Towards an algebraic classification of recognizable sets of lambda-terms[1]

Sylvain Salvati

INRIA Bordeaux sud-ouest, LaBRI, université de Bordeaux

Automata, Concurrency and Timed Systems (ACTS) III

[1]With the financial support of ANR 2010 BLAN 0202 01 FREC

Outline

Recognizable sets of λ-terms

Recognizability and congruence

Eilenberg theorem: towards an algebraic classification of classes of C-recognizable languages

Varieties of locally finite CCCs

Outline

Recognizable sets of λ-terms

Recognizability and congruence

Eilenberg theorem: towards an algebraic classification of classes of C-recognizable languages

Varieties of locally finite CCCs

λ-calculus: syntax

Given a finite set of atomic types $\mathcal{A}$, simple types are:

$$\mathcal{T}_{\mathcal{A}} := \mathcal{A} | (\mathcal{T}_{\mathcal{A}} \to \mathcal{T}_{\mathcal{A}})$$

λ-calculus: syntax

Given a finite set of atomic types $\mathcal{A}$, simple types are:

$$\mathcal{T}_{\mathcal{A}} := \mathcal{A} | (\mathcal{T}_{\mathcal{A}} \to \mathcal{T}_{\mathcal{A}})$$

A higher order signature (HOS) is a tuple $\Sigma = (\mathcal{A}, \mathcal{C}, \tau)$ where:

- $\mathcal{A}$ is a finite set of atomic types,
- $\mathcal{C}$ is a finite set of constants,
- τ is a function from $\mathcal{C}$ to $\mathcal{T}_{\mathcal{A}}$.

λ-calculus: syntax

Given a finite set of atomic types $\mathcal{A}$, simple types are:

$$\mathcal{T}_{\mathcal{A}} := \mathcal{A} | (\mathcal{T}_{\mathcal{A}} \to \mathcal{T}_{\mathcal{A}})$$

A higher order signature (HOS) is a tuple $\Sigma = (\mathcal{A}, \mathcal{C}, \tau)$ where:

- $\mathcal{A}$ is a finite set of atomic types,
- $\mathcal{C}$ is a finite set of constants,
- τ is a function from $\mathcal{C}$ to $\mathcal{T}_{\mathcal{A}}$.

λ-terms built on Σ are defined as:

- for $\alpha \in \mathcal{T}_{\mathcal{A}}$, $x^{\alpha} \in \Lambda_{\Sigma}^{\alpha}$,
- $c \in \Lambda_{\Sigma}^{\tau(c)}$,
- if $M_1 \in \Lambda_{\Sigma}^{\alpha_2 \to \alpha_1}$, $M_2 \in \Lambda_{\Sigma}^{\alpha_2}$, then $(M_1 M_2) \in \Lambda_{\Sigma}^{\alpha_1}$,
- if $M \in \Lambda_{\Sigma}^{\alpha_1}$, then $\lambda x^{\alpha_2}.M \in \Lambda_{\Sigma}^{\alpha_2 \to \alpha_1}$.

λ-calculus: operational semantics

λ-calculus is a theory of function and computation.
Computation is done with the relation of $\beta\eta$-contraction ($\rightarrow_{\beta\eta}$):

$$\frac{(\lambda x.M)N}{(\lambda x.M)N \rightarrow_{\beta\eta} M[x := N]} \qquad \frac{\lambda x.Mx \quad x \notin FV(M)}{\lambda x.Mx \rightarrow_{\beta\eta} M} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(MM_1) \rightarrow_{\beta\eta} (MM_2)}$$

$$\frac{M_1 \rightarrow_{\beta\eta} M_2}{(M_1 M) \rightarrow_{\beta\eta} (M_2 M)} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(\lambda x.M_1) \rightarrow_{\beta\eta} (\lambda x.M_2)}$$

λ-calculus: operational semantics

λ-calculus is a theory of function and computation.
Computation is done with the relation of $\beta\eta$-contraction ($\rightarrow_{\beta\eta}$):

$$\frac{(\lambda x.M)N}{(\lambda x.M)N \rightarrow_{\beta\eta} M[x := N]} \qquad \frac{\lambda x.Mx \quad x \notin FV(M)}{\lambda x.Mx \rightarrow_{\beta\eta} M} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(MM_1) \rightarrow_{\beta\eta} (MM_2)}$$

$$\frac{M_1 \rightarrow_{\beta\eta} M_2}{(M_1 M) \rightarrow_{\beta\eta} (M_2 M)} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(\lambda x.M_1) \rightarrow_{\beta\eta} (\lambda x.M_2)}$$

$\beta\eta$-reduction ($\xrightarrow{*}_{\beta\eta}$): reflexive transitive closure of $\beta\eta$-contraction
$\beta\eta$-conversion: symetric closure of $\beta\eta$-reduction

λ-calculus: operational semantics

λ-calculus is a theory of function and computation.
Computation is done with the relation of $\beta\eta$-contraction ($\rightarrow_{\beta\eta}$):

$$\frac{(\lambda x.M)N}{(\lambda x.M)N \rightarrow_{\beta\eta} M[x := N]} \qquad \frac{\lambda x.Mx \quad x \notin FV(M)}{\lambda x.Mx \rightarrow_{\beta\eta} M} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(MM_1) \rightarrow_{\beta\eta} (MM_2)}$$

$$\frac{M_1 \rightarrow_{\beta\eta} M_2}{(M_1 M) \rightarrow_{\beta\eta} (M_2 M)} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(\lambda x.M_1) \rightarrow_{\beta\eta} (\lambda x.M_2)}$$

$\beta\eta$-reduction ($\xrightarrow{*}_{\beta\eta}$): reflexive transitive closure of $\beta\eta$-contraction
$\beta\eta$-conversion: symetric closure of $\beta\eta$-reduction

Theorem (Church-Rosser)

$\beta\eta$-conversion is confluent

λ-calculus: operational semantics

λ-calculus is a theory of function and computation.
Computation is done with the relation of $\beta\eta$-contraction ($\rightarrow_{\beta\eta}$):

$$\frac{(\lambda x.M)N}{(\lambda x.M)N \rightarrow_{\beta\eta} M[x := N]} \qquad \frac{\lambda x.Mx \quad x \notin FV(M)}{\lambda x.Mx \rightarrow_{\beta\eta} M} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(MM_1) \rightarrow_{\beta\eta} (MM_2)}$$

$$\frac{M_1 \rightarrow_{\beta\eta} M_2}{(M_1 M) \rightarrow_{\beta\eta} (M_2 M)} \qquad \frac{M_1 \rightarrow_{\beta\eta} M_2}{(\lambda x.M_1) \rightarrow_{\beta\eta} (\lambda x.M_2)}$$

$\beta\eta$-reduction ($\overset{*}{\rightarrow}_{\beta\eta}$): reflexive transitive closure of $\beta\eta$-contraction
$\beta\eta$-conversion: symetric closure of $\beta\eta$-reduction

Theorem (Church-Rosser)

$\beta\eta$-conversion is confluent

Theorem (Strong Normalisation)

Given M in $\Lambda^{\alpha}_{\Sigma}$, there is no infinite sequence of $\beta\eta$-contraction starting in M.

