Towards an algebraic classification of recognizable sets of lambda-terms

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 FREG
Outline

Recognizable sets of $\lambda$-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 \( \lambda \)-terms

Recognizability and congruence

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

Varieties of locally finite CCCs
\[\lambda\text{-calculus: syntax}\]

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

\[ \mathcal{T}_A := \mathcal{A}\mid(\mathcal{T}_A \rightarrow \mathcal{T}_A) \]
\textit{\lambda\text{-calculus: syntax}}

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

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

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,
- $\tau$ 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:

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

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,
- $\tau$ is a function from $\mathcal{C}$ to $\mathcal{T}_\mathcal{A}$.

λ-terms built on $\Sigma$ are defined as:
- for $\alpha \in \mathcal{T}_\mathcal{A}$, $x^\alpha \in \Lambda^\alpha_{\Sigma}$,
- $c \in \Lambda^{\tau(c)}_{\Sigma}$,
- if $M_1 \in \Lambda^{\alpha_2 \rightarrow \alpha_1}_{\Sigma}$, $M_2 \in \Lambda^{\alpha_2}_{\Sigma}$, then $(M_1 M_2) \in \Lambda^{\alpha_1}_{\Sigma}$,
- if $M \in \Lambda^{\alpha_1}_{\Sigma}$, then $\lambda x^{\alpha_2}. M \in \Lambda^{\alpha_2 \rightarrow \alpha_1}_{\Sigma}$. 
\(\lambda\)-calculus: operational semantics

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

\[
\begin{align*}
(\lambda x.M)N & \rightarrow_{\beta\eta} M[x := N] \\
\lambda x.M & \rightarrow_{\beta\eta} M \\
M_1 & \rightarrow_{\beta\eta} M_2 \\
(M_1 M) & \rightarrow_{\beta\eta} (M_2 M)
\end{align*}
\]

\[
\begin{align*}
\lambda x.M & \notin \text{FV}(M) \\
M_1 & \rightarrow_{\beta\eta} M_2 \\
(MM_1) & \rightarrow_{\beta\eta} (MM_2)
\end{align*}
\]

\(\beta\eta\)-reduction (\(\ast \rightarrow_{\beta\eta}\)): reflexive transitive closure of \(\beta\eta\)-contraction

\(\beta\eta\)-conversion: symmetric 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 \(M\).
λ-calculus: operational semantics

λ-calculus is a theory of function and computation. Computation is done with the relation of βη-contraction (→\textsubscript{βη}):

\[
\begin{align*}
(\lambda x. M) N & \quad \text{\textbf{\textcolor{red}{\lambda}}x. Mx \quad x \notin FV(M)} & \quad M_1 \rightarrow_{\beta\eta} M_2
\end{align*}
\]

\[
\begin{align*}
\lambda x. M[x := N] & \quad \text{\textbf{\textcolor{red}{\lambda}}x. M \rightarrow_{\beta\eta} M} & \quad (MM_1) \rightarrow_{\beta\eta} (MM_2)
\end{align*}
\]

\[
\begin{align*}
M_1 \rightarrow_{\beta\eta} M_2 & \quad \text{\textbf{\textcolor{red}{\lambda}}x. M \rightarrow_{\beta\eta} \lambda x. M_2}
\end{align*}
\]

\[
\begin{align*}
(M_1 M) \rightarrow_{\beta\eta} (MM_2) & \quad (\lambda x. M_1) \rightarrow_{\beta\eta} (\lambda x. M_2)
\end{align*}
\]

\[\beta\eta\text{-reduction (\rightarrow^*_\text{βη}}): \text{ reflexive transitive closure of } \beta\eta\text{-contraction}\]

\[\beta\eta\text{-conversion: symmetric closure of } \beta\eta\text{-reduction}\]
\textbf{λ-calculus: operational semantics}

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

\[
\frac{(λx.M)N}{(λx.M)N \rightarrow_{βη} M[x := N]} \quad \frac{λx.Mx \quad x \notin FV(M)}{λx.Mx \rightarrow_{βη} M} \quad \frac{M_1 \rightarrow_{βη} M_2}{M_1 \rightarrow_{βη} M_2} \quad \frac{(MM_1) \rightarrow_{βη} (MM_2)}{(MM_1) \rightarrow_{βη} (MM_2)}
\]

$βη$-reduction ($\xrightarrow{*_{βη}}$): reflexive transitive closure of $βη$-contraction

