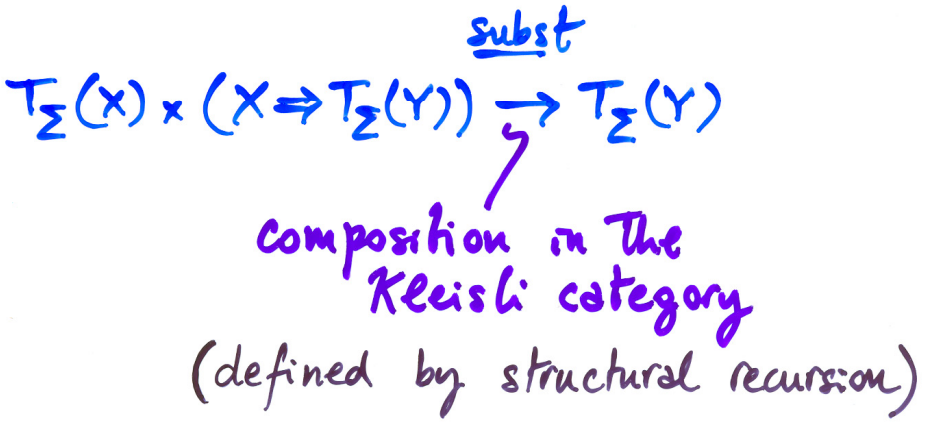
- Substitution operation

$$\frac{x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash t : \sigma \qquad \Gamma \vdash t_i : \sigma_i \ (i = 1..n)}{\Gamma \vdash t[{}^{t_1}/_{x_1}, \ldots, {}^{t_n}/_{x_n}] : \sigma}$$

- ! It is straightforward to automatically generate such a substitution program!
- ? Is it correct? In which sense?
- ? Does it terminate?
- ? What is its type?

MATHEMATICAL MODEL





- Lawvere theories

ALGEBRAIC LANGUAGES WITH VARIABLE BINDING

- Binding signatures [Aczel, ..., Power & Tanaka, ...]

  Example: Untyped/mono-sorted λ-calculus

  V, Λ

  $\mathrm{var} : V \to \Lambda$

  $\mathrm{app} : \Lambda, \Lambda \to \Lambda$

  $\mathrm{abs} : [V]\Lambda \to \Lambda$

- Substitution operation

$$\frac{x_1, \ldots, x_n, x \vdash t \qquad x_1, \ldots, x_n \vdash u}{x_1, \ldots, x_n \vdash t[u/x]}$$

! It is possible to automatically generate such a substitution program!

issues: Correctness (termination, ...)
Implementation of binding (de Bruijn, ...)
Typing (single variable vs. simultaneous substitution)

MATHEMATICAL MODEL

[Fiore, Plotkin, Turi]

- Main observation
  - $\Lambda = \{\Lambda(\Gamma)\}_{\Gamma \subseteq_{fin} Var}$ terms in context
  - $\Lambda(\Gamma) \times (\Gamma \Rightarrow \Delta) \to \Lambda(\Delta)$

    $t, \rho \mapsto t[\rho]$ renaming action

$$\begin{cases} t[id] = t \\ t[\rho][\rho'] = t[\rho \cdot \rho'] \end{cases}$$

$\Lambda \in \underline{Set}^{\mathbb{F}}$

$\mathbb{F}$: a category of contexts and context renamings

$\underline{Set}$: a richer universe of types

- Also:

$V \in \underline{Set}^{\mathbb{F}}$

$V(\Gamma) = \{x \mid x \in \Gamma\}$

OPERATIONS = JUDGEMENTS

$\underline{var} : V \to \Lambda$ $\qquad \dfrac{x \in \Gamma}{var(x) \in \Lambda(\Gamma)}$

$\underline{app} : \Lambda \times \Lambda \to \Lambda$ $\qquad \dfrac{t \in \Lambda(\Gamma) \quad t' \in \Lambda(\Gamma)}{app(t,t') \in \Lambda(\Gamma)}$

$\underline{abs} : \Lambda^{V} \to \Lambda$ $\qquad \dfrac{t \in \Lambda(\Gamma, x)}{abs(x.t) \in \Lambda(\Gamma)}$

Fact:

$$(\Lambda^{V})(\Gamma) = \Lambda(\underbrace{\Gamma, *}_{\text{context extension}})$$

—a universal construction—

SINGLE-VARIABLE SUBSTITUTION

$$\Sigma \curvearrowright \underline{Set}^{\mathbb{F}} \curvearrowleft T_{\Sigma}$$

signature endofunctor — free term monad

$$\Sigma(X) = X^{\overbrace{2}^{\text{arity of application}}} + X^{\overbrace{V}^{\text{arity of abstraction}}}$$

$$T_{\Sigma}(X) = \mu Z.\, X + \Sigma(Z)$$

- λ-terms (up-to α-equivalence)

$$\Lambda = T_{\Sigma}(V)$$

That is,

$$\Lambda = V + \Lambda \times \Lambda + \Lambda^{V}$$

- Substitution

$$\Lambda^{V} \times \Lambda \longrightarrow \Lambda$$

A DECOMPOSITION OF SUBSTITUTION

$(TV)^V \times TV$ — $t[x_1 \dots x_n, x], u[x_1 \dots x_n]$

(1) $\cong$ — $t[x_1 \dots x_n, *], u[x_1 \dots x_n]$

$T(V+1) \times TV$

(2) $\downarrow$

$T(V+1) \times (TV)^{V+1}$ — $t[x_1 \dots x_n, *], \begin{pmatrix} x_1 \mapsto x_1 \\ x_n \mapsto x_n \\ * \mapsto u \end{pmatrix}$