Simply typed λ-calculus generalizes trees

The ranked alphabet $\{e; g; f\}$ where $\mathrm{rank}(e) = 0$, $\mathrm{rank}(g) = 1$, $\mathrm{rank}(f) = 2$ can be represented by the following second order constants:

$$e : o,\ g : o \to o,\ f : o \to o \to o$$

Simply typed λ-calculus generalizes trees

The ranked alphabet $\{e; g; f\}$ where $\mathrm{rank}(e) = 0$, $\mathrm{rank}(g) = 1$, $\mathrm{rank}(f) = 2$ can be represented by the following second order constants:

$$e : o,\ g : o \to o,\ f : o \to o \to o$$

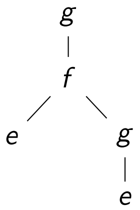
the term $g(f(e, g(e)))$ is represented by the λ-term $g(f\ e\ (g\ e))$

Simply typed λ-calculus generalizes trees

The ranked alphabet $\{e; g; f\}$ where $\mathrm{rank}(e) = 0$, $\mathrm{rank}(g) = 1$, $\mathrm{rank}(f) = 2$ can be represented by the following second order constants:

$$e : o,\ g : o \to o,\ f : o \to o \to o$$

the term $g(f(e, g(e)))$ is represented by the λ-term $g(f\ e\ (g\ e))$
The Böhm tree of the λ-term is the same as the graphic representation of the term:

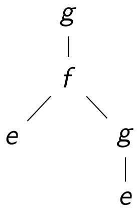


Simply typed λ-calculus generalizes trees

The ranked alphabet $\{e; g; f\}$ where $\mathrm{rank}(e) = 0$, $\mathrm{rank}(g) = 1$, $\mathrm{rank}(f) = 2$ can be represented by the following second order constants:

$$e : o,\ g : o \to o,\ f : o \to o \to o$$

the term $g(f(e, g(e)))$ is represented by the λ-term $g(f\ e\ (g\ e))$
The Böhm tree of the λ-term is the same as the graphic representation of the term:



A λ-term whose normal form represent a tree is a λ-*tree*.

Simply typed λ-calculus generalizes strings

The elements of $\{a; b\}^*$ can be represented with the constants:

$$a : o \to o,\ b : o \to o$$

Strings are represented by terms of type $o \to o$:

the string *aba* is represented by $/aba/ = \lambda x^o.a(b(a\,x^o))$

Simply typed λ-calculus generalizes strings

The elements of $\{a; b\}^*$ can be represented with the constants:

$$a : o \to o, \; b : o \to o$$

Strings are represented by terms of type $o \to o$:

the string *aba* is represented by $/aba/ = \lambda x^o.a(b(a\, x^o))$

Concatenation is then $s_1 + s_2 = \lambda x^o.s_1(s_2(x^o))$:

$$\begin{array}{rcl} /ab/ + /bb/ & = & \lambda x^o.a(b(x^o)) + \lambda x^o.b(b(x^o)) \\ & = & \lambda x^o.(\lambda y^o.a(b\, y^o))((\lambda z^o.b(b\, z^o))x^o) \\ & =_{\beta\eta} & \lambda x^o.a(b(b(b\, z^o))) \end{array}$$

and the empty string is $\lambda x^o.x^o$

Simply typed λ-calculus generalizes strings

The elements of $\{a; b\}^*$ can be represented with the constants:

$$a : o \to o,\ b : o \to o$$

Strings are represented by terms of type $o \to o$:

the string *aba* is represented by $/aba/ = \lambda x^o.a(b(a\,x^o))$

Concatenation is then $s_1 + s_2 = \lambda x^o.s_1(s_2(x^o))$:

$$\begin{aligned}
/ab/ + /bb/ &= \lambda x^o.a(b(x^o)) + \lambda x^o.b(b(x^o)) \\
&= \lambda x^o.(\lambda y^o.a(b\,y^o))((\lambda z^o.b(b\,z^o))x^o) \\
&=_{\beta\eta} \lambda x^o.a(b(b(b\,z^o)))
\end{aligned}$$

and the empty string is $\lambda x^o.x^o$

A λ-term whose normal form represent a string is a λ-*string*.

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^{\alpha})_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^{\alpha}$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^{\alpha}$ to $\mathcal{M}^{\beta}$.

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^\alpha$ to $\mathcal{M}^\beta$.
- ι maps constants of type α to $\mathcal{M}^\alpha$

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^\alpha$ to $\mathcal{M}^\beta$.
- ι maps constants of type α to $\mathcal{M}^\alpha$

A variable assignment $\chi : V \to \bigcup_{\alpha \in \mathcal{T}(\Sigma)} \mathcal{M}^\alpha$ so that $\chi(x^\alpha) \in \mathcal{M}^\alpha$.

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^{\alpha})_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^{\alpha}$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^{\alpha}$ to $\mathcal{M}^{\beta}$.
- ι maps constants of type α to $\mathcal{M}^{\alpha}$

A variable assignment $\chi : V \to \bigcup_{\alpha \in \mathcal{T}(\Sigma)} \mathcal{M}^{\alpha}$ so that $\chi(x^{\alpha}) \in \mathcal{M}^{\alpha}$.
The semantics of λ-terms in $\mathbb{M}$ is inductively defined by:

- $[\![c]\!]_{\chi}^{\mathbb{M}} = \iota(c)$,

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.
- $\mathcal{M}^{\alpha\to\beta}$ is the set of all functions from $\mathcal{M}^\alpha$ to $\mathcal{M}^\beta$.
- ι maps constants of type α to $\mathcal{M}^\alpha$

A variable assignment $\chi : V \to \bigcup_{\alpha\in\mathcal{T}(\Sigma)} \mathcal{M}^\alpha$ so that $\chi(x^\alpha) \in \mathcal{M}^\alpha$.
The semantics of λ-terms in $\mathbb{M}$ is inductively defined by:

- $[\![c]\!]^{\mathbb{M}}_\chi = \iota(c)$,
- $[\![x^\alpha]\!]^{\mathbb{M}}_\chi = \chi(x^\alpha)$,

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^\alpha$ to $\mathcal{M}^\beta$.
- ι maps constants of type α to $\mathcal{M}^\alpha$