$βη$-conversion: symmetric closure of $βη$-reduction

**Theorem (Church-Rosser)**

$βη$-conversion is confluent
\(\lambda\)-calculus: operational semantics

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

\[
\begin{array}{c}
(\lambda x. M) N \\
\hline
(\lambda x. M) N \to_{\beta\eta} M[x := N]
\end{array}
\quad
\begin{array}{c}
\lambda x. M x \\
\hline
\lambda x. M x \to_{\beta\eta} M
\end{array}
\quad
\begin{array}{c}
M_1 \to_{\beta\eta} M_2
\end{array}
\]

\[
\begin{array}{c}
(\lambda x. M_1) \to_{\beta\eta} (\lambda x. M_2)
\end{array}
\quad
\begin{array}{c}
(M M_1) \to_{\beta\eta} (M M_2)
\end{array}
\]

\(\beta\eta\)-reduction (\(\to_{\ast\beta\eta}\)): reflexive transitive closure of \(\beta\eta\)-contraction

\(\beta\eta\)-conversion: symmetric closure of \(\beta\eta\)-reduction

**Theorem (Church-Rosser)**

\(\beta\eta\)-conversion is confluent

**Theorem (Strong Normalisation)**

Given \(M\) in \(\Lambda_\Sigma^\alpha\), there is no infinite sequence of \(\beta\eta\)-contraction starting in \(M\).
Simply typed $\lambda$-calculus generalizes trees

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

$$e : o, \quad g : o \to o, \quad f : o \to o \to o$$
Simply typed $\lambda$-calculus generalizes trees

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

\[ e : o, \quad g : o \rightarrow o, \quad f : o \rightarrow o \rightarrow o \]

the term $g(f(e, g(e)))$ is represented by the $\lambda$-term $g(f\_e(g\_e))$
Simply typed $\lambda$-calculus generalizes trees

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

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

the term $g(f(e, g(e)))$ is represented by the $\lambda$-term $g(f \, e \, (g \, e))$. The Böhm tree of the $\lambda$-term is the same as the graphic representation of the term:
Simply typed $\lambda$-calculus generalizes trees

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

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

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

The Böhm tree of the $\lambda$-term is the same as the graphic representation of the term:

```
    g
   /
  f
 /\n/  \
/   \
/     \
/       \
e      g
   /\    /\ 
  /   \/   \
 /     e    \
```

A $\lambda$-term whose normal form represent a tree is a $\lambda$-tree.
Simply typed $\lambda$-calculus generalizes strings

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

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

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

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

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

A $\lambda$-term whose normal form represent a string is a $\lambda$-string.
Simply typed $\lambda$-calculus generalizes strings

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

$$a : o \rightarrow o, \ b : o \rightarrow o$$

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

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

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

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

and the empty string is $\lambda x^o.x^o$
Simply typed $\lambda$-calculus generalizes strings

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

$$a : o \rightarrow o, \quad b : o \rightarrow o$$

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

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

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

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

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

A $\lambda$-term whose normal form represent a string is a $\lambda$-string.
Finite models for recognizability in the simply typed $\lambda$-calculus

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

- The sets $M^\alpha$ are finite.
Finite models for recognizability in the simply typed $\lambda$-calculus

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

- The sets $M^\alpha$ are finite.
- $M^{\alpha \rightarrow \beta}$ is the set of all functions from $M^\alpha$ to $M^\beta$. 
Finite models for recognizability in the simply typed \( \lambda \)-calculus

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

- The sets \( \mathcal{M}^\alpha \) are finite.
- \( \mathcal{M}^{\alpha \to \beta} \) is the set of all functions from \( \mathcal{M}^\alpha \) to \( \mathcal{M}^\beta \).
- \( \iota \) maps constants of type \( \alpha \) to \( \mathcal{M}^\alpha \).
Finite models for recognizability in the simply typed λ-calculus

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

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

A variable assignment $\chi : V \rightarrow \bigcup_{\alpha \in T(\Sigma)} \mathcal{M}^\alpha$ so that $\chi(x^\alpha) \in \mathcal{M}^\alpha$. 
Finite models for recognizability in the simply typed λ-calculus

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

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

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

The semantics of λ-terms in $\mathcal{M}$ is inductively defined by:

- $\llbracket c \rrbracket^\mathcal{M}_\chi = \iota(c)$,
Finite models for recognizability in the simply typed \(\lambda\)-calculus

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

- The sets \(\mathcal{M}^\alpha\) are finite.
- \(\mathcal{M}^{\alpha \rightarrow \beta}\) is the set of all functions from \(\mathcal{M}^\alpha\) to \(\mathcal{M}^\beta\).
- \(\iota\) maps constants of type \(\alpha\) to \(\mathcal{M}^\alpha\)

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

The semantics of \(\lambda\)-terms in \(\mathcal{M}\) is inductively defined by:

- \(\llbracket \text{c} \rrbracket^\mathcal{M}_\chi = \iota(c)\),
- \(\llbracket x^\alpha \rrbracket^\mathcal{M}_\chi = \chi(x^\alpha)\),
Finite models for recognizability in the simply typed $\lambda$-calculus

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

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

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

The semantics of $\lambda$-terms in $M$ is inductively defined by:

- $\llbracket c \rrbracket^M_\chi = \iota(c)$,
- $\llbracket x^\alpha \rrbracket^M_\chi = \chi(x^\alpha)$,
- $\llbracket MN \rrbracket^M_\chi = \llbracket M \rrbracket^M_\chi (\llbracket N \rrbracket^M_\chi)$,
Finite models for recognizability in the simply typed $\lambda$-calculus

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

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

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

The semantics of $\lambda$-terms in $\bar{M}$ is inductively defined by:

- $\lfloor c \rfloor^\bar{M}_\chi = \iota(c)$,
- $\lfloor x^\alpha \rfloor^\bar{M}_\chi = \chi(x^\alpha)$,
- $\lfloor MN \rfloor^\bar{M}_\chi = \lfloor M \rfloor^\bar{M}_\chi(\lfloor N \rfloor^\bar{M}_\chi)$,
- $\lfloor \lambda x^\alpha.M \rfloor^\bar{M}_\chi(a) = \lfloor M \rfloor^\bar{M}_\chi^{\leftarrow[x^\alpha := a]}$ with $a \in M^\alpha$. 
Finite models for recognizability in the simply typed $\lambda$-calculus

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

$$R = \{ M | FV(M) = \emptyset \land [M]^\mathbb{M} \in \mathcal{N} \}$$
Finite models for recognizability in the simply typed \( \lambda \)-calculus

**Definition:**
A set of \( \lambda \)-terms \( R \subseteq \Lambda^\alpha_\Sigma \) is recognizable iff there is a finite full model \( \mathcal{M} = (\langle M^\alpha \rangle_{\alpha \in T(\Sigma)}, \iota) \), \( \mathcal{N} \subseteq M^\alpha \):

\[
R = \{ M \mid FV(M) = \emptyset \land [M]^{\mathcal{M}} \in \mathcal{N} \}
\]

Note:
- Recognizable sets are closed under \( \beta\eta \)
- The emptiness of recognizable sets subsumes \( \lambda \)-definability which is undecidable (Loader 1993).
Properties of recognizable sets of $\lambda$-terms

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

- The class of recognizable sets of $\lambda$-terms is closed under Boolean operations.

- It is also closed under inverse homomorphism of $\lambda$-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).
Properties of recognizable sets of \( \lambda \)-terms

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

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

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

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

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

- $\mathcal{C}$ is a category,
- it has a terminal object $1$,
- for every pair of objects $\alpha$ and $\beta$, 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 $\alpha^\beta$ such that $\text{Hom}(\alpha \times \beta, \delta) \cong \text{Hom}(\alpha, \delta^\beta)$.

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

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

- $\mathcal{C}$ is a category,
- it has a terminal object $1$,
$\mathcal{C}$ is a Cartesian Close Category if:

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

- $\mathcal{C}$ is a category,
- it has a terminal object $1$, and
- for every pair of objects $\alpha$ and $\beta$, 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 $\alpha^\beta$ such that $\text{Hom}(\alpha \times \beta, \delta) \cong \text{Hom}(\alpha, \delta^\beta)$.

A CCC-functor is a morphism of CCC, i.e. it commutes with products and exponentials.
A 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 $\alpha$ and $\beta$, 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 $\alpha^\beta$ such that $\text{Hom}(\alpha \times \beta, \delta) \cong \text{Hom}(\alpha, \delta^\beta)$.

A CCC-functor is a morphism of CCC, i.e. it commutes with products and exponentials.
Given a HOS $\Sigma$, $\Lambda_{\Sigma}$ (up to $\beta\eta$-convertibility) forms a CCC:

- **Objects:** types and products of types

Remark: when $\Gamma$ is empty then $M$ is an arrow $1 \to \alpha$. Given a congruence $\equiv$ on $\Lambda_{\Sigma}$, $\Lambda_{\Sigma}/\equiv$ forms a CCC, the arrows are equivalence classes of $\lambda$-terms.
Cartesian Closed Categories and congruences

Given a HOS $\Sigma$, $\Lambda_\Sigma$ (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 $\beta \rightarrow \alpha$. 

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

We write $F \equiv$ for the surjective functor from $\Lambda_\Sigma$ to $\Lambda_\Sigma/\equiv$. 
Given a HOS $\Sigma$, $\Lambda_{\Sigma}$ (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 $\Gamma$ is empty then $M$ is an arrow $1 \to \alpha$. 

