

# 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 dividends, but not right away.

Gian-Carlo Rota

# What is substitution?

## Substitution

From Wikipedia, the free encyclopedia

**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

- This page was last modified 21:38, 20 May 2006.
  - All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details).
- Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc.



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

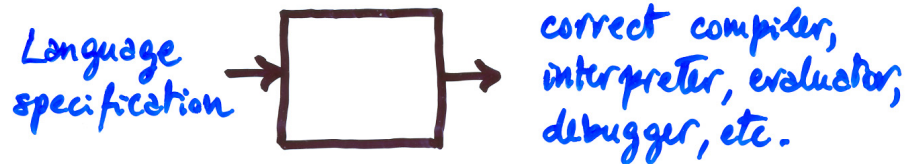
## WHAT IS SUBSTITUTION?

- 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:

syntex  $t :: x \mid t_1(t_2) \mid \lambda x. t$   
semantics  $(\lambda x. t) t' \rightarrow t[t'/x]$   $\rangle$  need to be formalised

- Mathematics  $\leftrightarrow$  Computer science
  - combinatorial structures  $\leftrightarrow$  data structure
  - algebra  $\leftrightarrow$  type theory

## AN ANALYSIS OF SUBSTITUTION

- Algebraic languages
  - Algebraic languages with variable binding
- } variables vs. occurrences
- Type-theoretic aspects
  - Dependently-sorted  $\left\{ \begin{array}{l} \text{algebraic languages} \\ \text{type theory} \end{array} \right.$
  - A general unifying framework

## ALGEBRAIC LANGUAGES, DATA TYPES, ETC.

- Signatures of many-sorted operators

$0 : \sigma_1, \dots, \sigma_n \rightarrow \sigma$  a data type in ML

- Substitution operation

$$\frac{z_1 : \sigma_1, \dots, z_n : \sigma_n \vdash t : \sigma \quad \Gamma \vdash b_i : \sigma_i \ (i=1..n)}{\Gamma \vdash t[b_1/z_1, \dots, b_n/z_n] : \sigma}$$

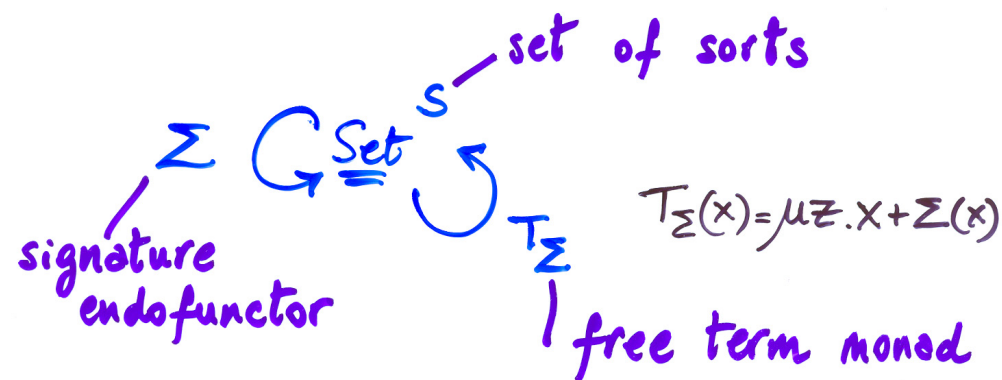
! It is straightforward to automatically generate such a substitution program!

? Is it correct? In which sense?

? Does it terminate?

? What is its type?

## MATHEMATICAL MODEL



$$T_\Sigma(X) \times (X \Rightarrow T_\Sigma(Y)) \xrightarrow{\text{subst}} T_\Sigma(Y)$$

composition in the Kleisli category  
(defined by structural recursion)

► Lawvere theories



## ALGEBRAIC LANGUAGES WITH VARIABLE BINDING

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

Example: Untyped/mono-sorted  $\lambda$ -calculus

$V, \mathcal{L}$

$\text{var} : V \rightarrow \mathcal{L}$

$\text{app} : \mathcal{L}, \mathcal{L} \rightarrow \mathcal{L}$

$\text{abs} : [V] \mathcal{L} \rightarrow \mathcal{L}$

- Substitution operation

$$\frac{x_1, \dots, x_n, x \vdash t \quad x_1, \dots, x_n, u \vdash u}{x_1, \dots, x_n \vdash t[u/x]}$$

!! 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, Turbitt]

- Main observation

•  $\mathcal{L} = \{ \mathcal{L}(\Gamma) \}_{\Gamma \in \text{fnVar}}$  terms in context

•  $\mathcal{L}(\Gamma) \times (\Gamma \Rightarrow \Delta) \rightarrow \mathcal{L}(\Delta)$

