Substructural Abstract Syntax with Variable Binding and Single-Variable Substitution

Sanjiv Ranchod

(joint work with Marcelo Fiore)

University of Cambridge

LICS 2025, Singapore

Introduction

Focus:

Substructural Syntax with Variable Binding

Substitution

Introduction

Focus:

Substructural Syntax with Variable Binding
Substitution

Substructural Syntax:

Cartesian $\rightsquigarrow$ weakening + contraction + exchange

Introduction

Focus:

Substructural Syntax with Variable Binding
Substitution

Substructural Syntax:

Cartesian $\rightsquigarrow$ weakening + contraction + exchange
Linear $\rightsquigarrow$ exchange
Affine $\rightsquigarrow$ weakening + exchange
Relevant $\rightsquigarrow$ contraction + exchange

Introduction

Focus:

Substructural Syntax with Variable Binding
Substitution

Substructural Syntax:

Cartesian $\rightsquigarrow$ weakening + contraction + exchange
Linear $\rightsquigarrow$ exchange
Affine $\rightsquigarrow$ weakening + exchange
Relevant $\rightsquigarrow$ contraction + exchange

Variable Binding:

Signatures involve operations which may bind variables

Introduction

Focus:

Substructural Syntax with Variable Binding
Substitution

Substructural Syntax:

Cartesian $\rightsquigarrow$ weakening + contraction + exchange
Linear $\rightsquigarrow$ exchange
Affine $\rightsquigarrow$ weakening + exchange
Relevant $\rightsquigarrow$ contraction + exchange

Variable Binding:

Signatures involve operations which may bind variables

Substitution:

Simultaneous substitution
Capture-avoiding single-variable substitution

Introduction

Simultaneous Substitution:

Cartesian: Fiore-Plotkin-Turi (1999)

Introduction

Simultaneous Substitution:

Cartesian: Fiore-Plotkin-Turi (1999)
Linear: Tanaka (2005)
Affine: Tanaka-Power (2006)

Introduction

Simultaneous Substitution:

Cartesian: Fiore-Plotkin-Turi (1999)
Linear: Tanaka (2005)
Affine: Tanaka-Power (2006)

Single-Variable Substitution:

Cartesian: Fiore-Plotkin-Turi (1999)
Others: Open

Setting

Category Theoretic "Presheaf Model"

Contexts: (Universal) monoidal category generated by structural rules

Syntax: Covariant presheaves over contexts

$P(\Gamma) = \{\text{terms for syntax } P \text{ in context } \Gamma\}$

Setting

Category Theoretic "Presheaf Model"

Contexts: (Universal) monoidal category generated by structural rules

Syntax: Covariant presheaves over contexts

$P(\Gamma) = \{\text{terms for syntax } P \text{ in context } \Gamma\}$

eg. Linear Setting

Symmetric object: $(A, s : A \otimes A \to A \otimes A)$
$s \rightsquigarrow$ exchange

Contexts : $\mathbb{B}$ = free monoidal category over symmetric object

Syntax : $\mathcal{B} = \mathbf{Set}^{\mathbb{B}}$ = combinatorial species

Setting

Category Theoretic "Presheaf Model"

Contexts: (Universal) monoidal category generated by structural rules

Syntax: Covariant presheaves over contexts

$P(\Gamma) = \{\text{terms for syntax } P \text{ in context } \Gamma\}$

eg. Linear Setting

Symmetric object: $(A, s : A \otimes A \to A \otimes A)$
$s \rightsquigarrow$ exchange

Contexts : $\mathbb{B}$ = free monoidal category over symmetric object

Syntax : $\mathcal{B} = \mathbf{Set}^{\mathbb{B}}$ = combinatorial species

Context Extension: $\delta \rightsquigarrow$ endofunctor on presheaves

$$\delta(P)(\Gamma) = P(\Gamma + 1)$$

Axiomatisation of Substitution

Data:

Syntax: P

Substitution: $\sigma : \delta(P) \otimes P \to P$

Variables: $\nu : 1 \to \delta(P)$

Axiomatisation of Substitution

Data:

Syntax: P

Substitution: $\sigma : \delta(P) \otimes P \to P$

Variables: $\nu : 1 \to \delta(P)$

Axioms:

Account for structure of contexts $\rightsquigarrow$ different for each case

Finite equational presentation

Axiomatisation of Substitution

Data:

Syntax: P

Substitution: $\sigma : \delta(P) \otimes P \to P$

Variables: $\nu : 1 \to \delta(P)$

Axioms:

Account for structure of contexts $\rightsquigarrow$ different for each case

Finite equational presentation

eg. Linear Axioms:

Two Unitor Laws $\rightsquigarrow$ Behaviour of Variables

Two Operad Laws $\rightsquigarrow$ Successive Substitutions

Axiomatisation of Substitution

Data:

Syntax: P

Substitution: $\sigma : \delta(P) \otimes P \to P$

Variables: $\nu : 1 \to \delta(P)$

Axioms:

Account for structure of contexts $\rightsquigarrow$ different for each case

Finite equational presentation

eg. Linear Axioms:

- Two Unitor Laws $\rightsquigarrow$ Behaviour of Variables
- Two Operad Laws $\rightsquigarrow$ Successive Substitutions
  - equivalently: Extended Substitution Lemma

Abstract Syntax for a Binding Signature

Binding Signature: $\Sigma \rightsquigarrow$ Endofunctor on presheaves

δ describes binding

Abstract Syntax for a Binding Signature

Binding Signature: $\Sigma \rightsquigarrow$ Endofunctor on presheaves

δ describes binding

Variables: $V \rightsquigarrow$ Presheaf, $V = \mathcal{Y}(1)$

Abstract Syntax for a Binding Signature

Binding Signature: $\Sigma \rightsquigarrow$ Endofunctor on presheaves

δ describes binding

Variables: $V \rightsquigarrow$ Presheaf, $V = \mathcal{Y}(1)$

Abstract Syntax:

$TV \rightsquigarrow$ Free Σ-algebra on V

Fixed point: $TV = \mu X.V + \Sigma(X)$

Representation Independent

Abstract Syntax for a Binding Signature

Binding Signature: $\Sigma \rightsquigarrow$ Endofunctor on presheaves

δ describes binding

Variables: $V \rightsquigarrow$ Presheaf, $V = \mathcal{Y}(1)$

Abstract Syntax:

$TV \rightsquigarrow$ Free Σ-algebra on V

Fixed point: $TV = \mu X.V + \Sigma(X)$

Representation Independent

eg. Linear Lambda Calculus

$\Sigma_\lambda(X) = X \otimes X + \delta(X) \rightsquigarrow$ application and abstraction

Abstract Syntax : $\Lambda = \mu X.V + X \otimes X + \delta(X)$

Substitution for Abstract Syntax

Main Theorem:

The abstract syntax is equipped with structural recursively defined and universal substitution structure.

Substitution for Abstract Syntax

Main Theorem:

The abstract syntax is equipped with structural recursively defined and universal substitution structure.

Problem:

Substitution is constructed on $\delta(TV)$, which is not a priori defined inductively.

Substitution for Abstract Syntax

Main Theorem:

The abstract syntax is equipped with structural recursively defined and universal substitution structure.

Problem:

Substitution is constructed on $\delta(TV)$, which is not a priori defined inductively.

Cartesian Solution: Fiore-Plotkin-Turi (1999)

Approach for Other Cases:

Show $\delta(TV)$ admits structural recursion by being an initial algebra.