A variable assignment $\chi : V \to \bigcup_{\alpha \in \mathcal{T}(\Sigma)} \mathcal{M}^\alpha$ so that $\chi(x^\alpha) \in \mathcal{M}^\alpha$.
The semantics of λ-terms in $\mathbb{M}$ is inductively defined by:

- $[\![c]\!]^{\mathbb{M}}_\chi = \iota(c)$,
- $[\![x^\alpha]\!]^{\mathbb{M}}_\chi = \chi(x^\alpha)$,
- $[\![MN]\!]^{\mathbb{M}}_\chi = [\![M]\!]^{\mathbb{M}}_\chi([\![N]\!]^{\mathbb{M}}_\chi)$,

Finite models for recognizability in the simply typed λ-calculus

Let Σ be a HOS. $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$ is a finite model of Σ if:

- The sets $\mathcal{M}^\alpha$ are finite.
- $\mathcal{M}^{\alpha \to \beta}$ is the set of all functions from $\mathcal{M}^\alpha$ to $\mathcal{M}^\beta$.
- ι maps constants of type α to $\mathcal{M}^\alpha$

A variable assignment $\chi : V \to \bigcup_{\alpha \in \mathcal{T}(\Sigma)} \mathcal{M}^\alpha$ so that $\chi(x^\alpha) \in \mathcal{M}^\alpha$.
The semantics of λ-terms in $\mathbb{M}$ is inductively defined by:

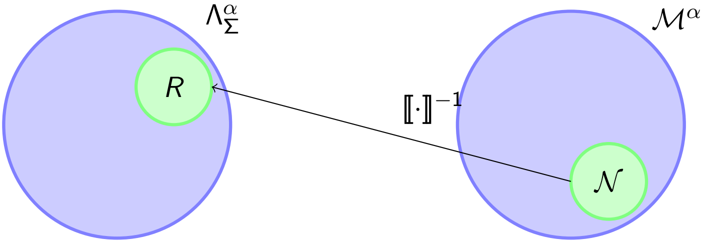
- $[\![c]\!]_\chi^{\mathbb{M}} = \iota(c)$,
- $[\![x^\alpha]\!]_\chi^{\mathbb{M}} = \chi(x^\alpha)$,
- $[\![MN]\!]_\chi^{\mathbb{M}} = [\![M]\!]_\chi^{\mathbb{M}}([\![N]\!]_\chi^{\mathbb{M}})$,
- $[\![\lambda x^\alpha.M]\!]_\chi^{\mathbb{M}}(a) = [\![M]\!]_{\chi \leftarrow [x^\alpha := a]}^{\mathbb{M}}$ with $a \in \mathcal{M}^\alpha$.

Finite models for recognizability in the simply typed λ-calculus

Definition:

A set of λ-terms $R \subseteq \Lambda_{\Sigma}^{\alpha}$ is recognizable iff there is a finite full model $\mathbb{M} = ((\mathcal{M}^{\alpha})_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$, $\mathcal{N} \subseteq \mathcal{M}^{\alpha}$:

$$R = \{M | FV(M) = \emptyset \wedge [\![M]\!]^{\mathbb{M}} \in \mathcal{N}\}$$

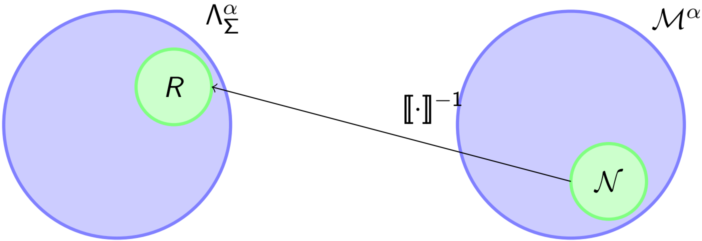


Finite models for recognizability in the simply typed λ-calculus

Definition:

A set of λ-terms $R \subseteq \Lambda^{\alpha}_{\Sigma}$ is recognizable iff there is a finite full model $\mathbb{M} = ((\mathcal{M}^{\alpha})_{\alpha \in \mathcal{T}(\Sigma)}, \iota)$, $\mathcal{N} \subseteq \mathcal{M}^{\alpha}$:

$$R = \{M | FV(M) = \emptyset \wedge [\![M]\!]^{\mathbb{M}} \in \mathcal{N}\}$$



Note:

- recognizable sets are closed under $=_{\beta\eta}$
- the emptiness of recognizable sets subsumes λ-definability which is undecidable (Loader 1993).

Properties of recognizable sets of λ-terms

- R is a recognizable set of λ-strings iff $\{w \mid /w/ \in R\}$ is a recognizable set of strings (similarly for λ-trees/trees).

Properties of recognizable sets of λ-terms

- R is a recognizable set of λ-strings iff $\{w \mid /w/ \in R\}$ is a recognizable set of strings (similarly for λ-trees/trees).
- The class of recognizable sets of λ-terms is closed under Boolean operations.

Properties of recognizable sets of λ-terms

- R is a recognizable set of λ-strings iff $\{w \mid /w/ \in R\}$ is a recognizable set of strings (similarly for λ-trees/trees).
- The class of recognizable sets of λ-terms is closed under Boolean operations.
- It is also closed under inverse homomorphism of λ-terms (CCC-functor).

Properties of recognizable sets of λ-terms

- R is a recognizable set of λ-strings iff $\{w \mid /w/ \in R\}$ is a recognizable set of strings (similarly for λ-trees/trees).
- The class of recognizable sets of λ-terms is closed under Boolean operations.
- It is also closed under inverse homomorphism of λ-terms (CCC-functor).
- There is a mechanical (equivalent) characterization of recognizability in terms of intersection types.

Properties of recognizable sets of λ-terms

- R is a recognizable set of λ-strings iff $\{w \mid /w/ \in R\}$ is a recognizable set of strings (similarly for λ-trees/trees).
- The class of recognizable sets of λ-terms is closed under Boolean operations.
- It is also closed under inverse homomorphism of λ-terms (CCC-functor).
- There is a mechanical (equivalent) characterization of recognizability in terms of intersection types.
- An approach based on finite standard model gives a simple proof of the decidability of the acceptance by a Büchi tree automaton of the infinite tree generated by a higher-order programming scheme (S., Srivathsan, Walukiewicz).