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

$$\begin{cases} t[\alpha] = t \\ t[p][p'] = t[p \cdot p'] \end{cases}$$

$$\mathcal{L} \in \underline{\text{Set}}^{\mathbb{F}}$$

a category of contexts and context renamings

a richer universe of types

- Also:

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

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

## OPERATIONS = JUDGEMENTS

$$\underline{\text{var}}: V \rightarrow \mathcal{L}$$

$$\frac{z \in \Gamma}{\underline{\text{var}}(z) \in \mathcal{L}(\Gamma)}$$

$$\underline{\text{app}}: \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$$

$$\frac{t \in \mathcal{L}(\Gamma) \quad t' \in \mathcal{L}(\Gamma)}{\underline{\text{app}}(t, t') \in \mathcal{L}(\Gamma)}$$

$$\underline{\text{abs}}: \underbrace{\mathcal{L}^V}_{\gamma} \rightarrow \mathcal{L}$$

$$\frac{t \in \mathcal{L}(\Gamma, z)}{\underline{\text{abs}}(x.t) \in \mathcal{L}(\Gamma)}$$

Fact:

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

— a universal construction —

## SINGLE-VARIABLE SUBSTITUTION

$$\Sigma \hookrightarrow \text{Set}^{\mathbb{F}} \hookrightarrow T_{\Sigma}$$

signature endofunctor      free term monad

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

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

- $\lambda$ -terms (up-to  $\alpha$ -equivalence)

$$\mathcal{L} = T_{\Sigma}(V)$$

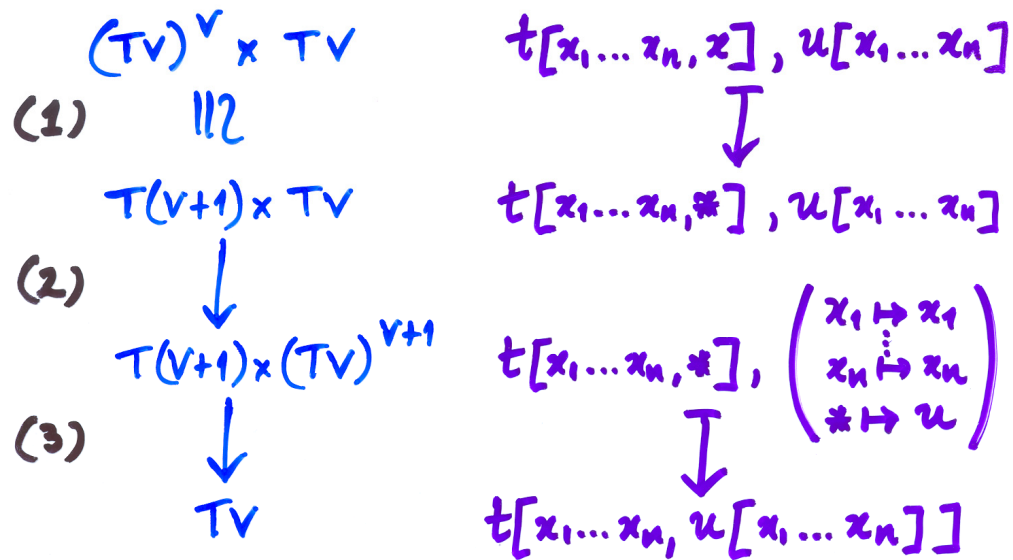
That is,

$$\mathcal{L} = V + \mathcal{L} \times \mathcal{L} + \mathcal{L}^V$$

- Substitution

$$\mathcal{L}^V \times \mathcal{L} \rightarrow \mathcal{L}$$

# A DECOMPOSITION OF SUBSTITUTION



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

↳ structural recursion respecting bindings, induced by

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

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

↳ 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 = ...

Levels  
C+1

x

x

Indices  
1

x+1

x-1

[de Bruijn]

(\*\* 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?

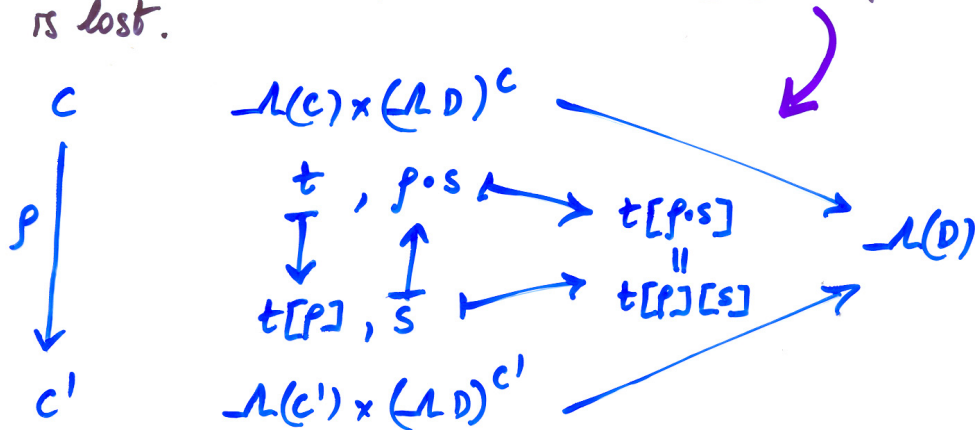
$$\left\{ \text{subst}_{c,D} : \Lambda(c) \times (\Lambda D)^c \rightarrow \Lambda(D) \right\}_{\text{nat. in } c, D \in F}$$