Uniformity Property and Leibniz Isomorphism

Uniformity Property

Given the following situation:

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{F} & \mathcal{C} \\ {\scriptstyle H}\downarrow & \cong & \downarrow{\scriptstyle H} \\ \mathcal{D} & \xrightarrow[G]{} & \mathcal{D} \end{array}$$

H is a left adjoint and μF exists $\implies H(\mu F) \cong \mu G$

Uniformity Property and Leibniz Isomorphism

Uniformity Property

Given the following situation:

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{F} & \mathcal{C} \\ {\scriptstyle H}\downarrow & \cong & \downarrow{\scriptstyle H} \\ \mathcal{D} & \xrightarrow[G]{} & \mathcal{D} \end{array}$$

H is a left adjoint and μF exists $\implies H(\mu F) \cong \mu G$

Leibniz Isomorphism

How to "simplify" $\delta(X \otimes Y)$

Different for each settings

eg. Linear Setting: $\mathcal{L} : \delta(X \otimes Y) \cong \delta(X) \otimes Y + X \otimes \delta(Y)$

Linear Lambda Calculus

$$
\begin{array}{ccc}
\mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\
\left\downarrow{\scriptstyle \langle \mathrm{Id},\delta \rangle}\right. & & \left\downarrow{\scriptstyle \langle \mathrm{Id},\delta \rangle}\right. \\
\mathcal{B} \times \mathcal{B} & \overset{G}{\dashrightarrow} & \mathcal{B} \times \mathcal{B}
\end{array}
$$

Linear Lambda Calculus

X

$$
\begin{array}{ccc}
\mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\
{\scriptstyle \langle \mathrm{Id},\delta\rangle}\downarrow & & \downarrow{\scriptstyle \langle \mathrm{Id},\delta\rangle} \\
\mathcal{B}\times\mathcal{B} & \overset{G}{\dashrightarrow} & \mathcal{B}\times\mathcal{B}
\end{array}
$$

Linear Lambda Calculus

$$X \longmapsto \langle X, \delta(X) \rangle$$

$$\begin{array}{ccc} \mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\ {\scriptstyle \langle \mathrm{Id}, \delta \rangle}\downarrow & & \downarrow{\scriptstyle \langle \mathrm{Id}, \delta \rangle} \\ \mathcal{B} \times \mathcal{B} & \overset{G}{\dashrightarrow} & \mathcal{B} \times \mathcal{B} \end{array}$$

Linear Lambda Calculus

$$
\begin{array}{ccc}
X & \longmapsto & V + X \otimes X + \delta(X) \\
\big\downarrow & & \\
\langle X, \delta(X) \rangle & &
\end{array}
$$

$$
\begin{array}{ccc}
\mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\
{\scriptstyle \langle \mathrm{Id}, \delta \rangle}\big\downarrow & & \big\downarrow{\scriptstyle \langle \mathrm{Id}, \delta \rangle} \\
\mathcal{B} \times \mathcal{B} & \overset{G}{\dashrightarrow} & \mathcal{B} \times \mathcal{B}
\end{array}
$$

Linear Lambda Calculus

$$
\begin{array}{ccc}
X & \longmapsto & V + X \otimes X + \delta(X) \\
\big\downarrow & & \big\downarrow \\
\langle X, \delta(X) \rangle & & \langle V+X\otimes X+\delta(X),\ \delta(V+X\otimes X+\delta(X)) \rangle
\end{array}
$$

$$
\begin{array}{ccc}
\mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\
\big\downarrow{\scriptstyle \langle \mathrm{Id}, \delta \rangle} & & \big\downarrow{\scriptstyle \langle \mathrm{Id}, \delta \rangle} \\
\mathcal{B} \times \mathcal{B} & \xrightarrow{G} & \mathcal{B} \times \mathcal{B}
\end{array}
$$