(3) $\downarrow$ — $\downarrow$

TV — $t[x_1 \dots x_n, u[x_1 \dots x_n]]$

---

(1) $(TV)^V \cong T(V+1)$

↳ structural recursion respecting bindings, induced by

$$\begin{array}{ccc} Set^F & \xrightarrow{\Sigma} & Set^F \\ {\scriptstyle (-)^V}\downarrow & \cong & \downarrow{\scriptstyle (-)^V} \\ Set^F & \xrightarrow[\Sigma]{} & Set^F \end{array}$$

(2) $TV \to (TV)^{V+1}$

↳ base case

(3) Kleisli composition = textual substitution

DIRECT IMPLEMENTATION

```
(*** Contexts ***)

type Ctxt = int ;

val EmptyCtxt = 0 ;

fun Cext C = C+1 ;

fun new C = ...

fun up x = ...

fun old x = ...
```

[de Bruijn]

	levels	indices
	$C+1$	1
	x	$x+1$
	x	$x-1$

```
(*** Terms ***)

datatype
  Lam = var of Ctxt | app of Lam * Lam | abs of Lam ;
```

```
(*** Renamings ***)

type
  Renaming = Ctxt * (int -> int) * Ctxt ;

fun Rext (C,r,D)
= ( Cext C ,
    fn x
    => if x = new C then new D else up( r( old x ) ) ,
    Cext D ) ;

(*** Renaming actions ***)

  infix act ;

  fun ( var i ) act (_,r,_) = var(r i)
    | ( app(t1,t2) ) act R = app( t1 act R , t2 act R )
    | ( abs t ) act R = abs( t act (Rext R) ) ;
```



```
(*** Single-variable substitution ***)

fun subst C ( var x , u )
    = if x = new C then u else var(old x)
  | subst C ( app(t1,t2) , u )
    = app( subst C (t1,u) , subst C (t2,u) )
  | subst C ( abs(t) , u )
    = let
        fun Up C t
        = t act ( C , up , Cext C ) ;
        fun Swap C t
        = t act
          ( Cext(Cext C) ,
            fn x
            => if x = new( Cext C ) then up( new C )
               else if x = up( new C ) then new( Cext C )
               else x ,
            Cext(Cext C) ) ;
      in
        abs( subst (Cext C) ( Swap C t , Up C u ) )
      end ;
```

What is the type of SIMULTANEOUS SUBSTITUTION ?

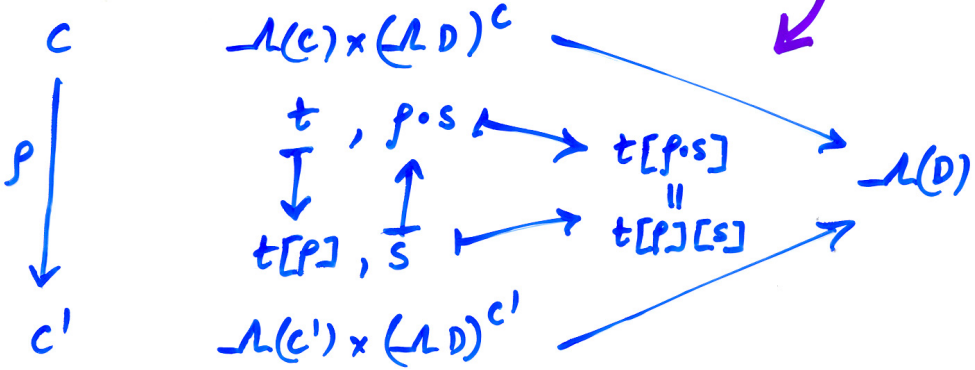
$$\left\{ \underline{subst}_{C,D} : \mathcal{L}(C) \times (\mathcal{L}D)^C \to \mathcal{L}(D) \right\}_{\text{nat. in } C,D \in \mathbb{F}}$$

guarantees that substitution is compatible with renaming

Could be encoded as:

$$\left\{ \underline{subst}_D : \sum_{C \in \mathbb{F}} \mathcal{L}(C) \times (\mathcal{L}D)^C \to \mathcal{L}(D) \right\}_{\text{nat. in } D \in \mathbb{F}}$$

but this is not quite right, as naturality in C is lost.



? How can we ensure it ?

THE TYPE OF SUBSTITUTION

$$\frac{\left\{ \mathcal{L}(C) \times (\mathcal{L}D)^C \xrightarrow{\underline{subst}_{C,D}} \mathcal{L}(D) \right\}_{\text{nat. in } C,D \in \mathbb{F}}}{\mathcal{L} \circ \mathcal{L} \xrightarrow[\underline{subst}]{} \mathcal{L} \quad \text{in } \underline{Set}^{\mathbb{F}}}$$

where

$$(\mathcal{L} \circ \mathcal{L})(D) = \int^{C \in \mathbb{F}} \mathcal{L}(C) \times (\mathcal{L}D)^C$$

} the quotient of $\sum_{C \in \mathbb{F}} \mathcal{L}(C) \times (\mathcal{L}D)^C$ under $t, \rho \cdot s \sim t[\rho], s$

NB: $\underline{subst}_{C,D}(t, \rho \cdot s) = \underline{subst}_{C',D}(t[\rho], s)$

THE SUBSTITUTION MONOIDAL STRUCTURE (OR COMPOSITION)

- $X, Y \in \underline{\mathbf{Set}}^{\mathbb{F}} \mapsto X \circ Y \in \underline{\mathbf{Set}}^{\mathbb{F}}$

$$(X \circ Y)(\mathbb{D}) = \int^{\mathbb{C} \in \mathbb{F}} X(\mathbb{C}) \times (Y\mathbb{D})^{\mathbb{C}}$$

$$x, \langle y_{pi} \rangle_{i \in C} \sim x[p], \langle y_i \rangle_{i \in C'}$$