Outline

Cartesian Closed Category

$\mathcal{C}$ is a Cartesian Close Category if:

- $\mathcal{C}$ is a category,

Cartesian Closed Category

$\mathcal{C}$ is a Cartesian Close Category if:

- $\mathcal{C}$ is a category,
- it has a terminal object 1,

Cartesian Closed Category

$\mathcal{C}$ is a Cartesian Close Category if:

- $\mathcal{C}$ is a category,
- it has a terminal object 1,
- for every pair of objects α and β, there is:
  - a *product-object* $\alpha \times \beta$, with associated projection $\pi_1 : \alpha \times \beta \to \alpha$ and $\pi_2 : \alpha \times \beta \to \beta$,

Cartesian Closed Category

$\mathcal{C}$ is a Cartesian Close Category if:

- $\mathcal{C}$ is a category,
- it has a terminal object 1,
- for every pair of objects α and β, there is:
  - a *product-object* $\alpha \times \beta$, with associated projection $\pi_1 : \alpha \times \beta \to \alpha$ and $\pi_2 : \alpha \times \beta \to \beta$,
  - an *exponential-object* α^β such that $\mathit{Hom}(\alpha \times \beta, \delta) \cong \mathit{Hom}(\alpha, \delta^\beta)$

Cartesian Closed Category

$\mathcal{C}$ is a Cartesian Close Category if:

- $\mathcal{C}$ is a category,
- it has a terminal object 1,
- for every pair of objects α and β, there is:
  - a *product-object* $\alpha \times \beta$, with associated projection $\pi_1 : \alpha \times \beta \rightarrow \alpha$ and $\pi_2 : \alpha \times \beta \rightarrow \beta$,
  - an *exponential-object* α^β such that $Hom(\alpha \times \beta, \delta) \cong Hom(\alpha, \delta^\beta)$

A CCC-functor is a morphism of CCC, *i.e.* it commutes with products and exponentials.

Cartesian Closed Categories and congruences

Given a HOS Σ, Λ_Σ (up to $\beta\eta$-convertibility) forms a CCC:

- Objects: types and products of types

Cartesian Closed Categories and congruences

Given a HOS Σ, Λ_Σ (up to $\beta\eta$-convertibility) forms a CCC:

- Objects: types and products of types
- Arrows: $\Gamma \vdash M : \alpha$ where:
  - $\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$ is interpreted as the object $\beta = \alpha_1 \times \ldots \times \alpha_n$
  - M is an arrow $\beta \to \alpha$.

Cartesian Closed Categories and congruences

Given a HOS Σ, Λ_Σ (up to $\beta\eta$-convertibility) forms a CCC:

- Objects: types and products of types
- Arrows: $\Gamma \vdash M : \alpha$ where:
  - $\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$ is interpreted as the object $\beta = \alpha_1 \times \ldots \times \alpha_n$
  - M is an arrow $\beta \rightarrow \alpha$.
  - remark: when Γ is empty then M is an arrow $1 \rightarrow \alpha$.

Cartesian Closed Categories and congruences

Given a HOS Σ, Λ_Σ (up to $\beta\eta$-convertibility) forms a CCC:

- Objects: types and products of types
- Arrows: $\Gamma \vdash M : \alpha$ where:
  - $\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$ is interpreted as the object $\beta = \alpha_1 \times \ldots \times \alpha_n$
  - M is an arrow $\beta \to \alpha$.
  - remark: when Γ is empty then M is an arrow $1 \to \alpha$.

Given a congruence $\equiv$ on Λ_Σ, $\Lambda_\Sigma/\equiv$ forms a CCC, the arrows are equivalence classes of λ-terms.

Cartesian Closed Categories and congruences

Given a HOS Σ, Λ_Σ (up to $\beta\eta$-convertibility) forms a CCC:

- Objects: types and products of types
- Arrows: $\Gamma \vdash M : \alpha$ where:
  - $\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$ is interpreted as the object $\beta = \alpha_1 \times \ldots \times \alpha_n$
  - M is an arrow $\beta \to \alpha$.
  - remark: when Γ is empty then M is an arrow $1 \to \alpha$.

Given a congruence $\equiv$ on Λ_Σ, $\Lambda_\Sigma/\equiv$ forms a CCC, the arrows are equivalence classes of λ-terms.
We write $F_\equiv$ for the surjective functor from Λ_Σ to $\Lambda_\Sigma/\equiv$.

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq \mathit{Hom}(\beta, \alpha)$ and f_1, f_2 in $\mathit{Hom}(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq \mathit{Hom}(\beta, \alpha)$ and f_1, f_2 in $\mathit{Hom}(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,
- when $\mathcal{C} = \Lambda_\Sigma$, $\Lambda_\Sigma / {\sim_A}$ is *the syntactic CCC* associated to the language A,

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq Hom(\beta, \alpha)$ and f_1, f_2 in $Hom(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,
- when $\mathcal{C} = \Lambda_\Sigma$, $\Lambda_\Sigma/{\sim_A}$ is *the syntactic CCC* associated to the language A,
- whenever $\approx$ is a congruence on $\mathcal{C}$ and $A = F_{\approx}^{-1}(B)$ for $B \subseteq Hom_{\Lambda_\Sigma/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_\Sigma/{\approx} \to \Lambda_\Sigma/{\sim_A}$,

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq Hom(\beta, \alpha)$ and f_1, f_2 in $Hom(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,
- when $\mathcal{C} = \Lambda_\Sigma$, $\Lambda_\Sigma/{\sim_A}$ is *the syntactic CCC* associated to the language A,
- whenever $\approx$ is a congruence on $\mathcal{C}$ and $A = F_{\approx}^{-1}(B)$ for $B \subseteq Hom_{\Lambda_\Sigma/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_\Sigma/{\approx} \to \Lambda_\Sigma/{\sim_A}$,
- the syntactic CCC of a recognizable set of λ-terms is locally finite (*i.e.* for every α, β, $Hom_{\Lambda_\Sigma/\sim_R}(\alpha, \beta)$ is finite),

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq Hom(\beta, \alpha)$ and f_1, f_2 in $Hom(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,
- when $\mathcal{C} = \Lambda_\Sigma$, $\Lambda_\Sigma/{\sim_A}$ is *the syntactic CCC* associated to the language A,
- whenever $\approx$ is a congruence on $\mathcal{C}$ and $A = F_{\approx}^{-1}(B)$ for $B \subseteq Hom_{\Lambda_\Sigma/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_\Sigma/\approx \to \Lambda_\Sigma/{\sim_A}$,
- the syntactic CCC of a recognizable set of λ-terms is locally finite (*i.e.* for every α, β, $Hom_{\Lambda_\Sigma/\sim_R}(\alpha, \beta)$ is finite),
- **conjecture:** every language of λ-terms that has a locally finite syntactic CCC is recognizable.