Linear Lambda Calculus

$$
\begin{array}{ccc}
X & \longmapsto & V + X \otimes X + \delta(X) \\
\downarrow & & \downarrow \\
 & & \langle V+X\otimes X+\delta(X), \; \delta(V+X\otimes X+\delta(X))\rangle \\
 & & \downarrow \mathcal{L} \; \cong \\
\langle X, \delta(X)\rangle & & \langle V+X\otimes X+\delta(X), \; \delta(V)+\delta(X)\otimes X+X\otimes\delta(X)+\delta\delta(X)\rangle
\end{array}
$$

$$
\begin{array}{ccc}
\mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\
\downarrow \langle \mathrm{Id}, \delta \rangle & & \downarrow \langle \mathrm{Id}, \delta \rangle \\
\mathcal{B} \times \mathcal{B} & \xrightarrow{G} & \mathcal{B} \times \mathcal{B}
\end{array}
$$

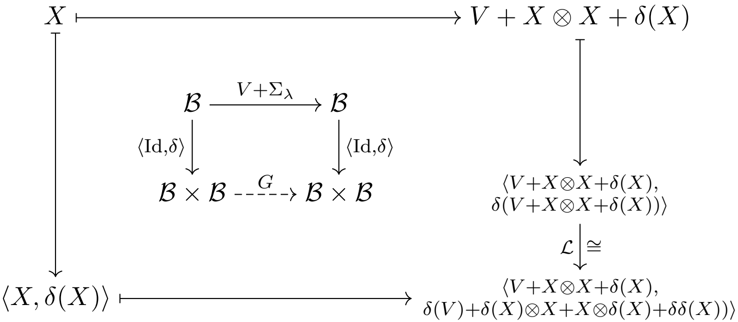
Linear Lambda Calculus

$$\begin{array}{ccc} X & \longmapsto & V + X \otimes X + \delta(X) \\ \big\downarrow & & \big\downarrow \\ & & \langle V+X\otimes X+\delta(X), \\ & & \delta(V+X\otimes X+\delta(X))\rangle \\ & & \mathcal{L}\big\downarrow\cong \\ \langle X, \delta(X)\rangle & \longmapsto & \langle V+X\otimes X+\delta(X), \\ & & \delta(V)+\delta(X)\otimes X+X\otimes\delta(X)+\delta\delta(X))\rangle \end{array}$$

$$\begin{array}{ccc} \mathcal{B} & \xrightarrow{V+\Sigma_\lambda} & \mathcal{B} \\ {\scriptstyle \langle \mathrm{Id},\delta\rangle}\big\downarrow & & \big\downarrow{\scriptstyle \langle \mathrm{Id},\delta\rangle} \\ \mathcal{B}\times\mathcal{B} & \overset{G}{\dashrightarrow} & \mathcal{B}\times\mathcal{B} \end{array}$$

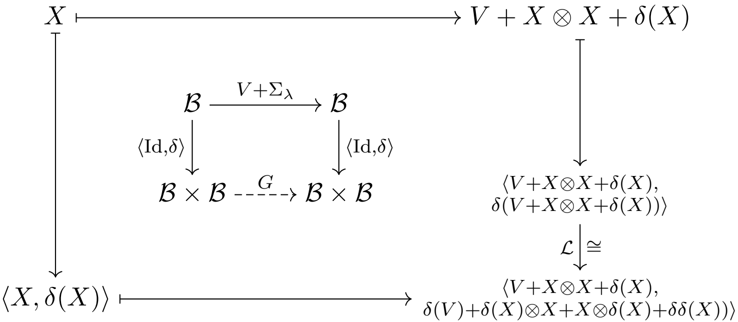
$$G(X, Y) = \langle V + X \otimes X + Y, \delta(V) + Y \otimes X + X \otimes Y + \delta(Y)\rangle$$

Linear Lambda Calculus



$$G(X,Y) = \langle V + X \otimes X + Y, \delta(V) + \underbrace{Y \otimes X + X \otimes Y + \delta(Y)}_{\text{Derived Functor: } \Sigma_\lambda^\dagger(X,Y)} \rangle$$