- There are coherent natural isomorphisms:

$$(X \circ Y) \circ Z \cong X \circ (Y \circ Z)$$

$$V \circ X \cong X \cong X \circ V$$

$$(x \langle y_i \rangle_i) \langle z_j \rangle_j \leftrightarrow x \langle y_i \langle z_j \rangle_j \rangle_i$$

$$i_0 \langle x_i \rangle_i \leftrightarrow x_{i_0}$$

$$x \langle i \rangle_i \leftrightarrow x$$

SPECIFICATION OF SUBSTITUTION

$$X \circ X \xrightarrow{s} X \xleftarrow{v} V$$

s: substitution operation

v: provision of variables

satisfying monoid laws

- Associativity = substitution lemma

$$\begin{array}{ccc} (x\langle y_i \rangle_i)\langle z_j \rangle_j & \leftrightarrow & x\langle y_i \langle z_j \rangle_j \rangle_i \\ \downarrow & & \downarrow \\ (x[y_i]_i)\langle z_j \rangle_j & & x\langle y_i [z_j]_j \rangle_i \\ \downarrow & & \downarrow \\ (x[y_i]_i)[z_j]_j & = & x[y_i[z_j]_j]_i \end{array}$$

- $v_{i_0}[x_i]_i = x_{i_0}$
- $x[v_i]_i = x$

A GENERAL ABSTRACT VIEW

- Mathematical universe

$I/\mathcal{U}$ — pointed objects = objects with a provision of variables.

$F \downarrow$

$\Sigma \curvearrowright \mathcal{U}$

signature endofunctor with an F-strength

universe of types with a substitution tensor product $\circ$ and an object of variables I, the unit.

$$\Sigma(X)\circ Y \longrightarrow \Sigma(X\circ Y)$$

$$\underline{st}_{X,\,I\downarrow Y}$$

- Mathematical models: Σ-monoids (substitution algebras)

$$\begin{array}{ccccc} & & \Sigma X & & \\ & & \downarrow x & & \\ X\circ X & \xrightarrow{s} & X & \xleftarrow{v} & I \end{array} \quad \Big\} \text{ a monoid}$$

$$\begin{array}{ccccc} \Sigma(X)\circ X & \xrightarrow{\underline{st}_{X,v}} & \Sigma(X\circ X) & \xrightarrow{\Sigma s} & \Sigma X \\ x\circ id \downarrow & & & & \downarrow x \\ X\circ X & & \xrightarrow{\quad s \quad} & & X \end{array}$$

THEOREM

The free monoid

$$\underline{Mon}(\mathcal{U}) \overset{Y}{\longrightarrow} \mathcal{U}$$

construction generalises to

$$\Sigma\text{-}\underline{Mon}(\mathcal{U}) \overset{Y}{\longrightarrow} \mathcal{U} \qquad (M)$$

$M0 = \mu Z.\, I + \Sigma Z$ is the initial Σ-monoid

where

$$M(X) = \mu Z.\, I + X\circ Z + \Sigma Z$$

with monoid structure uniquely determined by

$$\begin{array}{ccc} I\circ MX & & \\ e\circ id \downarrow & \overset{\cong}{\searrow} & \\ MX\circ MX & \xrightarrow{s} & MX \end{array}$$

$$\begin{array}{ccccc} (X\circ MX)\circ MX \cong X\circ(MX\circ MX) & \xrightarrow{id\circ s} & X\circ MX \\ a\circ id \downarrow & & \downarrow a \\ MX\circ MX & \xrightarrow{\quad s \quad} & MX \end{array}$$

$$\begin{array}{ccccc} \Sigma(MX)\circ MX & \xrightarrow{\underline{st}_{MX,e}} & \Sigma(MX\circ MX) & \xrightarrow{\Sigma s} & \Sigma MX \\ t\circ id \downarrow & & & & \downarrow t \\ MX\circ MX & & \xrightarrow{\quad s \quad} & & MX \end{array}$$

where $[e,a,t] : I + X\circ MX + \Sigma MX \xrightarrow{\cong} MX$.

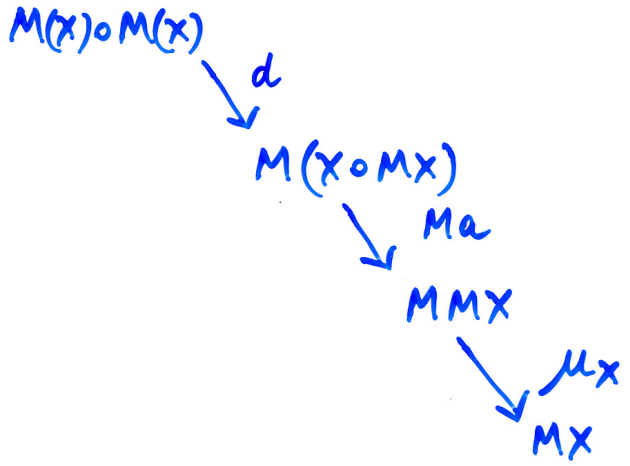
IMPLEMENTATION

[Stoughton]

```
(*** Simultaneous substitution ***)

type
  Substitution = Ctxt * (int -> Lam) * Ctxt ;

fun Sext (C,s,D)
  = ( Cext C ,
      fn x
      => if x = new C then var(new D)
         else ( s(old x) ) act ( D , up , Cext D )
      Cext D ) ;

fun subst ( var x ) (_,s,_)
    = s x
  | subst ( app(t1,t2) ) S
    = app( subst t1 S , subst t2 S )
  | subst ( abs(t) ) S
    = abs( subst t (Sext S) ) ;
```