Syntactic CCC of a language

Given a CCC $\mathcal{C}$ and $A \subseteq Hom(\beta, \alpha)$ and f_1, f_2 in $Hom(\theta, \delta)$, we have:

$$f_1 \sim_A f_2 \text{ iff } \forall C[].C[f_1] \in A \Leftrightarrow C[f_2] \in A$$

- $\sim_A$ is a congruence of CCC,
- when $\mathcal{C} = \Lambda_\Sigma$, $\Lambda_\Sigma/\sim_A$ is *the syntactic CCC* associated to the language A,
- whenever $\approx$ is a congruence on $\mathcal{C}$ and $A = F_{\approx}^{-1}(B)$ for $B \subseteq Hom_{\Lambda_\Sigma/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_\Sigma/\approx \to \Lambda_\Sigma/\sim_A$,
- the syntactic CCC of a recognizable set of λ-terms is locally finite (*i.e.* for every α, β, $Hom_{\Lambda_\Sigma/\sim_R}(\alpha, \beta)$ is finite),
- **conjecture:** every language of λ-terms that has a locally finite syntactic CCC is recognizable.

For the moment we call C-recognizable a language whose syntactic CCC is locally finite.

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

This implies:

- $u \equiv_R v$ iff $/u/ \sim_{R'} /v/$,

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

This implies:

- $u \equiv_R v$ iff $/u/ \sim_{R'} /v/$,
- $\sim_{R'}$ is an extension of $\equiv_R$ to higher-order functions over strings,

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

This implies:

- $u \equiv_R v$ iff $/u/ \sim_{R'} /v/$,
- $\sim_{R'}$ is an extension of $\equiv_R$ to higher-order functions over strings,
- the CCC associated to $\sim_{R'}$ is embedding the syntactic monoid of R (it is concretely represented by $Hom(o, o)$),

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

This implies:

- $u \equiv_R v$ iff $/u/ \sim_{R'} /v/$,
- $\sim_{R'}$ is an extension of $\equiv_R$ to higher-order functions over strings,
- the CCC associated to $\sim_{R'}$ is embedding the syntactic monoid of R (it is concretely represented by $Hom(o, o)$),
- every λ-string language that has a locally finite syntactic CCC is a recognizable set of λ-terms.

Embedding of syntactic monoid within syntactic CCC

Given R a recognizable set of strings, and R' be the recognizable set of λ-terms representing the elements of R:

- $u \equiv_R v$ iff for every w_1, w_2, $w_1 u w_2 \in R \Leftrightarrow w_1 v w_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every M, $M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$, $w_0 u w_1 \ldots w_{n-1} u w_n \in R \Leftrightarrow w_0 v w_1 \ldots w_{n-1} v w_n \in R$

This implies:

- $u \equiv_R v$ iff $/u/ \sim_{R'} /v/$,
- $\sim_{R'}$ is an extension of $\equiv_R$ to higher-order functions over strings,
- the CCC associated to $\sim_{R'}$ is embedding the syntactic monoid of R (it is concretely represented by $Hom(o, o)$),
- every λ-string language that has a locally finite syntactic CCC is a recognizable set of λ-terms.

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of λ-terms.

Outline

Classification of recognizable sets of λ-terms

We have syntactic objects that fully characterize languages of λ-terms:

- can we classify these languages in terms of properties of their syntactic CCCs?

Classification of recognizable sets of λ-terms

We have syntactic objects that fully characterize languages of λ-terms:

- can we classify these languages in terms of properties of their syntactic CCCs?
- we try to extend classification tools used for recognizable string languages:

Classification of recognizable sets of λ-terms

We have syntactic objects that fully characterize languages of λ-terms:

- can we classify these languages in terms of properties of their syntactic CCCs?
- we try to extend classification tools used for recognizable string languages:
  - we define varieties of locally finite CCCs,

Classification of recognizable sets of λ-terms

We have syntactic objects that fully characterize languages of λ-terms:

- can we classify these languages in terms of properties of their syntactic CCCs?
- we try to extend classification tools used for recognizable string languages:
  - we define varieties of locally finite CCCs,
  - and varieties of languages of λ-terms.

Varieties of finite monoids

A variety of finite monoids $\mathbf{V}$ is a class of finite monoids with the following closure properties:

- If M_1 and M_2 are $\mathbf{V}$, then $M_1 \times M_2$ is also in $\mathbf{V}$,
- If M_1 is a submonoid of M_2 and M_2 is in $\mathbf{V}$, then M_1 is also in $\mathbf{V}$
- If M is in $\mathbf{V}$ and $\equiv$ is a congruence on M, then $M/\equiv$ is in $\mathbf{V}$

Varieties of languages

A variety of recognizable languages $\mathcal{V}$ is a class of recognizable languages with the following closure properties ($\Sigma\mathcal{V}$ is the class of languages in $\mathcal{V}$ on alphabet Σ):

- $\Sigma\mathcal{V}$ is closed under Boolean operations,
- If R is in $\Sigma\mathcal{V}$, then $a^{-1}R$ and Ra^{-1} are in $\Sigma\mathcal{V}$ for every a in Σ.
- If $f : \Gamma^* \to \Sigma^*$ is a morphism of monoid, then $R \in \Sigma\mathcal{V}$ implies $f^{-1}(A) \in \Gamma\mathcal{V}$.

Eilenberg theorem

Given a recognizable language of strings R, we let M_R be its syntactic monoid.
Given $\mathcal{V}$ a variety of languages and $\mathbf{V}$ a variety of finite monoids we let:

- $\overline{\mathcal{V}}$ be the variety of finite monoids generated by $\{M_R \mid R \in \Sigma\mathcal{V} \text{ for some } \Sigma\}$,

Eilenberg theorem

Given a recognizable language of strings R, we let M_R be its syntactic monoid.
Given $\mathcal{V}$ a variety of languages and $\mathbf{V}$ a variety of finite monoids we let:

- $\overline{\mathcal{V}}$ be the variety of finite monoids generated by $\{M_R \mid R \in \Sigma\mathcal{V} \text{ for some } \Sigma\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma\}$.

Eilenberg theorem

Given a recognizable language of strings R, we let M_R be its syntactic monoid.
Given $\mathcal{V}$ a variety of languages and $\mathbf{V}$ a variety of finite monoids we let:

- $\overline{\mathcal{V}}$ be the variety of finite monoids generated by $\{M_R \mid R \in \Sigma\mathcal{V} \text{ for some } \Sigma\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of languages,

Eilenberg theorem

Given a recognizable language of strings R, we let M_R be its syntactic monoid.
Given $\mathcal{V}$ a variety of languages and $\mathbf{V}$ a variety of finite monoids we let:

- $\overline{\mathcal{V}}$ be the variety of finite monoids generated by $\{M_R \mid R \in \Sigma\mathcal{V} \text{ for some } \Sigma\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of languages,
- $\mathcal{V} = \widetilde{\overline{\mathcal{V}}}$,

Eilenberg theorem

Given a recognizable language of strings R, we let M_R be its syntactic monoid.
Given $\mathcal{V}$ a variety of languages and $\mathbf{V}$ a variety of finite monoids we let:

- $\overline{\mathcal{V}}$ be the variety of finite monoids generated by $\{M_R \mid R \in \Sigma\mathcal{V} \text{ for some } \Sigma\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of languages,
- $\mathcal{V} = \widetilde{\overline{\mathcal{V}}}$,
- $\mathbf{V} = \overline{\widetilde{\mathbf{V}}}$

An application of Eilenberg Theorem

If we let:

- $\mathcal{SF}$ = the variety of star-free languages = first-order definable languages
- $\mathcal{AP}$ = the variety of aperiodic monoids

We obtain Schützenberger-McNaughton-Papert's result:

$$\overline{\mathcal{SF}} = \mathcal{AP}$$

Outline

Finitely generated CCCs

A CCC $\mathcal{C}$ is finitely generated if there is HOS Σ and a surjective CCC-functor $F : \Lambda_\Sigma \to \mathcal{C}$. F is called *a finite presentation* of $\mathcal{C}$.