Linear Lambda Calculus



$$G(X,Y) = \langle V + X \otimes X + Y, \delta(V) + \underbrace{Y \otimes X + X \otimes Y + \delta(Y)}_{\text{Derived Functor: } \Sigma^\dagger_\lambda(X,Y)} \rangle$$

Uniformity Property: $\langle \Lambda, \delta(\Lambda) \rangle$ is the fixed point of

$$\begin{cases} X = V + X \otimes X + Y \\ Y = \delta(V) + X \otimes Y + Y \otimes X + \delta(Y) \end{cases}$$

Generalised Structural Recursion

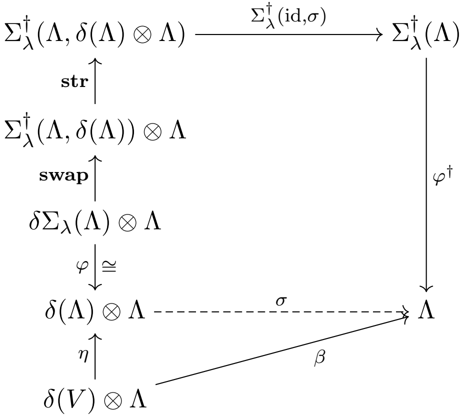
Bird-Paterson (1999): Generalised Structural Recursion

Λ initial $\rightsquigarrow$ admits iterator

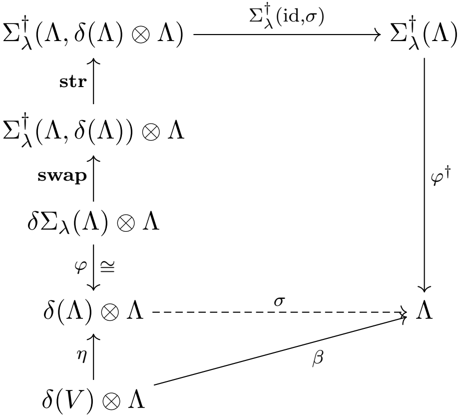
$\implies$ $\delta(\Lambda)$ admits generalised iterator $\rightsquigarrow$ corresponds to initiality conditions