A DECOMPOSITION OF SUBSTITUTION

The substitution operation decomposes as



APPLICATION TO OTHER KINDS OF SUBSTITUTION

occurrences vs. variables

- Grammars

$$w_0 X_1 w_1 \dots w_{n-1} X_n w_n \quad , \quad X_i \to W_i \ (i=1..n)$$

$$\downarrow$$

$$w_0 W_1 w_1 \dots w_{n-1} W_n w_n$$

- Proof trees

$$\frac{P_1, \dots, P_n}{P} \quad , \quad \begin{array}{c} \nabla \\ P_1 \end{array} \ \dots \ \begin{array}{c} \nabla \\ P_n \end{array}$$

$$\downarrow$$

$$\frac{\begin{array}{c} \nabla \\ P_1 \end{array} \ \dots \ \begin{array}{c} \nabla \\ P_n \end{array}}{P}$$

MATHEMATICAL MODELS

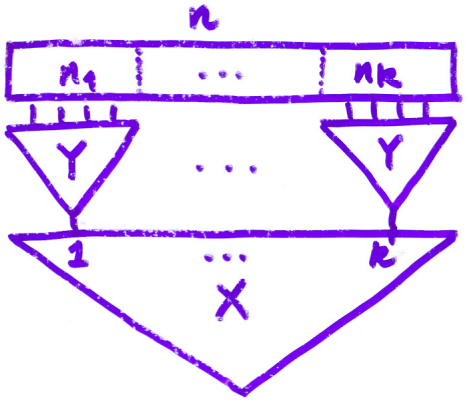
~ Linear Species ~ [Joyal]

- A "context" is just a sequence/linear-order of tokens.
- Model:

  $\underline{Set}^{\mathbb{N}}$, $\mathbb{N}$ = the set of natural numbers

  Substitution Tensor product:

$$(X \circ Y)(n) = \int^{k \in \mathbb{N}} X(k) \times \sum_{n_1 + \dots + n_k = n} \prod_{i=1}^{k} Y(n_i)$$



- Monoids = [planar] operads.

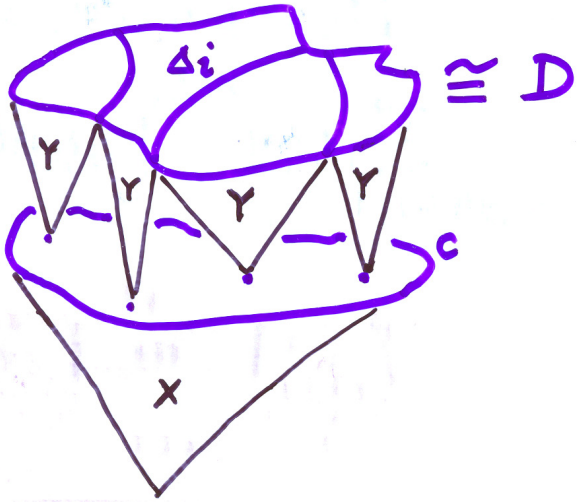
MATHEMATICAL MODELS

~ SPECIES ~ [Joyal]

- A "context" is a set of tokens.
- Model:

  $\underline{\mathrm{Set}}^{\mathbb{B}}$, $\mathbb{B}$ = the category of finite sets and bijections

Substitution tensor product:



$$(X \circ Y)(D) = \int^{C \in \mathbb{B}} X(C) \times \int^{\Delta \in \mathbb{B}^C} \prod_{i \in C} Y(\Delta_i) \times \mathbb{B}\left(\dot{+}_i \Delta_i, D\right)$$

- Monoids = [symmetric] operads

MANY-SORTED MODELS

- Many-sorted contexts

$C = (x_1 : \sigma_1, \ldots, x_n : \sigma_n) \leftrightarrow S \to \mathbb{F} : \sigma \mapsto \{x \mid (x : \sigma) \in C\}$

(S: set of sorts; $\mathbb{F}$: category of untyped contexts)

$\mathbb{F}[S] =^{\text{def}} \mathbb{F}^S$ = the category of S-sorted contexts

- Model:

$\left(\underline{\mathrm{Set}}^{\mathbb{F}[S]}\right)^S$ $\quad X_\sigma(C)$ = σ-sorted elements of type X in context C

Substitution tensor product:

$$(X \circ Y)_\sigma(D) = \int^{C \in \mathbb{F}[S]} X_\sigma(C) \times \prod_{(x:\tau) \in C} Y_\tau(D)$$

Unit:

$V = \{V_\sigma\}_{\sigma \in S}$, $V_\sigma(C) = C(\sigma)$

ACHIEVEMENTS

- General specification of substitution
  - single variable and simultaneous substitution
  - substitution for variables and occurrences
  - monoids w.r.t. substitution monoidal structure (clubs [Kelly])
- Framework for syntactic settings
  - initial algebra semantics that respects substitution
- Extraction of substitution programs
  - correctness
  - automation
- Reduction of simple type theory to algebra
  - E.g.

$$\frac{T \circ T \longrightarrow T}{T_\sigma(C) \times \prod_{(x:\tau)\in C} T_\tau(D) \longrightarrow T_\sigma(D)}$$

$\Rightarrow$ the cut rule

$$\frac{C \vdash t : \sigma \qquad (D \vdash t_x : \tau)_{(x:\tau)\in C}}{D \vdash t[t_x/x] : \sigma}$$

is admissible

- Generalised species of structures
  - many-sorted species

DEPENDENTLY-SORTED ALGEBRA

[Martin-löf, Cartmell, MaKKai]

Example: The theory of 2-categories

- Dependent sorts:
  - $\vdash 0 : *$
  - $x, y : 0 \vdash A(x,y) : *$
  - $x, y : 0,\ f, g : A(x,y) \vdash C(x,y,f,g) : *$
- Operations:
  - $x, y, z : 0,\ f : A(x,y),\ g : A(y,z) \vdash \underline{comp}(x,y,z,f,g) : A(x,z)$
  - $x : 0 \vdash \underline{id}(x) : A(x,x)$
  - $x, y : 0,\ f, g, h : A(x,y),\ \alpha : C(x,y,f,g),\ \beta : C(x,y,g,h) \vdash \underline{vcomp}(x,y,f,g,h,\alpha,\beta) : C(x,y,f,h)$

  ...
- Equations:

  ...