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

We write $F\equiv$ for the surjective functor from $\Lambda_{\Sigma}$ to $\Lambda_{\Sigma}/\equiv$. 

Given a HOS $\Sigma$, $\Lambda^\Sigma$ (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 $\Gamma$ is empty then $M$ is an arrow $1 \to \alpha$.

Given a congruence $\equiv$ on $\Lambda^\Sigma$, $\Lambda^\Sigma/\equiv$ forms a CCC, the arrows are equivalence classes of $\lambda$-terms.
Given a HOS $\Sigma$, $\Lambda_\Sigma$ (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 $\Gamma$ is empty then $M$ is an arrow $1 \rightarrow \alpha$.

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

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

\[
f_1 \sim_A f_2 \iff \forall C[]. C[f_1] \in A \iff C[f_2] \in A
\]

\( \sim_A \) is a congruence of CCC,
Syntactic CCC of a language

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

$$f_1 \sim_A f_2 \iff \forall C[]. C[f_1] \in A \iff C[f_2] \in A$$

$\sim_A$ 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 $C$ and $A \subseteq \text{Hom} (\beta, \alpha)$ and $f_1, f_2$ in $\text{Hom} (\theta, \delta)$, 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 $C = \Lambda\Sigma$, $\Lambda\Sigma/\sim_A$ is the syntactic CCC associated to the language $A$,
- whenever $\approx$ is a congruence on $C$ and $A = F^{-1}_{\approx}(B)$ for $B \subseteq \text{Hom}_{\Lambda\Sigma/\approx} (\beta, \alpha)$, then there is a surjective functor $G : \Lambda\Sigma/\approx \to \Lambda\Sigma/\sim_A$. 

For the moment we call C-recognizable a language whose syntactic CCC is locally finite (i.e. for every $\alpha, \beta$, $\text{Hom}_{\Lambda\Sigma/\approx} (\alpha, \beta)$ is finite),

conjecture: every language of $\lambda$-terms that has a locally finite syntactic CCC is recognizable.
Syntactic CCC of a language

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

\[ f_1 \sim_A f_2 \iff \forall C[]. C[f_1] \in A \leftrightarrow C[f_2] \in A \]

- $\sim_A$ 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 $B \subseteq \text{Hom}_{\Lambda_{\Sigma}/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_{\Sigma}/\approx \rightarrow \Lambda_{\Sigma}/\sim_A$,
- the syntactic CCC of a recognizable set of $\lambda$-terms is locally finite (i.e. for every $\alpha, \beta$, $\text{Hom}_{\Lambda_{\Sigma}/\sim_R}(\alpha, \beta)$ is finite).
Syntactic CCC of a language

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

$$f_1 \sim_A f_2 \iff \forall C[]. C[f_1] \in A \iff C[f_2] \in A$$

- $\sim_A$ 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 $B \subseteq \text{Hom}_{\Lambda\Sigma/\approx}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda\Sigma/\approx \rightarrow \Lambda\Sigma/\sim_A$,
- the syntactic CCC of a recognizable set of $\lambda$-terms is locally finite (i.e. for every $\alpha$, $\beta$, $\text{Hom}_{\Lambda\Sigma/\approx}(\alpha, \beta)$ is finite),
- conjecture: every language of $\lambda$-terms that has a locally finite syntactic CCC is recognizable.
Syntactic CCC of a language

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

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

- $\sim_A$ is a congruence of CCC,
- when $C = \Lambda_\Sigma$, $\Lambda_\Sigma/\sim_A$ is the syntactic CCC associated to the language $A$,
- whenever $\approx$ is a congruence on $C$ and $A = F_{\approx}^{-1}(B)$ for $B \subseteq \text{Hom}_{\Lambda_\Sigma/\sim}(\beta, \alpha)$, then there is a surjective functor $G : \Lambda_\Sigma/\approx \rightarrow \Lambda_\Sigma/\sim_A$,
- the syntactic CCC of a recognizable set of $\lambda$-terms is locally finite (i.e. for every $\alpha, \beta$, $\text{Hom}_{\Lambda_\Sigma/\sim_R}(\alpha, \beta)$ is finite),
- **conjecture:** every language of $\lambda$-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 $\lambda$-terms representing the elements of $R$:

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R$

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of $\lambda$-terms.
Embedding of syntactic monoid within syntactic CCC

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

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every $M, M/u/ \in R' \iff M/v/ \in R'$

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of $\lambda$-terms.
Embedding of syntactic monoid within syntactic CCC