guarantees that  
substitution is compatible  
with renaming

Could be encoded as:

$$\left\{ \text{subst}_D : \sum_{c \in F} \Lambda(c) \times (\Lambda D)^c \rightarrow \Lambda(D) \right\}_{\text{nat. in } D \in F}$$

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



[?] How can we ensure it?

## THE TYPE OF SUBSTITUTION

$$\frac{\left\{ \Lambda(c) \times (\Lambda D)^c \xrightarrow{\text{subst}_{c,D}} \Lambda(D) \right\}_{\text{nat. in } c, D \in F}}{\Lambda \circ \Lambda \xrightarrow{\text{subst}} \Lambda \text{ in } \text{Set}^F}$$

where

$$(\Lambda \circ \Lambda)(D) = \int_{c \in F} \Lambda(c) \times (\Lambda D)^c$$

} the quotient of  
 $\sum_{c \in F} \Lambda(c) \times (\Lambda D)^c$   
 under  $t, p.s \sim t[p], s$

NB:  $\text{subst}_{c,D}(t, p.s) = \text{subst}_{c',D}(t[p], s)$

(OR COMPOSITION)

## THE SUBSTITUTION MONOIDAL STRUCTURE

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

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

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

- 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

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

substitution operation

provision of variables

satisfying monoid laws

- Associativity = substitution lemma

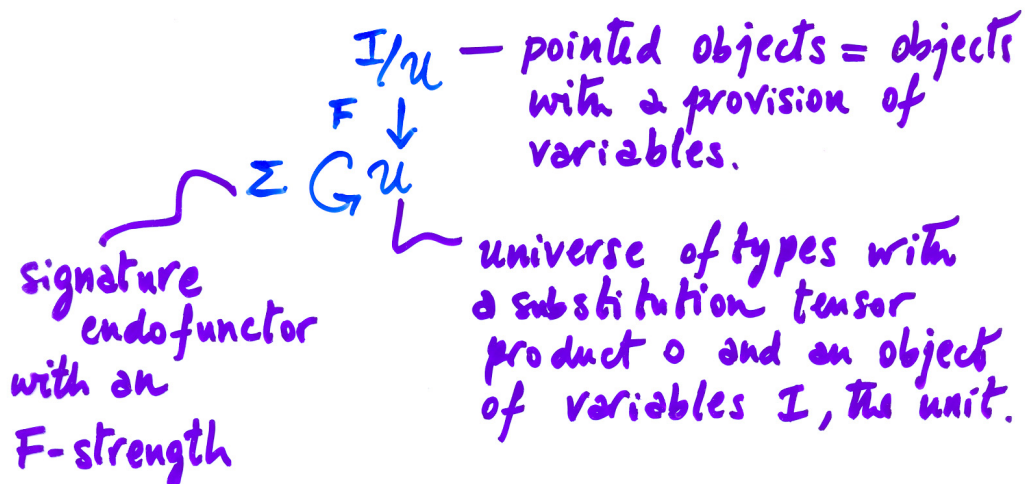
$$\begin{array}{ccc} (x \langle y_i \rangle_i) \langle z_j \rangle_j & \leftrightarrow & x \langle y_i \langle z_j \rangle_j \rangle_i \\ \downarrow & & \downarrow \\ (x[y_i]_i) \langle z_j \rangle_j & & x \langle y_i[z_j]_j \rangle_i \\ \downarrow & & \downarrow \\ (x[y_i]_i)[z_j]_j & = & x[y_i[z_j]_j]_i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

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

- $x[v_i]_i = x$

## A GENERAL ABSTRACT VIEW

- Mathematical universe



$$\Sigma(x) \circ y \xrightarrow{\text{st}_x, \downarrow_y} \Sigma(x \circ y)$$

- Mathematical models:  $\Sigma$ -monoids (substitution algebras)

$$\begin{array}{c} \Sigma x \\ \downarrow x \\ x \circ x \xrightarrow{s} x \xleftarrow{v} I \end{array} \} \text{ a monoid}$$

$$\begin{array}{ccccc} \Sigma(x) \circ x & \xrightarrow{\text{st}_x, v} & \Sigma(x \circ x) & \xrightarrow{\Sigma s} & \Sigma x \\ \downarrow x_{\text{id}} & & & & \downarrow x \\ x \circ x & \xrightarrow{s} & & & x \end{array}$$