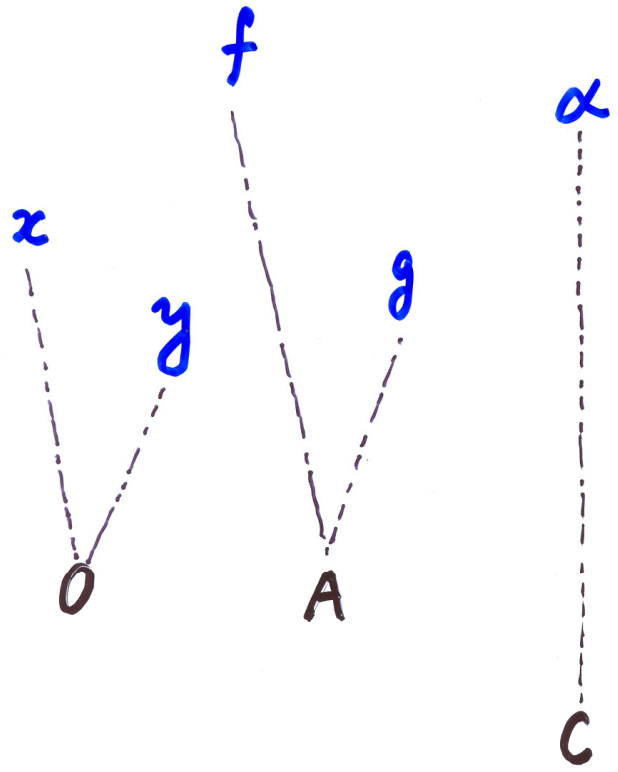
SYNTAX IS PERVASIVE ... AND PERVERSE!

!? What is a dependently-sorted context !?

Example: A context for $x \overset{f}{\underset{g}{\Downarrow \alpha}} y$.

$$x, y : O,\ f, g : A(x, y),\ \alpha : C(x, y, f, g)$$

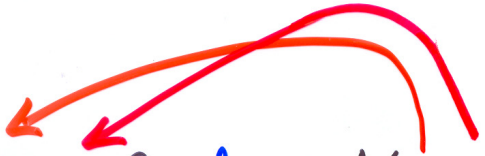
GRAPHICAL REPRESENTATION



SYNTAX IS PERVASIVE ... AND PERVERSE!

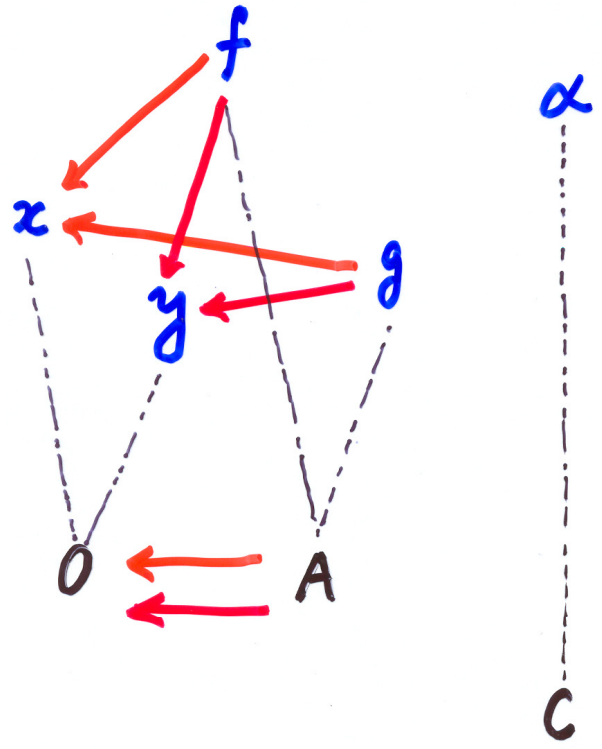
!? What is a dependently-sorted context !?

Example: A context for $x \overset{f}{\underset{g}{\Downarrow \alpha}} y$.