Finitely generated CCCs

A CCC $\mathcal{C}$ is finitely generated if there is HOS Σ and a surjective CCC-functor $F : \Lambda_\Sigma \to \mathcal{C}$. F is called *a finite presentation* of $\mathcal{C}$.

- A locally finite CCC may not be finitely generated (ex: Heyting algebra with infinitely many generators).

Finitely generated CCCs

A CCC $\mathcal{C}$ is finitely generated if there is HOS Σ and a surjective CCC-functor $F : \Lambda_\Sigma \to \mathcal{C}$. F is called *a finite presentation* of $\mathcal{C}$.

- A locally finite CCC may not be finitely generated (ex: Heyting algebra with infinitely many generators).
- To obtain an extension of Eilenberg Theorem we need to impose that we only consider finitely generated CCCs.

Product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects, a simple idea to generalize the direct product of monoids is to take $\mathcal{C}_1 \times \mathcal{C}_2$ with:

- the objects of $\mathcal{C}_1 \times \mathcal{C}_2$ is the same as the ones of $\mathcal{C}_1$ and $\mathcal{C}_2$,

Product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects, a simple idea to generalize the direct product of monoids is to take $\mathcal{C}_1 \times \mathcal{C}_2$ with:

- the objects of $\mathcal{C}_1 \times \mathcal{C}_2$ is the same as the ones of $\mathcal{C}_1$ and $\mathcal{C}_2$,
- $Hom_{\mathcal{C}_1 \times \mathcal{C}_2}(\alpha, \beta) = Hom_{\mathcal{C}_1}(\alpha, \beta) \times Hom_{\mathcal{C}_2}(\alpha, \beta)$

Product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects, a simple idea to generalize the direct product of monoids is to take $\mathcal{C}_1 \times \mathcal{C}_2$ with:

- the objects of $\mathcal{C}_1 \times \mathcal{C}_2$ is the same as the ones of $\mathcal{C}_1$ and $\mathcal{C}_2$,
- $Hom_{\mathcal{C}_1 \times \mathcal{C}_2}(\alpha, \beta) = Hom_{\mathcal{C}_1}(\alpha, \beta) \times Hom_{\mathcal{C}_2}(\alpha, \beta)$
- $\mathcal{C}_1 \times \mathcal{C}_2$, is a locally finite CCC,

Product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects, a simple idea to generalize the direct product of monoids is to take $\mathcal{C}_1 \times \mathcal{C}_2$ with:

- the objects of $\mathcal{C}_1 \times \mathcal{C}_2$ is the same as the ones of $\mathcal{C}_1$ and $\mathcal{C}_2$,
- $Hom_{\mathcal{C}_1 \times \mathcal{C}_2}(\alpha, \beta) = Hom_{\mathcal{C}_1}(\alpha, \beta) \times Hom_{\mathcal{C}_2}(\alpha, \beta)$
- $\mathcal{C}_1 \times \mathcal{C}_2$, is a locally finite CCC,
- but $\mathcal{C}_1 \times \mathcal{C}_2$ may not be finitely generated...

Product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects, a simple idea to generalize the direct product of monoids is to take $\mathcal{C}_1 \times \mathcal{C}_2$ with:

- the objects of $\mathcal{C}_1 \times \mathcal{C}_2$ is the same as the ones of $\mathcal{C}_1$ and $\mathcal{C}_2$,
- $Hom_{\mathcal{C}_1 \times \mathcal{C}_2}(\alpha, \beta) = Hom_{\mathcal{C}_1}(\alpha, \beta) \times Hom_{\mathcal{C}_2}(\alpha, \beta)$
- $\mathcal{C}_1 \times \mathcal{C}_2$, is a locally finite CCC,
- but $\mathcal{C}_1 \times \mathcal{C}_2$ may not be finitely generated...

Thus given two finite presentation F_1 and F_2 of $\mathcal{C}_1$ and $\mathcal{C}_2$, we define $\mathcal{C}_1 \times_{F_1, F_2} \mathcal{C}_2$ to be the sub-CCC of $\mathcal{C}_1 \times \mathcal{C}_2$ generated by the arrows:

$$\bigcup_{c \in \Sigma_1} \{F_1(c)\} \times Hom_{\mathcal{C}_2}(1, \tau_1(c)) \cup \bigcup_{c \in \Sigma_2} Hom_{\mathcal{C}_1}(1, \tau_2(c)) \times \{F_2(c)\}$$

Direct product of monoids and product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects and which are generated only by string signatures:

- for every presentation F_1, G_1 and F_2, G_2 of respectively $\mathcal{C}_1$ and $\mathcal{C}_2$ we have

$$\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2 = \mathcal{C}_1 \times_{G_1,G_2} \mathcal{C}_2$$

Direct product of monoids and product of CCCs

Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two locally finite and finitely generated CCCs, that have the same objects and which are generated only by string signatures:

- for every presentation F_1, G_1 and F_2, G_2 of respectively $\mathcal{C}_1$ and $\mathcal{C}_2$ we have

$$\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2 = \mathcal{C}_1 \times_{G_1,G_2} \mathcal{C}_2$$

- a question is whether for every locally finite and finitely generated CCC, $\mathcal{C}_1$ and $\mathcal{C}_2$ we can find a canonical sub-CCC of $\mathcal{C}_1 \times \mathcal{C}_2$ that is finitely generated.

α-syntactic and α-separated CCC

A *CCC* $\mathcal{C}$ is said α-syntactic if there is a subset of A of $Hom(1, \alpha)$ such that for every f_1, f_2 in $Hom(\theta, \delta)$:

$$f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2$$

α-syntactic and α-separated CCC

A *CCC* $\mathcal{C}$ is said α-syntactic if there is a subset of A of $Hom(1, \alpha)$ such that for every f_1, f_2 in $Hom(\theta, \delta)$:

$$f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2$$

We then have:

- if $\mathcal{C}_1$, $\mathcal{C}_2$ are two α-syntactic CCC, then for every presentation F_1 and F_2 of $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2$ is α-syntactic.