## THEOREM

The free monoid

$$\underline{\text{Mon}}(\mathcal{U}) \xleftarrow{\gamma} \mathcal{U}$$

construction generalises to

$$\Sigma\text{-}\underline{\text{Mon}}(\mathcal{U}) \xleftarrow{\gamma} \mathcal{U}$$

$M_0 = \mu z. I + \Sigma z$  is the initial  $\Sigma$ -monoid

where

$$M(x) = \mu z. I + x \circ z + \Sigma z$$

with monoid structure uniquely determined by

$$\begin{array}{ccc} I \circ Mx & & \\ \text{eoid} \downarrow & \searrow \cong & \\ Mx \circ Mx & \xrightarrow{s} & Mx \end{array}$$

$$\begin{array}{ccc} (x \circ Mx) \circ Mx \cong x \circ (Mx \circ Mx) & \xrightarrow{\text{id} \circ s} & x \circ Mx \\ \text{aoid} \downarrow & & \downarrow a \\ Mx \circ Mx & \xrightarrow{s} & Mx \end{array}$$

$$\begin{array}{ccccc} \Sigma(Mx) \circ Mx & \xrightarrow{\text{st}_{Mx, e}} & \Sigma(Mx \circ Mx) & \xrightarrow{\Sigma s} & \Sigma Mx \\ \text{toid} \downarrow & & & & \downarrow t \\ Mx \circ Mx & \xrightarrow{s} & & & Mx \end{array}$$

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

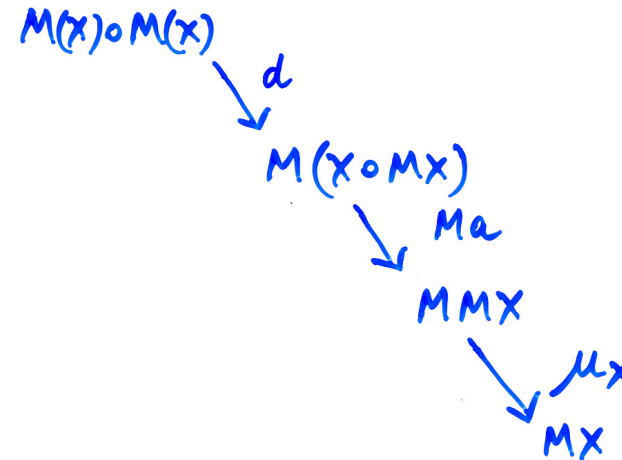
= app( subst t1 S , subst t2 S )

| 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

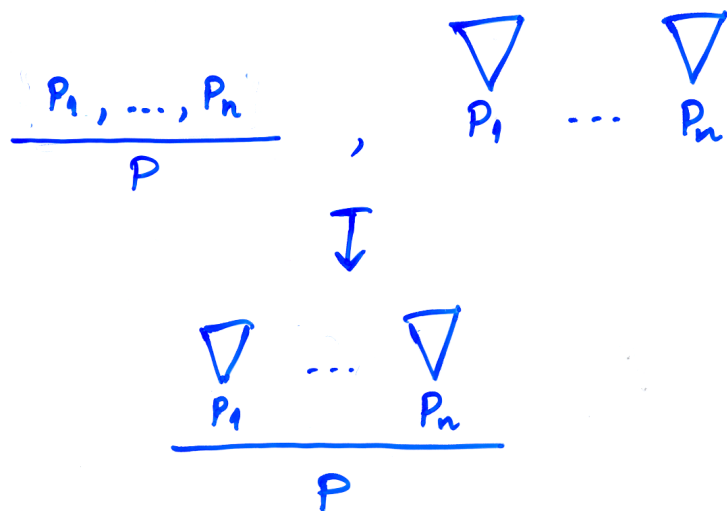
#### Grammars

$$w_0 x_1 w_1 \dots w_{n-1} x_n w_n, \quad x_i \rightarrow w_i (i=1..n)$$



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

#### Proof trees



## MATHEMATICAL MODELS

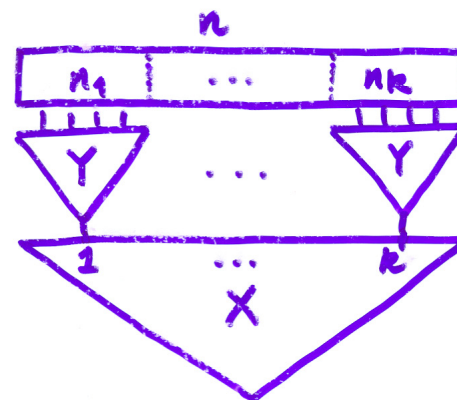
### ~ Linear Species ~ [Joyal]