Matthes-Uustalu (2003): Useful special case

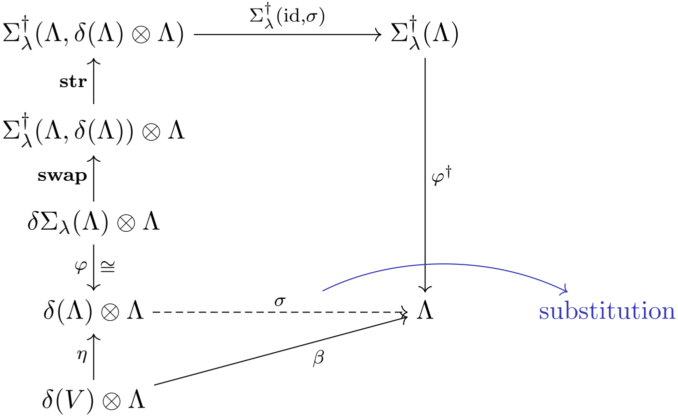
Substitution for Abstract Syntax



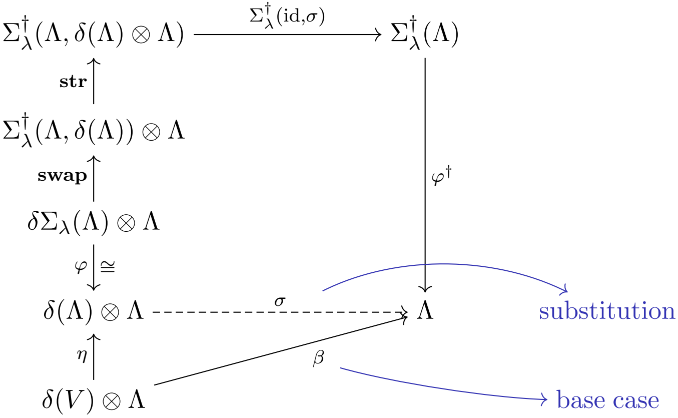
Substitution for Abstract Syntax



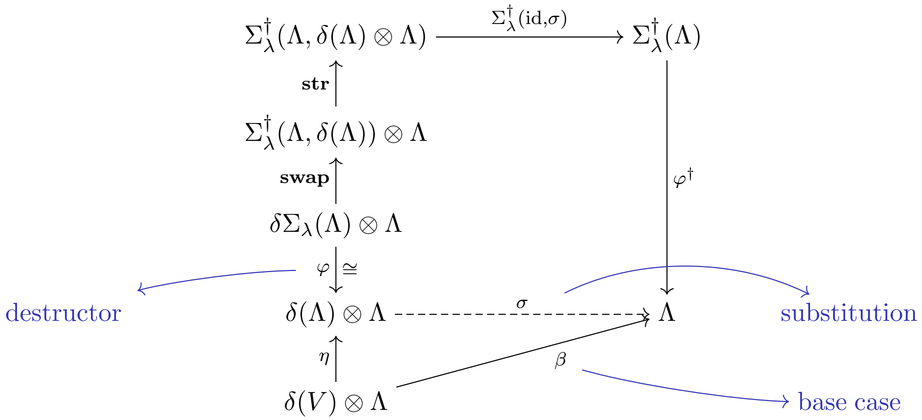
Substitution for Abstract Syntax



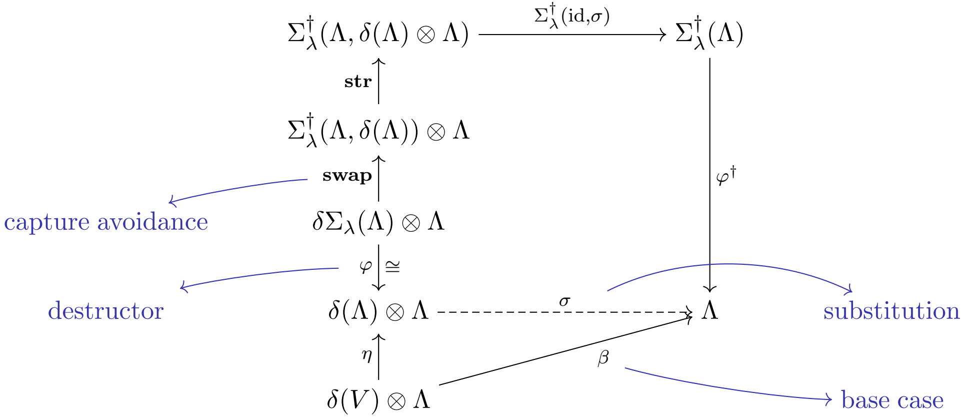
Substitution for Abstract Syntax



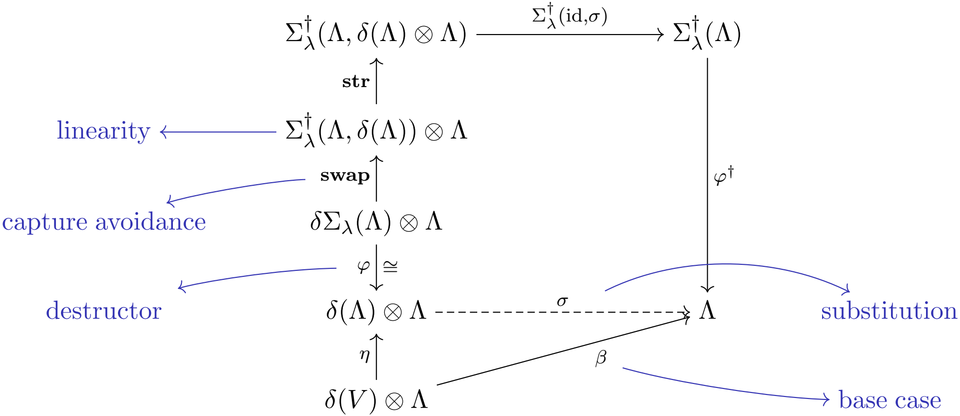
Substitution for Abstract Syntax



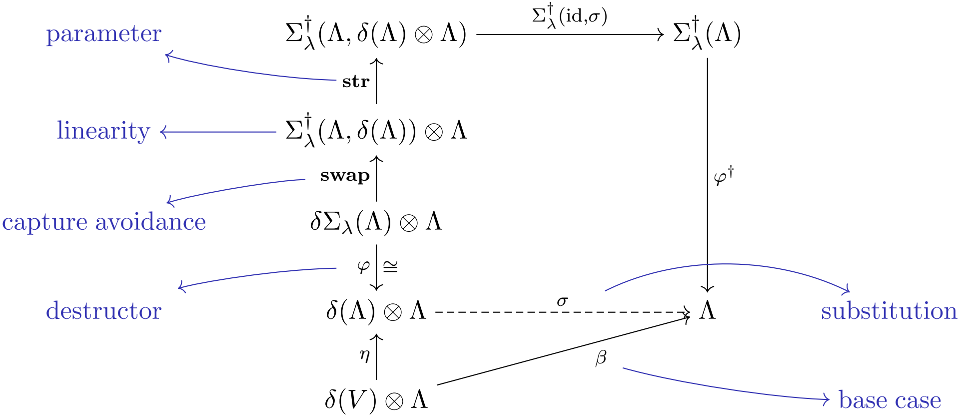
Substitution for Abstract Syntax



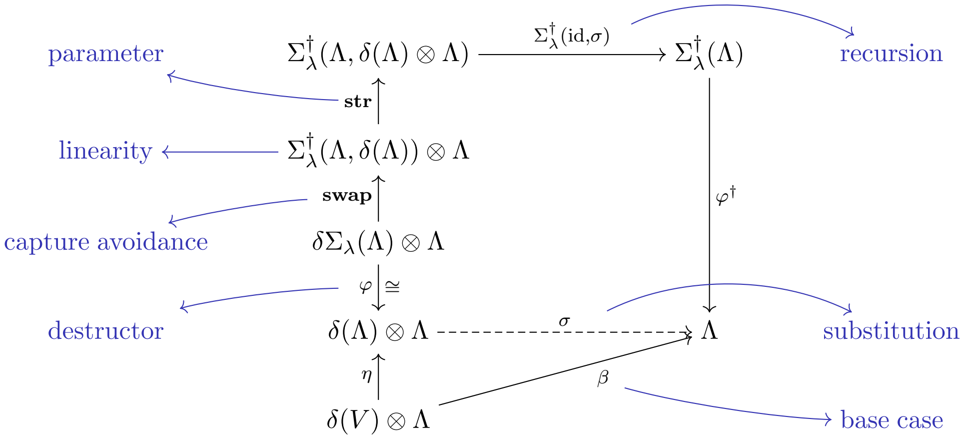
Substitution for Abstract Syntax



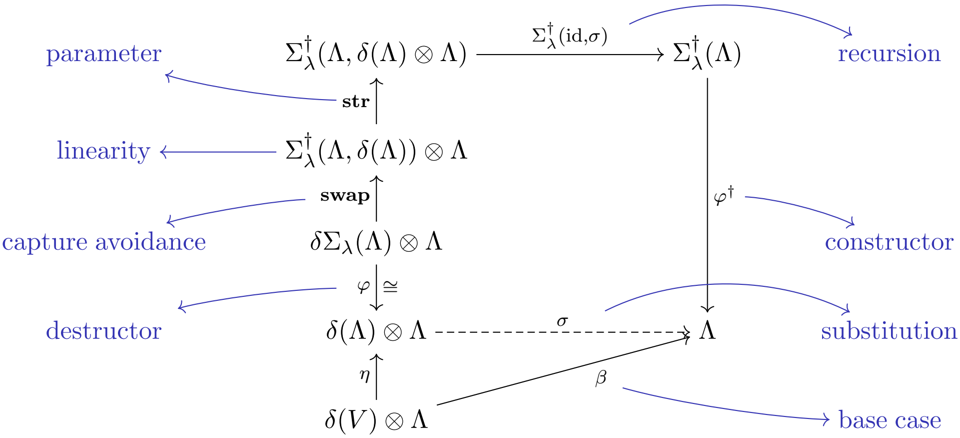
Substitution for Abstract Syntax



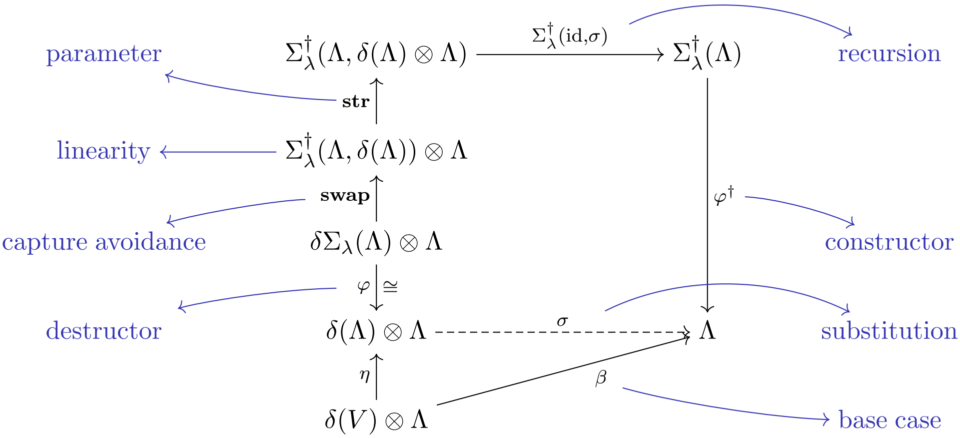
Substitution for Abstract Syntax



Substitution for Abstract Syntax



Substitution for Abstract Syntax