Given \( R \) a recognizable set of strings, and \( R' \) be the recognizable set of \( \lambda \)-terms representing the elements of \( R \):

\begin{itemize}
  \item \( u \equiv_R v \) iff for every \( w_1, w_2, w_1uw_2 \in R \Leftrightarrow w_1vw_2 \in R \)
  \item \[u\] \( \sim_{R'} \) \[v\] iff for every \( M, M/u/ \in R' \Leftrightarrow M/v/ \in R' \)
  \item or equivalently iff for every \( w_0, \ldots, w_n \), \\
\quad \( w_0uw_1 \ldots w_{n-1}uw_n \in R \Leftrightarrow w_0vw_1 \ldots w_{n-1}vw_n \in R \)
\end{itemize}

This implies:

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

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of \( \lambda \)-terms.
Embedding of syntactic monoid within syntactic CCC

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

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \Leftrightarrow w_1vw_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every $M, M/u/ \in R' \Leftrightarrow M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$,
  $$w_0uw_1 \ldots w_{n-1}uw_n \in R \Leftrightarrow w_0vw_1 \ldots w_{n-1}vw_n \in R$$

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 $\lambda$-terms representing the elements of $R$:

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every $M, M/u/ \in R' \iff M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$,
  
  $w_0uw_1 \ldots w_{n-1}uw_n \in R \iff w_0vw_1 \ldots w_{n-1}vw_n \in R$

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 \( \lambda \)-terms representing the elements of \( R \):

- \( u \equiv_R v \) iff for every \( w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R \)
- \( /u/ \sim_{R'} /v/ \) iff for every \( M, M/u/ \in R' \iff M/v/ \in R' \)
- or equivalently iff for every \( w_0, \ldots, w_n, \)
  \( w_0uw_1 \ldots w_{n-1}uw_n \in R \iff w_0vw_1 \ldots w_{n-1}vw_n \in R \)

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 \( Hom(o,o) \)),

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of \( \lambda \)-terms.
Embedding of syntactic monoid within syntactic CCC

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

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every $M, M/u/ \in R' \iff M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n,$
  $w_0uw_1 \ldots w_{n-1}uw_n \in R \iff w_0vw_1 \ldots w_{n-1}vw_n \in R$

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 $\lambda$-string language that has a locally finite syntactic CCC is a recognizable set of $\lambda$-terms.
Embedding of syntactic monoid within syntactic CCC

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

- $u \equiv_R v$ iff for every $w_1, w_2, w_1uw_2 \in R \iff w_1vw_2 \in R$
- $/u/ \sim_{R'} /v/$ iff for every $M, M/u/ \in R' \iff M/v/ \in R'$
- or equivalently iff for every $w_0, \ldots, w_n$,
  
  $w_0uw_1 \ldots w_{n-1}uw_n \in R \iff w_0vw_1 \ldots w_{n-1}vw_n \in R$

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 $\lambda$-string language that has a locally finite syntactic CCC is a recognizable set of $\lambda$-terms.

Remark: similar results hold for recognizable sets of trees seen as recognizable sets of $\lambda$-terms.
Outline

Recognizable sets of $\lambda$-terms

Recognizability and congruence

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

Varieties of locally finite CCCs
We have syntactic objects that fully characterize languages of \( \lambda \)-terms:

- can we classify these languages in terms of properties of their syntactic CCCs?
Classification of recognizable sets of $\lambda$-terms

We have syntactic objects that fully characterize languages of $\lambda$-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 $\lambda$-terms

We have syntactic objects that fully characterize languages of $\lambda$-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 $\lambda$-terms

We have syntactic objects that fully characterize languages of $\lambda$-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 $\lambda$-terms.
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 in $\mathbf{V}$, 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 $\mathbf{V}$
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 $\Sigma$):

- $\Sigma \mathcal{V}$ 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 $\Sigma$.
- If $f : \Gamma^* \rightarrow \Sigma^*$ 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 $\mathcal{V}$ 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 
  $\{M_R \mid R \in \Sigma \mathcal{V} \text{ for some } \Sigma\}$,
- $\tilde{\mathcal{V}}$ 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
  \[ \{ M_R \mid R \in \Sigma \mathcal{V} \text{ for some } \Sigma \}, \]
- $\tilde{\mathcal{V}}$ be the class of languages
  \[ \{ R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma \}. \]

We then have:

- $\tilde{\mathcal{V}}$ 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:

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

We then have:

- $\tilde{\mathcal{V}}$ is a variety of languages,
- $\mathcal{V} = \tilde{\mathcal{V}},$
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 
  \[ \{ M_R \mid R \in \Sigma \mathcal{V} \text{ for some } \Sigma \}, \]
- $\tilde{\mathcal{V}}$ be the class of languages 
  \[ \{ R \mid R \subseteq \Sigma^*, M_R \in \mathbf{V} \text{ for some } \Sigma \}. \]