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

- Model:  $\text{Set}^{\mathbb{N}}$ ,  $\mathbb{N}$  = the set of natural numbers

Substitution Tensor product:

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



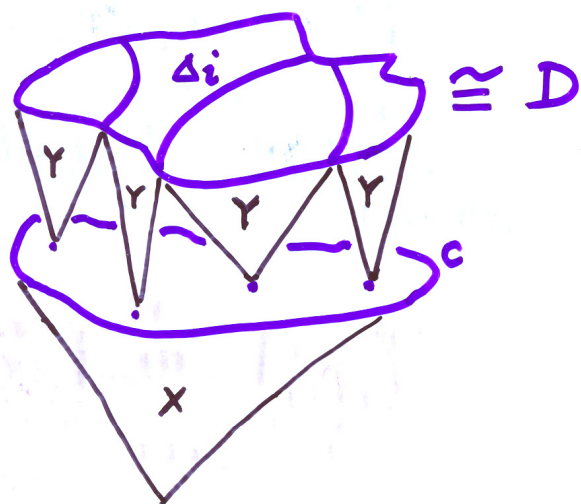
- Monoids = [planar] operads.



## MATHEMATICAL MODELS ~ SPECIES ~ [Joyal]

- A "context" is a set of tokens.
- Model:  $\text{Set}^{\mathbb{B}}$ ,  $\mathbb{B}$  = the category of finite sets and bijections

Substitution tensor product:



$$(X \circ Y)(D) = \int^{C \in \mathbb{B}} X(C) \times \int^{\Delta \in \mathbb{B}^C} \prod_{i \in C} Y(\Delta_i) \times \mathbb{B}(\biguplus_i \Delta_i, D)$$

- Monoids = [symmetric] operads

## MANY-SORTED MODELS

- Many-sorted contexts

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

set of sorts
category of untyped contexts

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

- Model:  $(\text{Set}^{\mathbb{F}[S]})^S$   $X_\sigma(C)$  =  $\sigma$ -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:

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

## 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.
 
$$\frac{\text{ToT} \rightarrow T}{\frac{T_\sigma(C) \times \prod_{(x:z) \in C} T_z(D) \longrightarrow T_\sigma(D)}{\Rightarrow \text{the cut rule}}}$$

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

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:
  - $x, y, z : 0, f : A(x, y), g : A(y, z)$ 

$$\vdash \text{comp}(x, y, z, f, g) : A(x, z)$$
  - $x : 0 \vdash \text{id}(x) : A(x, x)$
  - $x, y : 0, f, g, h : A(x, y),$ 

$$\alpha : C(x, y, f, g), \beta : C(x, y, g, h)$$

$$\vdash \text{vcomp}(x, y, f, g, h, \alpha, \beta) : C(x, y, f, h)$$
  - ...
- Equations:
  - ...

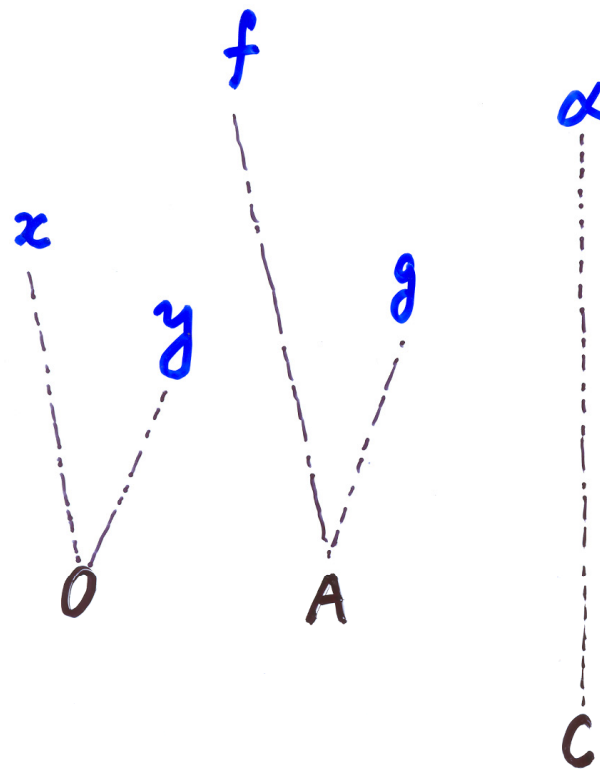
## SYNTAX IS PERVASIVE ... AND PERVERSE!

[!?] What is a dependently-sorted context!?