α-syntactic and α-separated CCC

A *CCC* $\mathcal{C}$ is said α-syntactic if there is a subset of A of $Hom(1, \alpha)$ such that for every f_1, f_2 in $Hom(\theta, \delta)$:

$$f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2$$

We then have:

- if $\mathcal{C}_1$, $\mathcal{C}_2$ are two α-syntactic CCC, then for every presentation F_1 and F_2 of $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2$ is α-syntactic.
- it can be the case that a locally finite finitely generated CCC $\mathcal{C}$ can not be constructed from α-syntactic CCCs using product, sub-CCC and quotient.

α-syntactic and α-separated CCC

A *CCC* $\mathcal{C}$ is said α-syntactic if there is a subset of A of $Hom(1,\alpha)$ such that for every f_1, f_2 in $Hom(\theta,\delta)$:

$$f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2$$

We then have:

- if $\mathcal{C}_1$, $\mathcal{C}_2$ are two α-syntactic CCC, then for every presentation F_1 and F_2 of $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2$ is α-syntactic.
- it can be the case that a locally finite finitely generated CCC $\mathcal{C}$ can not be constructed from α-syntactic CCCs using product, sub-CCC and quotient.
- but this is the case when $\mathcal{C}$ is α-separated:
  - for every f_1, f_2 in $Hom(\theta,\delta)$, $f_1 \neq f_2$ iff there is $C[]$ such that $C[f_1]$, $C[f_2]$ are in $Hom(1,\alpha)$ and $C[f_1] \neq C[f_2]$.

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- α is an object of $\mathcal{C}$ and $\mathcal{C}$ is α-separated

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- α is an object of $\mathcal{C}$ and $\mathcal{C}$ is α-separated
- for every $(\mathcal{C}_1, \alpha)$ and $(\mathcal{C}_2, \alpha)$, and every presentation of F_1 and F_2 of $\mathcal{C}_1$ and C_2, $(\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2, \alpha)$ is in **V**

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- α is an object of $\mathcal{C}$ and $\mathcal{C}$ is α-separated
- for every $(\mathcal{C}_1, \alpha)$ and $(\mathcal{C}_2, \alpha)$, and every presentation of F_1 and F_2 of $\mathcal{C}_1$ and C_2, $(\mathcal{C}_1 \times_{F_1, F_2} \mathcal{C}_2, \alpha)$ is in **V**
- if $(\mathcal{C}, \alpha)$ is in **V** and $\mathcal{C}'$ is a sub-CCC, then if C'' is the β-separated CCC obtained from C', (C'', β) is in **V**

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- α is an object of $\mathcal{C}$ and $\mathcal{C}$ is α-separated
- for every $(\mathcal{C}_1, \alpha)$ and $(\mathcal{C}_2, \alpha)$, and every presentation of F_1 and F_2 of $\mathcal{C}_1$ and C_2, $(\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2, \alpha)$ is in **V**
- if $(\mathcal{C}, \alpha)$ is in **V** and $\mathcal{C}'$ is a sub-CCC, then if C'' is the β-separated CCC obtained from C', (C'', β) is in **V**
- if $(\mathcal{C}, \alpha)$ is in **V**, $\approx$ is a congruence of $\mathcal{C}$, and $\mathcal{C}'$ is the β-separated CCC obtained from $\mathcal{C}/\approx$ then $(\mathcal{C}', \beta)$ is in **V**.

Varieties of locally finite CCC

A variety of locally finite CCC **V** is a class of pairs $(\mathcal{C}, \alpha)$ such that:

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- α is an object of $\mathcal{C}$ and $\mathcal{C}$ is α-separated
- for every $(\mathcal{C}_1, \alpha)$ and $(\mathcal{C}_2, \alpha)$, and every presentation of F_1 and F_2 of $\mathcal{C}_1$ and C_2, $(\mathcal{C}_1 \times_{F_1, F_2} \mathcal{C}_2, \alpha)$ is in **V**
- if $(\mathcal{C}, \alpha)$ is in **V** and $\mathcal{C}'$ is a sub-CCC, then if C'' is the β-separated CCC obtained from C', (C'', β) is in **V**
- if $(\mathcal{C}, \alpha)$ is in **V**, $\approx$ is a congruence of $\mathcal{C}$, and $\mathcal{C}'$ is the β-separated CCC obtained from $\mathcal{C}/\approx$ then $(\mathcal{C}', \beta)$ is in **V**.

we write $(\mathcal{C}_1, \beta) \prec (\mathcal{C}_2, \alpha)$ when $\mathcal{C}_1$ is an β-separated CCC obtained by taking and quotienting a sub-CCC of $\mathcal{C}_2$.

Towards varieties of languages

Given $\mathcal{C}$ a locally finite, finitely generated and α-separated CCC, A and A' included in $Hom(1, \alpha)$ we have:

- $\mathcal{C}/\sim_A = \mathcal{C}/\sim_B$ with $B = Hom(1, \alpha) - A$

Towards varieties of languages

Given $\mathcal{C}$ a locally finite, finitely generated and α-separated CCC, A and A' included in $Hom(1, \alpha)$ we have:

- $\mathcal{C}/\!\sim_A = \mathcal{C}/\!\sim_B$ with $B = Hom(1, \alpha) - A$
- $(\mathcal{C}/\!\sim_{A \cap A'}, \alpha) \prec (\mathcal{C}/\!\sim_A \times_{F,F} \mathcal{C}/\!\sim_{A'}, \alpha)$ for every presentation F of $\mathcal{C}$,

Towards varieties of languages

Given $\mathcal{C}$ a locally finite, finitely generated and α-separated CCC, A and A' included in $Hom(1, \alpha)$ we have:

- $\mathcal{C}/\sim_A = \mathcal{C}/\sim_B$ with $B = Hom(1, \alpha) - A$
- $(\mathcal{C}/\sim_{A \cap A'}, \alpha) \prec (\mathcal{C}/\sim_A \times_{F,F} \mathcal{C}/\sim_{A'}, \alpha)$ for every presentation F of $\mathcal{C}$,
- Given $C[]$ such that for every $f \in Hom(1, \beta)$, $C[f]$ is in $Hom(1, \alpha)$, if $C^{-1}[A] = \{f \in Hom(1, \beta) \mid C[f] \in A\}$ then $\mathcal{C}/\sim_{C^{-1}[A]}$ is a quotient CCC of $\mathcal{C}/\sim_A$

Towards varieties of languages

Given $\mathcal{C}$ a locally finite, finitely generated and α-separated CCC, A and A' included in $Hom(1, \alpha)$ we have:

- $\mathcal{C}/\sim_A = \mathcal{C}/\sim_B$ with $B = Hom(1, \alpha) - A$
- $(\mathcal{C}/\sim_{A \cap A'}, \alpha) \prec (\mathcal{C}/\sim_A \times_{F,F} \mathcal{C}/\sim_{A'}, \alpha)$ for every presentation F of $\mathcal{C}$,
- Given $C[]$ such that for every $f \in Hom(1, \beta)$, $C[f]$ is in $Hom(1, \alpha)$, if $C^{-1}[A] = \{f \in Hom(1, \beta) \mid C[f] \in A\}$ then $\mathcal{C}/\sim_{C^{-1}[A]}$ is a quotient CCC of $\mathcal{C}/\sim_A$

Given a CCC-functor $F : \mathcal{D} \to \mathcal{C}$ and β such that $F(\beta) = \alpha$, and $B = F^{-1}(A) \cap Hom_{\mathcal{D}}(1, \beta)$ then $(\mathcal{D}/\sim_B, \beta) \prec (\mathcal{C}/\sim_A, \alpha)$.

Varieties of λ-languages

A variety of C-recognizable sets of λ-terms $\mathcal{V}$ is a class of C-recognizable languages with the following closure properties ($(\Sigma, \alpha)\mathcal{V}$ is the class of languages in $\mathcal{V}$ on a HOS Σ whose elements have type α):

- $(\Sigma, \alpha)\mathcal{V}$ is closed under Boolean operations

Varieties of λ-languages

A variety of C-recognizable sets of λ-terms $\mathcal{V}$ is a class of C-recognizable languages with the following closure properties ($(\Sigma, \alpha)\mathcal{V}$ is the class of languages in $\mathcal{V}$ on a HOS Σ whose elements have type α):

- $(\Sigma, \alpha)\mathcal{V}$ is closed under Boolean operations
- Given $M \in \Lambda_{\Sigma}^{\beta \to \alpha}$, and R in $(\Sigma, \alpha)\mathcal{V}$, then $M^{-1}R = \{N \in \Lambda_{\Sigma}^{\beta} \mid MN \in R\}$ is in $(\Sigma, \beta)\mathcal{V}$,

Varieties of λ-languages

A variety of C-recognizable sets of λ-terms $\mathcal{V}$ is a class of C-recognizable languages with the following closure properties ($(\Sigma, \alpha)\mathcal{V}$ is the class of languages in $\mathcal{V}$ on a HOS Σ whose elements have type α):

- $(\Sigma, \alpha)\mathcal{V}$ is closed under Boolean operations
- Given $M \in \Lambda_{\Sigma}^{\beta \to \alpha}$, and R in $(\Sigma, \alpha)\mathcal{V}$, then $M^{-1}R = \{N \in \Lambda_{\Sigma}^{\beta} \mid MN \in R\}$ is in $(\Sigma, \beta)\mathcal{V}$,
- Given $F : \Lambda_{\Sigma_1} \to \Lambda_{\Sigma_2}$ a CCC-functor, if $R \in (\Sigma_2, \alpha)\mathcal{V}$ and $F(\beta) = \alpha$, then $F^{-1}(R) \cap \Lambda_{\Sigma_1}^{\beta} \in (\Sigma_1, \beta)\mathcal{V}$.

The correspondence

Given a C-recognizable set of λ-terms R, we let $\mathcal{C}_R$ be its syntactic CCC.
Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\overline{\mathcal{V}}$ be the variety of locally finite CCC generated by $\{(\mathcal{C}_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V}$ for some Σ and $\alpha\}$,

The correspondence

Given a C-recognizable set of λ-terms R, we let $\mathcal{C}_R$ be its syntactic CCC.
Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\overline{\mathcal{V}}$ be the variety of locally finite CCC generated by $\{(\mathcal{C}_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Lambda^{\alpha}_{\Sigma}, (\mathcal{C}_R, \alpha) \in \mathbf{V} \text{ for some } \Sigma \text{ and } \alpha\}$.

The correspondence

Given a C-recognizable set of λ-terms R, we let $\mathcal{C}_R$ be its syntactic CCC.
Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\overline{\mathcal{V}}$ be the variety of locally finite CCC generated by $\{(\mathcal{C}_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V}$ for some Σ and $\alpha\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Lambda_\Sigma^\alpha, (\mathcal{C}_R, \alpha) \in \mathbf{V}$ for some Σ and $\alpha\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of λ-languages,

The correspondence

Given a C-recognizable set of λ-terms R, we let $\mathcal{C}_R$ be its syntactic CCC.
Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\overline{\mathcal{V}}$ be the variety of locally finite CCC generated by $\{(\mathcal{C}_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V}$ for some Σ and $\alpha\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Lambda^{\alpha}_{\Sigma}, (\mathcal{C}_R, \alpha) \in \mathbf{V}$ for some Σ and $\alpha\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of λ-languages,
- $\mathcal{V} = \widetilde{\overline{\mathcal{V}}}$,

The correspondence

Given a C-recognizable set of λ-terms R, we let $\mathcal{C}_R$ be its syntactic CCC.
Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\overline{\mathcal{V}}$ be the variety of locally finite CCC generated by $\{(\mathcal{C}_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V}$ for some Σ and $\alpha\}$,
- $\widetilde{\mathbf{V}}$ be the class of languages $\{R \mid R \subseteq \Lambda^{\alpha}_{\Sigma}, (\mathcal{C}_R, \alpha) \in \mathbf{V}$ for some Σ and $\alpha\}$.

We then have:

- $\widetilde{\mathbf{V}}$ is a variety of λ-languages,
- $\mathcal{V} = \widetilde{\overline{\mathcal{V}}}$,
- $\mathbf{V} = \overline{\widetilde{\mathbf{V}}}$

Conclusion and future work.

- We have proved of an extension of the variety Theorem for C-recognizable languages.

Conclusion and future work.

- We have proved of an extension of the variety Theorem for C-recognizable languages.
- Variations on varieties:
  - tuning the relation $\prec$
  - using deduction systems to obtain structures similar so semigroups

Conclusion and future work.

- We have proved of an extension of the variety Theorem for C-recognizable languages.
- Variations on varieties:
  - tuning the relation $\prec$
  - using deduction systems to obtain structures similar so semigroups
- Equational definition of varieties.

Conclusion and future work.

- We have proved of an extension of the variety Theorem for C-recognizable languages.
- Variations on varieties:
  - tuning the relation $\prec$
  - using deduction systems to obtain structures similar so semigroups
- Equational definition of varieties.
- Applications of this work to languages of λ-terms that are neither λ-strings nor λ-trees rely on the conjecture recognizable $=$ C-recognizable.