We then have:
- $\tilde{\mathcal{V}}$ is a variety of languages,
- $\mathcal{V} = \tilde{\mathcal{V}}$,
- $\mathbf{V} = \overline{\mathcal{V}}$
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

Recognizable sets of $\lambda$-terms

Recognizability and congruence

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

Varieties of locally finite CCCs
Finitely generated CCCs

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

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.
Finitely generated CCCs

A CCC $\mathcal{C}$ is finitely generated if there is HOS $\Sigma$ 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 $\Sigma$ 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 $C_1$ and $C_2$ 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 $C_1 \times C_2$ with:

- the objects of $C_1 \times C_2$ is the same as the ones of $C_1$ and $C_2$,
Product of CCCs

Given $C_1$ and $C_2$ 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 $C_1 \times C_2$ with:

- the objects of $C_1 \times C_2$ is the same as the ones of $C_1$ and $C_2$,
- $\text{Hom}_{C_1 \times C_2}(\alpha, \beta) = \text{Hom}_{C_1}(\alpha, \beta) \times \text{Hom}_{C_2}(\alpha, \beta)$
Product of CCCs

Given $C_1$ and $C_2$ 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 $C_1 \times C_2$ with:

- the objects of $C_1 \times C_2$ is the same as the ones of $C_1$ and $C_2$,
- $\text{Hom}_{C_1 \times C_2}(\alpha, \beta) = \text{Hom}_{C_1}(\alpha, \beta) \times \text{Hom}_{C_2}(\alpha, \beta)$
- $C_1 \times C_2$, is a locally finite CCC,
Product of CCCs

Given $C_1$ and $C_2$ 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 $C_1 \times C_2$ with:

- the objects of $C_1 \times C_2$ is the same as the ones of $C_1$ and $C_2$,
- $\text{Hom}_{C_1 \times C_2} (\alpha, \beta) = \text{Hom}_{C_1} (\alpha, \beta) \times \text{Hom}_{C_2} (\alpha, \beta)$
- $C_1 \times C_2$, is a locally finite CCC,
- but $C_1 \times C_2$ may not be finitely generated...
Product of CCCs

Given $C_1$ and $C_2$ 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 $C_1 \times C_2$ with:

- the objects of $C_1 \times C_2$ is the same as the ones of $C_1$ and $C_2$,
- $\text{Hom}_{C_1 \times C_2}(\alpha, \beta) = \text{Hom}_{C_1}(\alpha, \beta) \times \text{Hom}_{C_2}(\alpha, \beta)$
- $C_1 \times C_2$, is a locally finite CCC,
- but $C_1 \times C_2$ may not be finitely generated...

Thus given two finite presentation $F_1$ and $F_2$ of $C_1$ and $C_2$, we define $C_1 \times F_1, F_2 C_2$ to be the sub-CCC of $C_1 \times C_2$ generated by the arrows:

$$
\bigcup_{c \in \Sigma_1} \{F_1(c)\} \times \text{Hom}_{C_2}(1, \tau_1(c)) \cup \bigcup_{c \in \Sigma_2} \text{Hom}_{C_1}(1, \tau_2(c)) \times \{F_2(c)\}
$$
Direct product of monoids and product of CCCs

Given $C_1$ and $C_2$ 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 $C_1$ and $C_2$ we have

$$C_1 \times_{F_1,F_2} C_2 = C_1 \times_{G_1,G_2} C_2$$
Given $C_1$ and $C_2$ 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 $C_1$ and $C_2$ we have

  \[ C_1 \times F_1, F_2 \, C_2 = C_1 \times G_1, G_2 \, C_2 \]

- a question is whether for every locally finite and finitely generated CCC, $C_1$ and $C_2$ we can find a canonical sub-CCC of $C_1 \times C_2$ that is finitely generated.
α-syntactic and α-separated CCC

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

$$f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2$$
α-syntactic and α-separated CCC

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

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

We then have:

- if $C_1, C_2$ are two α-syntactic CCC, then for every presentation $F_1$ and $F_2$ of $C_1$ and $C_2$, $C_1 \times_{F_1,F_2} C_2$ is α-syntactic.
\(\alpha\)-syntactic and \(\alpha\)-separated CCC

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

\[ f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2 \]

We then have:

- if \(\mathcal{C}_1, \mathcal{C}_2\) are two \(\alpha\)-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 \(\alpha\)-syntactic.