Example: A context for  $x \xrightarrow{f} y$  with a dependent arrow  $g$ .

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

## GRAPHICAL REPRESENTATION



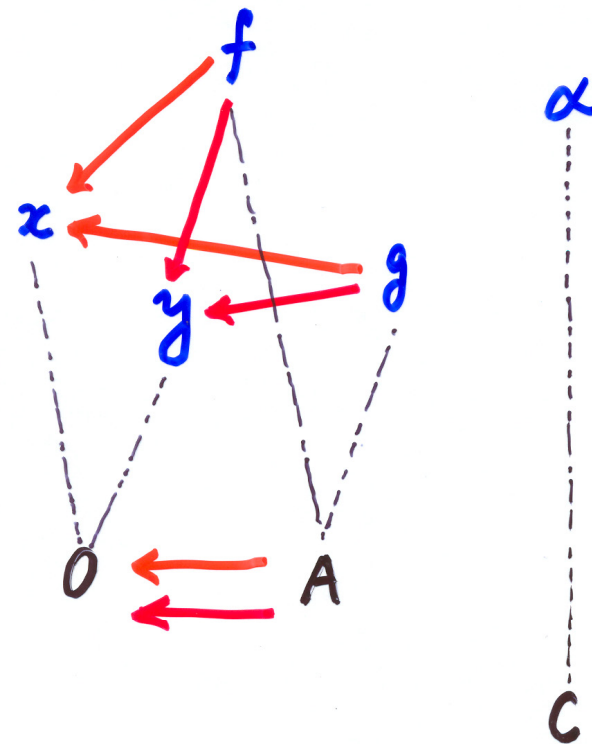
## SYNTAX IS PERVASIVE ... AND PERVERSE!

[!?] What is a dependently-sorted context!?

Example: A context for  $x \xrightarrow{f} y$ .

$x, y : 0, f, g : A(x, y), \alpha : C(x, y, f, g)$

## GRAPHICAL REPRESENTATION

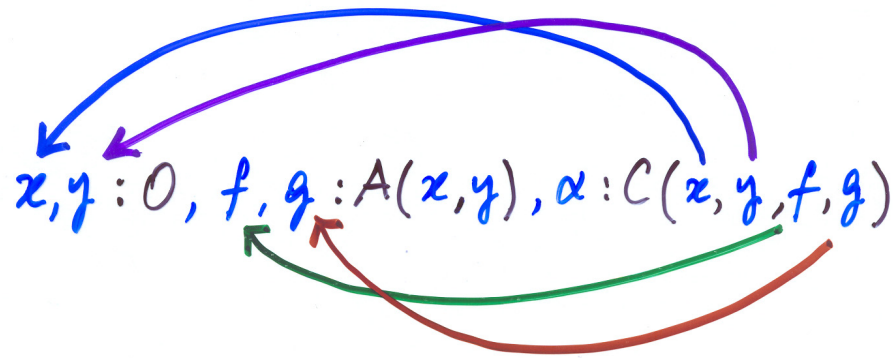


$x, y : 0 \vdash A(x, y) : *$

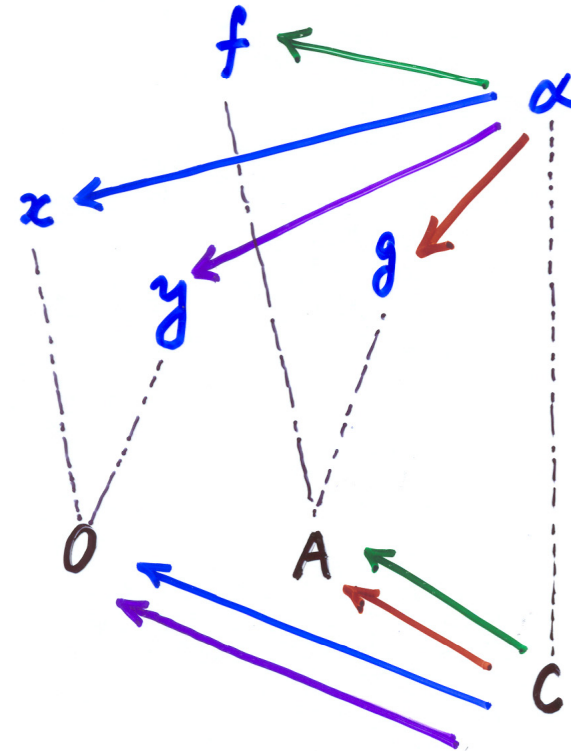
## SYNTAX IS PERVASIVE ... AND PERVERSE!

[!?] What is a dependently-sorted context!?

Example: A context for  $x \xrightarrow{f} \alpha \xrightarrow{g} y$ .