$$x, y : O,\ f, g : A(x, y),\ \alpha : C(x, y, f, g)$$

GRAPHICAL REPRESENTATION

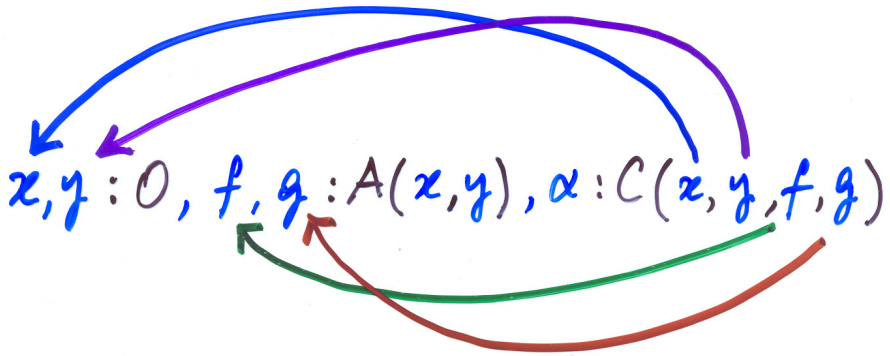


$$x, y : O \vdash A(x, y) : *$$

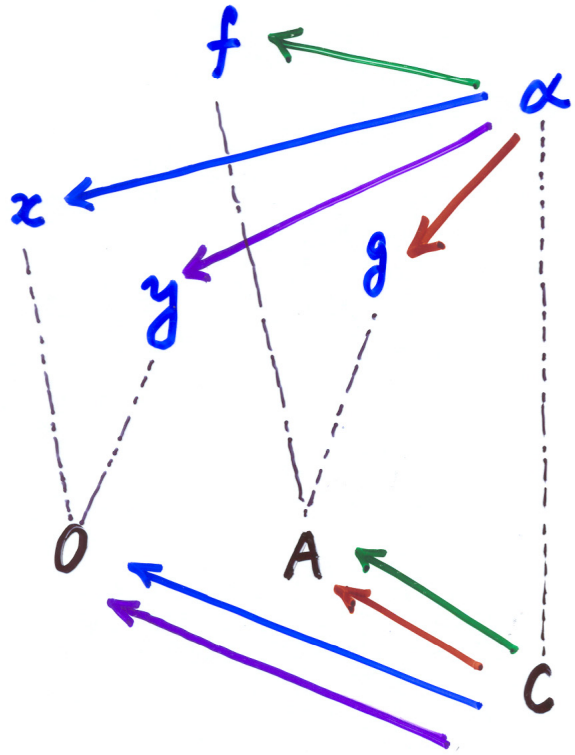
SYNTAX IS PERVASIVE ... AND PERVERSE!

!? What is a dependently-sorted context !?

Example: A context for $x \overset{f}{\underset{g}{\Downarrow \alpha}} y$.



GRAPHICAL REPRESENTATION

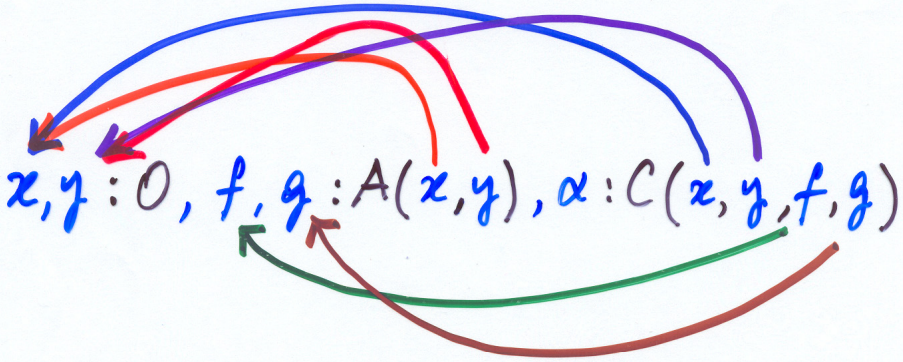


$x, y : O, f, g : A(x,y) \vdash C(x,y,f,g) : *$

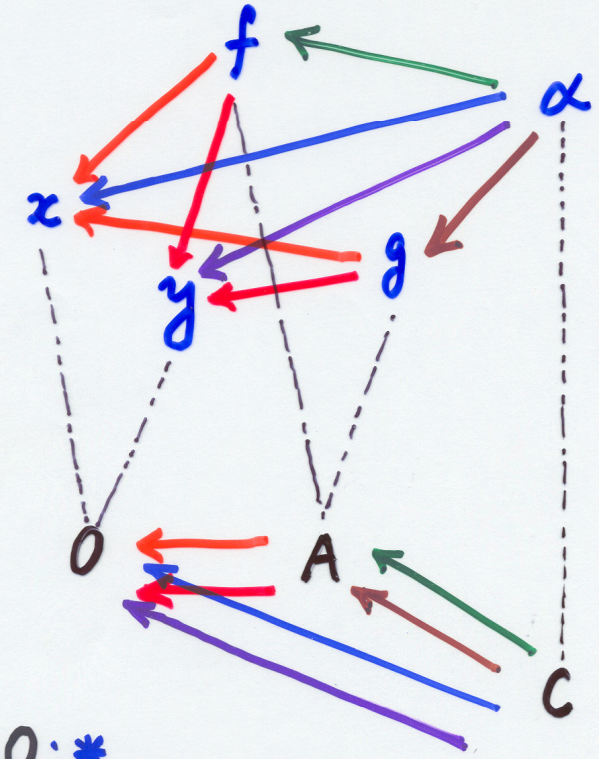
SYNTAX IS PERVASIVE ... AND PERVERSE!

!? What is a dependently-sorted context !?

Example: A context for $x \overset{f}{\underset{g}{\Downarrow \alpha}} y$.



GRAPHICAL REPRESENTATION



$: \mathbb{D} \to \mathbb{F}$

$\vdash O : *$

$x, y : O \vdash A(x, y) : *$

$x, y : O, f, g : A(x, y) \vdash C(x, y, f, g) : *$

↳ Category of dependent sorts:

$\mathbb{D} = \boxed{O \leftleftarrows A \leftleftarrows C}$

MATHEMATICAL MODEL

- Category of dependent $\mathbb{D}$-sorted contexts:

$$\mathbb{F}[\mathbb{D}] =^{def} \mathbb{F}^{\mathbb{D}}$$

- Model:

$$(\underline{Set}^{\mathbb{F}[\mathbb{D}]})^{\mathbb{D}}$$

$$x \in X_\sigma(C) \rightsquigarrow C \vdash x : \sigma(\dots x_i \dots)$$

where $x_i = X_{d_i}(C)(x)$

for $d_i : \sigma \to \sigma_i$ in $\mathbb{D}$

- Substitution tensor product:

$$(X \circ Y)_\sigma(D) = \int^{C \in \mathbb{F}[\mathbb{D}]} X_\sigma(C) \times \lim_{(z:\tau) \in C} Y_\tau(D)$$

$$\dots, z_i : \sigma_i(\dots), \dots \vdash x : \sigma(\dots x' \dots)$$

$$\dots \quad D \vdash y_i : \sigma_i(\dots)$$

$$\dots$$

$$\mapsto \quad D \vdash x[\dots y_i \dots] : \sigma(\dots x'[\dots y_i \dots] \dots)$$

▶ Next in the research programme:
Reduce type theory to algebra!