Thm: Λ is the initial Σ_λ-algebra with compatible substitution structure.

Summary

1. Axiomatise Substitution

Summary

1. Axiomatise Substitution

2. Model Binding Signature as Endofunctor Σ

Summary

1. Axiomatise Substitution
2. Model Binding Signature as Endofunctor Σ
3. Model Abstract Syntax as Free Algebra TV

Summary

1. Axiomatise Substitution
2. Model Binding Signature as Endofunctor Σ
3. Model Abstract Syntax as Free Algebra TV
4. Uniformity Transfers Universal Property to $\delta(TV)$

Summary

1. Axiomatise Substitution
2. Model Binding Signature as Endofunctor Σ
3. Model Abstract Syntax as Free Algebra TV
4. Uniformity Transfers Universal Property to $\delta(TV)$
5. Induce Substitution Structure on Abstract Syntax

Summary

1. Axiomatise Substitution
2. Model Binding Signature as Endofunctor Σ
3. Model Abstract Syntax as Free Algebra TV
4. Uniformity Transfers Universal Property to $\delta(TV)$
5. Induce Substitution Structure on Abstract Syntax
6. Show Induced Substitution Structure is Universal

Summary

1. Axiomatise Substitution
2. Model Binding Signature as Endofunctor Σ
3. Model Abstract Syntax as Free Algebra TV
4. Uniformity Transfers Universal Property to $\delta(TV)$
5. Induce Substitution Structure on Abstract Syntax
6. Show Induced Substitution Structure is Universal
7. Extract Program for Substitution

Future Work

Second-Order Theories for Linear, Affine and Relevant Settings

Future Work

Second-Order Theories for Linear, Affine and Relevant Settings

Single-Variable Substitution for Combined Settings

eg. Linear-Cartesian Setting