- it can be the case that a locally finite finitely generated CCC \(\mathcal{C}\) can not be constructed from \(\alpha\)-syntactic CCCs using product, sub-CCC and quotient.
\(\alpha\text{-syntactic and } \alpha\text{-separated CCC}

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

\[f_1 \sim_A f_2 \text{ if and only if } f_1 = f_2\]

We then have:

- if \(C_1, C_2\) are two \(\alpha\)-syntactic CCC, then for every presentation \(F_1\) and \(F_2\) of \(C_1\) and \(C_2\), \(C_1 \times_{F_1,F_2} C_2\) is \(\alpha\)-syntactic.
- it can be the case that a locally finite finitely generated CCC \(C\) can not be constructed from \(\alpha\)-syntactic CCCs using product, sub-CCC and quotient.
- but this is the case when \(C\) is \(\alpha\)-separated:
  - for every \(f_1, f_2\) in \(\text{Hom}(\theta, \delta)\), \(f_1 \neq f_2\) iff there is \(C[]\) such that \(C[f_1], C[f_2]\) are in \(\text{Hom}(1, \alpha)\) and \(C[f_1] \neq C[f_2]\).
Varieties of locally finite CCC

A variety of locally finite CCC $\mathbf{V}$ 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 $(C, \alpha)$ such that:

- $C$ is a locally finite and finitely generated CCC
- $\alpha$ is an object of $C$ and $C$ is $\alpha$-separated
Varieties of locally finite CCC

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

- $C$ is a locally finite and finitely generated CCC
- $\alpha$ is an object of $C$ and $C$ is $\alpha$-separated
- for every $(C_1, \alpha)$ and $(C_2, \alpha)$, and every presentation of $F_1$ and $F_2$ of $C_1$ and $C_2$, $(C_1 \times_{F_1,F_2} C_2, \alpha)$ is in $\mathbf{V}$
Varieties of locally finite CCC

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

- $\mathcal{C}$ is a locally finite and finitely generated CCC
- $\alpha$ is an object of $\mathcal{C}$ and $\mathcal{C}$ is $\alpha$-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 $\mathcal{C}_2$, $(\mathcal{C}_1 \times_{F_1,F_2} \mathcal{C}_2, \alpha)$ is in $\mathbf{V}$
- if $(\mathcal{C}, \alpha)$ is in $\mathbf{V}$ and $\mathcal{C}'$ is a sub-CCC, then if $\mathcal{C}''$ is the $\beta$-separated CCC obtained from $\mathcal{C}'$, $(\mathcal{C}'', \beta)$ is in $\mathbf{V}$
Varieties of locally finite CCC

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

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

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

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

- $C$ is a locally finite and finitely generated CCC
- $\alpha$ is an object of $C$ and $C$ is $\alpha$-separated
- for every $(C_1, \alpha)$ and $(C_2, \alpha)$, and every presentation of $F_1$ and $F_2$ of $C_1$ and $C_2$, $(C_1 \times_{F_1,F_2} C_2, \alpha)$ is in $V$
- if $(C, \alpha)$ is in $V$ and $C'$ is a sub-CCC, then if $C''$ is the $\beta$-separated CCC obtained from $C'$, $(C'', \beta)$ is in $V$
- if $(C, \alpha)$ is in $V$, $\sim$ is a congruence of $C$, and $C'$ is the $\beta$-separated CCC obtained from $C/\sim$ then $(C', \beta)$ is in $V$.

we write $(C_1, \beta) \prec (C_2, \alpha)$ when $C_1$ is an $\beta$-separated CCC obtained by taking and quotienting a sub-CCC of $C_2$. 
Towards varieties of languages

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

- $C/\sim_A = C/\sim_B$ with $B = \text{Hom}(1, \alpha) - A$
Towards varieties of languages

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

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

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

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

- $(C/\sim_{A \cap A'}, \alpha) \prec (C/\sim_A \times_{F, F} C/\sim_{A'}, \alpha)$ for every presentation $F$ of $C$,

- Given $C[]$ such that for every $f \in \text{Hom}(1, \beta)$, $C[f]$ is in $\text{Hom}(1, \alpha)$, if $C^{-1}[A] = \{f \in \text{Hom}(1, \beta) \mid C[f] \in A\}$ then $C/\sim_{C^{-1}[A]}$ is a quotient CCC of $C/\sim_A$
Towards varieties of languages

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

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

2. $(C/\sim_{A \cap A'}, \alpha) \prec (C/\sim_A \times_F F C/\sim_{A'}, \alpha)$ for every presentation $F$ of $C$,

3. Given $C[]$ such that for every $f \in \text{Hom}(1, \beta)$, $C[f]$ is in $\text{Hom}(1, \alpha)$, if $C^{-1}[A] = \{ f \in \text{Hom}(1, \beta) \mid C[f] \in A \}$ then $C/\sim_{C^{-1}[A]}$ is a quotient CCC of $C/\sim_A$