## GRAPHICAL REPRESENTATION



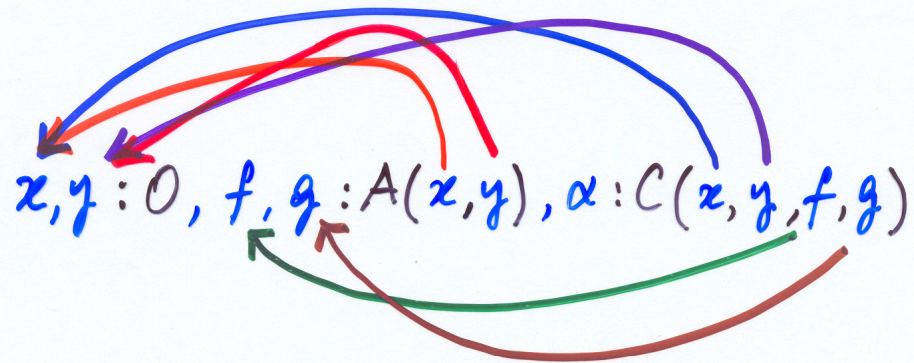
$$x, y : 0, f, g : A(x, y) \vdash C(x, y, f, g) : *$$



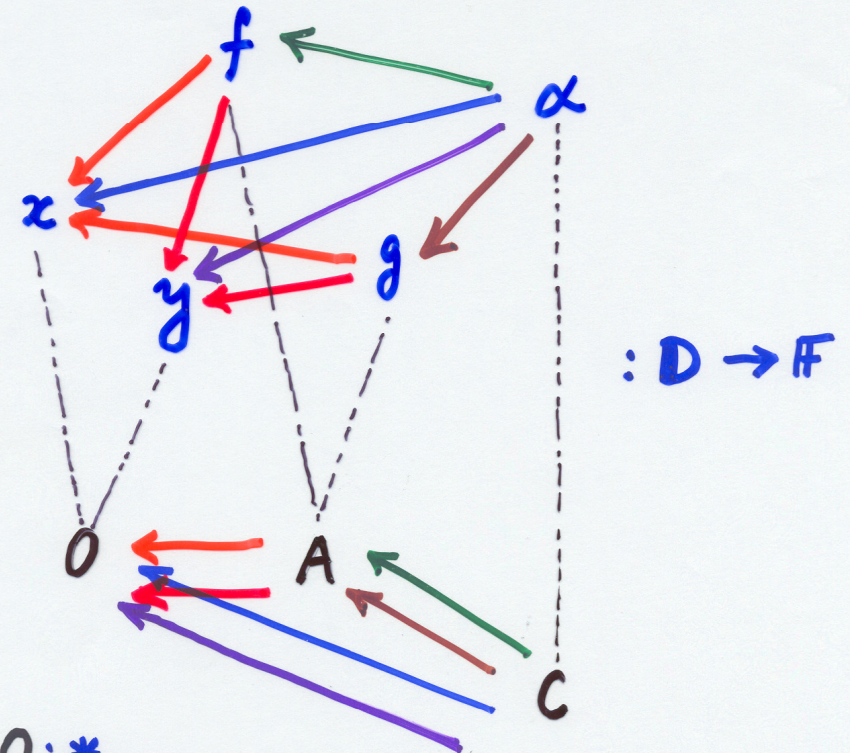
## SYNTAX IS PERVASIVE ... AND PERVERSE!

[!?] What is a dependently-sorted context!?

Example: A context for  $x \xrightarrow{f} \alpha \xrightarrow{g} y$ .



## GRAPHICAL REPRESENTATION



$\vdash 0 : *$

$x, y : 0 \vdash A(x, y) : *$

$x, y : 0, f, g : A(x, y) \vdash C(x, y, f, g) : *$

$\hookrightarrow$  Category of dependent sorts:

$$D = [0 \Leftarrow A \Leftarrow C]$$



## MATHEMATICAL MODEL

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

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

- Model:

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

$$x \in X_{\sigma}(c) \iff c \vdash x : \sigma(\dots x_i \dots)$$

where  $x_i = X_{d_i}(c)(x)$   
for  $d_i : \sigma \rightarrow \sigma_i$  in  $\mathbb{D}$

- Substitution tensor product:

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

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

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

...