MIXED MODELS

Example: DILL = Dual Intuitionistic Linear Logic
[Barber & Plotkin]

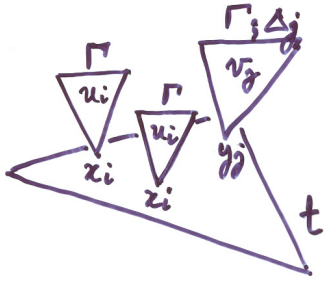
$$\Gamma ; \Delta \vdash t : \tau$$

Γ: intuitionistic (cartesian) context

Δ: linear (symmetric monoidal) context

- Cut rule:

$$\frac{x_1 : \sigma_1, \dots, x_m : \sigma_m ; y_1 : \tau_1, \dots, y_n : \tau_n \vdash t : \alpha \qquad \Gamma ; - \vdash u_i : \sigma_i \ (1 \le i \le m) \qquad \Gamma ; \Delta_j \vdash v_j : \tau_j \ (1 \le j \le n)}{\Gamma ; \Delta_1, \dots, \Delta_n \vdash t[u_i/x_i, v_j/y_j] : \alpha}$$



new feature — absent in mathematical examples —

Mathematical Model

- The category of (mono-sorted) mixed contexts $\mathbb{M}$ is the free symmetric monoidal category over the s.m. theory:
  PROP[MacLane]

$$L \to I \leftarrow I \oplus I \qquad (0 \to I,\ \text{commutative monoid})$$

linear variables can become intuitionistic

& weakening & contraction

Type theoretically:

$$\frac{\Gamma; x, \Delta \vdash t}{\Gamma, x; \Delta \vdash t}$$

That is,

$$B \to F \leftarrow F \otimes F \qquad (0 \to F,\ \text{c.m.}) \qquad \text{in } \mathcal{E} \text{ s.m.}$$

$$\mathbb{M} \xrightarrow[\text{s.m.}]{} \mathcal{E}$$

- Explicit description

$$C \xrightarrow{\ \rho\ } D, \quad \Gamma : C \to (L \leq I), \quad \Delta : D \to (L \leq I)$$

$$\left(= \begin{array}{l} \text{int. vars remain int.} \\ \forall x \in C. \\ (x:I) \in \Gamma \Rightarrow (\rho x : I) \in \Delta \end{array} \right)$$

s.t. $\forall (y:L) \in \Delta.\ \exists!\, (x:L) \in \Gamma.\ \rho(x) = y$

(Lin. vars are lin.)

Context Indexing

in Cat

$$\mathbb{M} \hookleftarrow \mathbb{F} \xrightarrow{\ 1\ } \mathbb{F} \times \mathbb{F}$$

induces

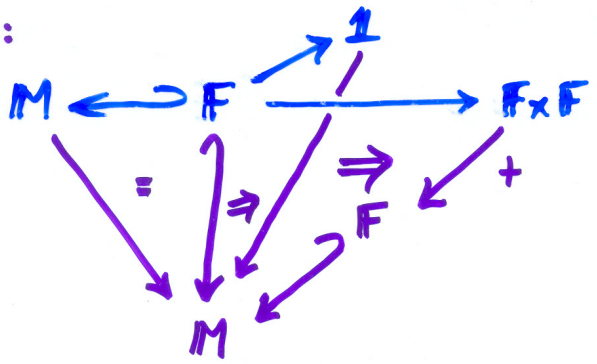
$$\mathcal{M} : \mathbb{M}^{op} \to \mathrm{Cat}$$

$$\langle \ldots, L, \ldots, I, \ldots \rangle \mapsto \ldots \times \mathbb{M} \times \ldots \times \mathbb{F} \times \ldots$$

CONTEXT INDEXING

In fact:

in $\underline{\mathbb{C}at}^{\mathbb{M}}$



induces

$$\mathcal{M} : \mathbb{M}^{op} \longrightarrow \underline{\mathbb{C}at}^{\mathbb{M}}$$

$$\langle \ldots, L, \ldots, I, \ldots \rangle \longmapsto \ldots \times \mathbb{M} \times \ldots \times \mathbb{F} \times \ldots \xrightarrow{\oplus} \mathbb{M}$$

MIXED-SUBSTITUTION TENSOR PRODUCT

- Model:

$$\underline{Set}^{\mathbb{M}}$$

Substitution tensor product:

$$(X \circ Y)(D) = \int^{C \in \mathbb{M}} X(C) \times \int^{\Delta \in \mathcal{M}(C)} \prod_{i \in |C|} Y(\Delta_i) \times \mathbb{M}(\underline{\oplus_C}(\Delta), D)$$