Given a CCC-functor $F : D \to C$ and $\beta$ such that $F(\beta) = \alpha$, and $B = F^{-1}(A) \cap \text{Hom}_D(1, \beta)$ then $(D/\sim_B, \beta) \prec (C/\sim_A, \alpha)$. 
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 $\Sigma$ whose elements have type $\alpha$):

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

2. Given $F : \Lambda_{\Sigma 1} \rightarrow \Lambda_{\Sigma 2}$ a CCC-functor, if $R \in (\Sigma_2, \alpha)\mathcal{V}$ and $F(\beta) = \alpha$, then $F^{-1}(R) \cap \Lambda_{\beta \Sigma 1} \in (\Sigma_1, \beta)\mathcal{V}$. 

Varieties of $\lambda$-languages

A variety of C-recognizable sets of $\lambda$-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 $\Sigma$ whose elements have type $\alpha$):

- $(\Sigma, \alpha)\mathcal{V}$ is closed under Boolean operations
- Given $M \in \Lambda_{\Sigma}^{\beta \rightarrow \alpha}$, 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 $\lambda$-languages

A variety of $C$-recognizable sets of $\lambda$-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 $\Sigma$ whose elements have type $\alpha)$:

- $(\Sigma, \alpha)\mathcal{V}$ is closed under Boolean operations
- Given $M \in \Lambda_{\Sigma}^{\beta \rightarrow \alpha}$, and $R$ in $(\Sigma, \alpha)\mathcal{V}$, then $M^{-1}R = \{ N \in \Lambda_{\Sigma}^{\beta} | MN \in R \}$ is in $(\Sigma, \beta)\mathcal{V}$,
- Given $F : \Lambda_{\Sigma_1} \rightarrow \Lambda_{\Sigma_2}$ 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 \( \lambda \)-terms \( R \), we let \( C_R \) be its syntactic CCC.

Given \( \mathcal{V} \) a variety of \( \lambda \)-languages and \( \mathbf{V} \) a variety of locally finite CCCs we let:

- \( \mathcal{V} \) be the variety of locally finite CCC generated by 
  \( \{(C_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\} \),
The correspondence

Given a $C$-recognizable set of $\lambda$-terms $R$, we let $C_R$ be its syntactic CCC.

Given $\mathcal{V}$ a variety of $\lambda$-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

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

We then have:

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

The correspondence

Given a C-recognizable set of λ-terms $R$, we let $C_R$ be its syntactic CCC.

Given $\mathcal{V}$ a variety of λ-languages and $\mathbf{V}$ a variety of locally finite CCCs we let:

- $\widetilde{\mathcal{V}}$ be the variety of locally finite CCC generated by
  \[ \{(C_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\}, \]

- $\widetilde{\mathbf{V}}$ be the class of languages
  \[ \{R \mid R \subseteq \Lambda_{\Sigma}^\alpha, (C_R, \alpha) \in \mathbf{V} \text{ for some } \Sigma \text{ and } \alpha\}. \]

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 $C_R$ be its syntactic CCC.

Given $\mathcal{V}$ a variety of λ-languages and $\mathcal{V}$ a variety of locally finite CCCs we let:

- $\widehat{\mathcal{V}}$ be the variety of locally finite CCC generated by
  \[\{(C_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\},\]
- $\mathfrak{V}$ be the class of languages
  \[\{R \mid R \subseteq \Lambda_\Sigma^\alpha, (C_R, \alpha) \in \mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\}.

We then have:

- $\mathfrak{V}$ is a variety of λ-languages,
- $\mathcal{V} = \widehat{\mathcal{V}}$,
The correspondence

Given a C-recognizable set of λ-terms $R$, we let $C_R$ be its syntactic CCC.

Given $\mathcal{V}$ a variety of λ-languages and $\mathcal{V}$ a variety of locally finite CCCs we let:

- $\bar{\mathcal{V}}$ be the variety of locally finite CCC generated by
  \[ \{(C_R, \alpha) \mid R \in (\Sigma, \alpha)\mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\}, \]
- $\tilde{\mathcal{V}}$ be the class of languages
  \[ \{R \mid R \subseteq \Lambda_\Sigma^\alpha, (C_R, \alpha) \in \mathcal{V} \text{ for some } \Sigma \text{ and } \alpha\}. \]

We then have:

- $\tilde{\mathcal{V}}$ is a variety of λ-languages,
- $\mathcal{V} = \tilde{\mathcal{V}}$,
- $\bar{\mathcal{V}} = \bar{\mathcal{V}}$,
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 ≺
  - 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 $\lambda$-terms that are neither $\lambda$-strings nor $\lambda$-trees rely on the conjecture recognizable $=$ C-recognizable.