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

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

## MIXED MODELS

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

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

intuitionistic (cartesian) context      linear (symmetric monoidal) context

- Cut rule:

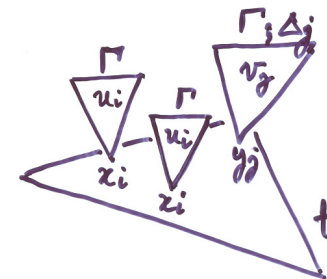
$$x_1 : \sigma_1, \dots, x_m : \sigma_m; y_1 : \tau_1, \dots, y_n : \tau_n \vdash t : \alpha$$

$$\Gamma; \vdash u_i : \sigma_i \quad (1 \leq i \leq m)$$

$$\Gamma; \Delta_j \vdash v_j : \tau_j \quad (1 \leq j \leq n)$$

---

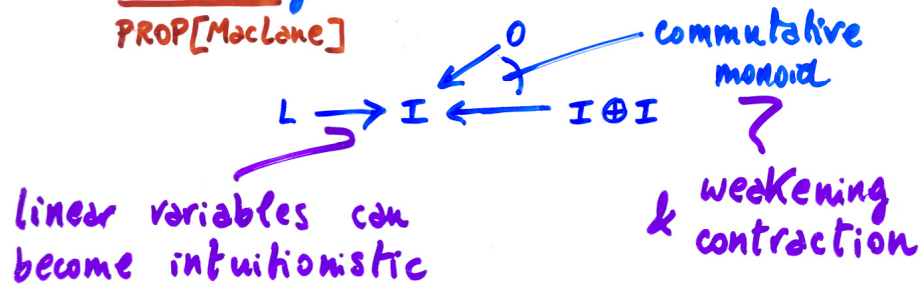

$$\Gamma; \Delta_1, \dots, \Delta_n \vdash t[u_i/x_i, v_j/y_j] : \alpha$$



new feature  
— absent in  
mathematical  
examples —

## MATHEMATICAL MODEL

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



Type theoretically:

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

That is,

$$\frac{B \rightarrow F \xleftarrow{0 \text{ c.m.}} F \oplus F \text{ in } \mathcal{B} \text{ s.m.}}{\mathcal{M} \xrightarrow{\text{s.m.}} \mathcal{B}}$$

- Explicit description

$$\begin{array}{c} C \xrightarrow{p} D \\ \Gamma \searrow \leq \swarrow \Delta \\ (L \leq I) \end{array} \left( \begin{array}{l} \text{int. vars remain int.} \\ = \forall x \in C. \\ (x:I) \in \Gamma \Rightarrow (px:I) \in \Delta \end{array} \right)$$

s.t.  $\forall (y:L) \in \Delta. \exists! (x:L) \in \Gamma. p(x) = y$   
 (lin. vars are lin.)

## CONTEXT INDEXING

$$\mathcal{M} \xleftrightarrow{\quad} \mathcal{F} \xrightarrow{\quad} \mathcal{F} \times \mathcal{F} \text{ in } \underline{\text{Cat}}$$

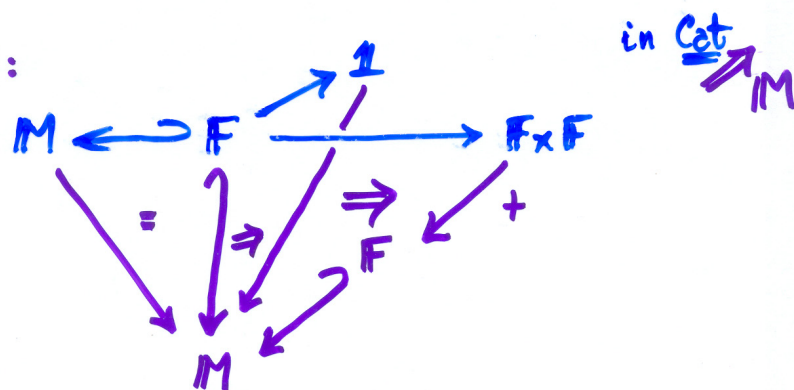
induces

$$\mathcal{M} : \mathcal{M}^{\text{op}} \rightarrow \text{Cat}$$

$$\langle \dots, L, \dots, I, \dots \rangle \mapsto \dots \times \mathcal{M} \times \dots \times \mathcal{F} \times \dots$$

## CONTEXT INDEXING

In fact:



induces

$$\mathcal{M}: \mathcal{M}^{\text{op}} \rightarrow \text{Cat}_{\mathcal{M}}$$

$$\langle \dots, L, \dots, I, \dots \rangle \mapsto \dots \times \mathcal{M} \times \dots \times \mathcal{F} \times \dots$$

$$\downarrow \oplus$$

$$\mathcal{M}$$

## MIXED-SUBSTITUTION TENSOR PRODUCT

● Model:  $\underline{\text{Set}}^{\mathcal{M}}$

Substitution tensor product:

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

$\mathcal{M}(c)$   
 $\bigoplus_c \downarrow$   
 $\mathcal{M}$   
 new feature — absent in mathematical examples —

● Monoids = MIXED OPERADS  $\rightarrow$  Generalise and combine Lawvere theories and [symmetric] operads.

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





## 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