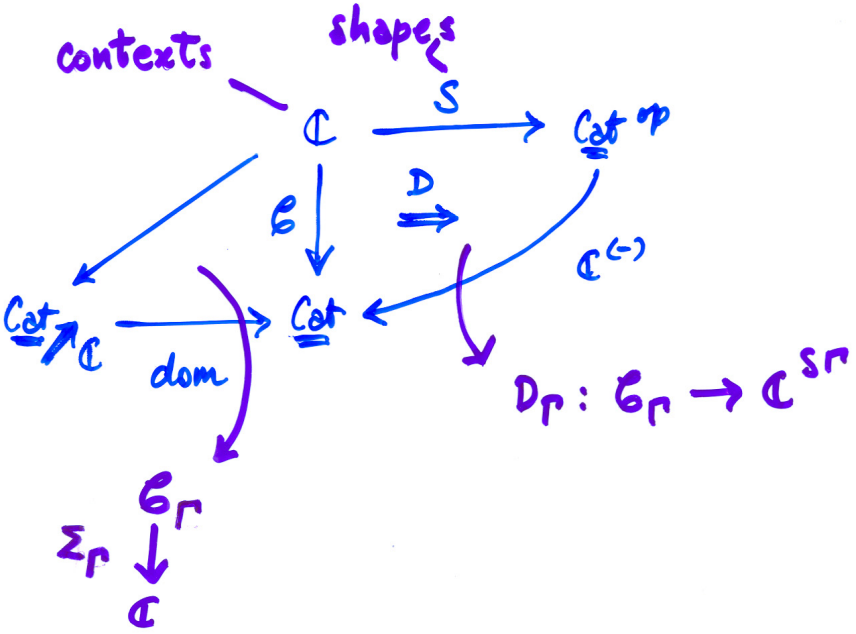
$$\oplus_C : \mathcal{M}(C) \to \mathbb{M}$$

new feature — absent in mathematical examples —

- Monoids = MIXED OPERADS → Generalise and combine Lawvere theories and [symmetric] operads.

- A COMBINATORIAL model of DILL ... and more.

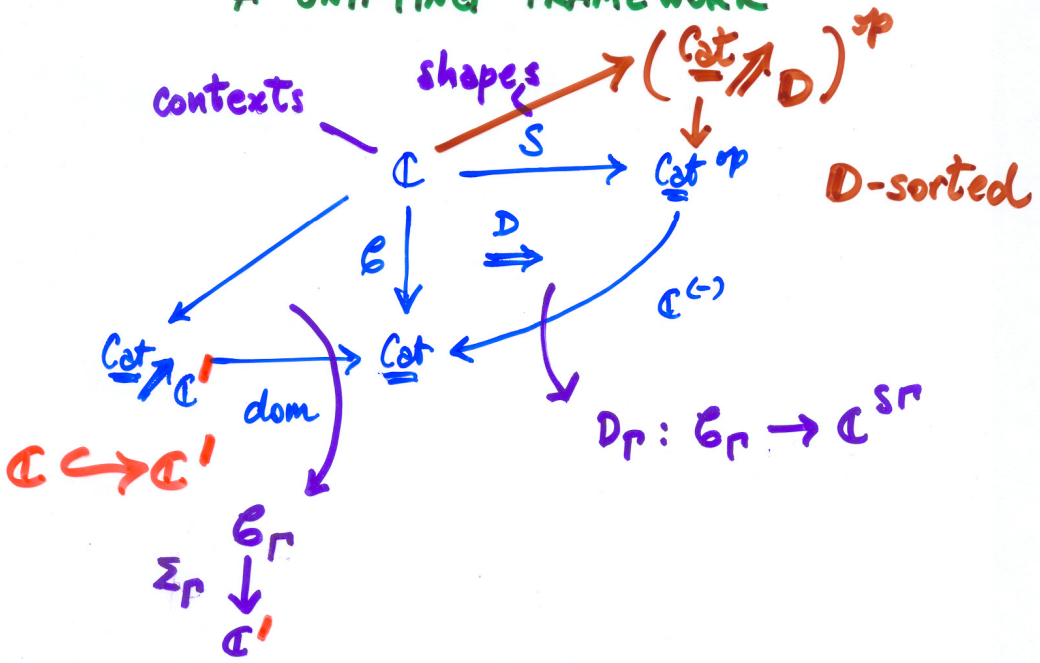
A UNIFYING FRAMEWORK



- Substitution tensor product (?)

$$(X \circ Y)(C) = \int^{\Gamma \in \mathbb{C}} X(\Gamma) \cdot \int^{\Delta \in \mathcal{C}_\Gamma} \left(\lim_{i \in S\Delta} Y(D_\Gamma \Delta) i \right) \cdot [\Sigma_\Gamma \Delta, C]$$

FURTHER GENERALISATIONS
A UNIFYING FRAMEWORK



- Substitution tensor product (?)

$$(X \circ Y)(C) = \int^{\Gamma \in \mathbb{C}} X(\Gamma) \cdot \int^{\Delta \in \mathcal{C}_\Gamma} \left(\lim_{i \in S\Delta} Y(D_\Gamma \Delta) i \right) \cdot [\Sigma_\Gamma \Delta, C]$$

TOWARDS A GENERAL THEORY

- For a class of examples (including the ones in this talk and more) from both computer science and mathematics we obtain a monoidal structure.
- In fact, these arise as composition in the Kleisli category for (cartesian) monads on profunctors (by distributive laws [Power & Tanaka] or liftings of Kleisli structures [Fiore, Gambino & Hyland]).

DIRECTIONS

- General abstract Theory of substitution tensor products
  - Categories of contexts as free monoidal theories
  - Comparison with / extension to clubs [Kelly]
- Equational theories (with Chungkil Hur).
  - Rewriting
- Reduction of dependent type theory to algebra.
  - NbE / TDPE
- Extraction of syntax from models.
  - Combinatorial data types
  - Signatures
- Generalised logic (= calculus of coends + ...) type theoretically.
  - Compilation by interpretation