Finitary semantics and regular languages of λ-terms

Vincent Moreau, joint work with Tito Nguyễn

CSL 2024

Friday the 23rd of February, 2024

IRIF, Université Paris Cité, Inria Paris

Introduction & motivations

The Church encoding

The type of words over a two-letter alphabet $\{a, b\}$ is

$$\mathrm{Church}_{\{a,b\}} \quad := \quad \underbrace{(\mathbb{o} \Rightarrow \mathbb{o})}_{a} \Rightarrow \underbrace{(\mathbb{o} \Rightarrow \mathbb{o})}_{b} \Rightarrow \underbrace{\mathbb{o}}_{\text{input}} \Rightarrow \underbrace{\mathbb{o}}_{\text{output}} \ .$$

Any finite word can be encoded as a λ-term through the Church encoding:

$$abb \in \{a, b\}^* \quad \rightsquigarrow \quad \lambda(a : \mathbb{o} \Rightarrow \mathbb{o}).\lambda(b : \mathbb{o} \Rightarrow \mathbb{o}).\lambda(e : \mathbb{o}).\, b\,(b\,(a\,e)) \ .$$

$\rightarrow$ **The simply typed λ-calculus generalizes finite words.**

Languages of λ-terms: the semantic side

If Q is a finite set, then any $M \in \Lambda(\mathrm{Church}_{\{a,b\}})$ can be interpreted as

$$[\![M]\!]_Q \quad \in \quad (Q \Rightarrow Q) \Rightarrow (Q \Rightarrow Q) \Rightarrow Q \Rightarrow Q \;=\; [\![\mathrm{Church}_{\{a,b\}}]\!]_Q \;.$$

For all $\delta_a : Q \to Q$, $\delta_b : Q \to Q$ and $q_0 \in Q$, then

$$[\![\lambda a.\lambda b.\lambda c.\, b\,(b\,(a\,c))]\!]_Q(\delta_a, \delta_b, q_0) \quad = \quad \delta_b(\delta_b(\delta_a(q_0)))\;,$$

$\to$ **Semantics of λ-calculus in finite sets generalize the interpretation in DFAs.**

Language λ-terms: the syntactic side

We consider the type

$$\mathrm{Bool} \quad := \quad \mathfrak{o} \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o}$$

whose only inhabitants, up to $\beta\eta$-conversion, are

$$\mathrm{true} \ := \ \lambda x.\lambda y.x \quad \text{and} \quad \mathrm{false} \ := \ \lambda x.\lambda y.y \ .$$

Hillebrand and Kanellakis have shown how an automaton can be encoded as a λ-term

$$R \quad \in \quad \Lambda(\underbrace{((B \Rightarrow B) \Rightarrow (B \Rightarrow B) \Rightarrow B \Rightarrow B)}_{\mathrm{Church}_{\{a,b\}} \text{ with } \mathfrak{o} \text{ replaced by } B} \Rightarrow \mathrm{Bool})$$

for some simple type B representing the set of finite states.

$\rightarrow$ **Regular languages can be recovered syntactically from the λ-calculus.**

This work

→ **We show that semantic and syntactic languages of λ-terms coincide.**

We can reason by proving the three following implications:

$$\begin{tikzcd}
 & \text{Regular for some finitary CCC} \arrow[dr] & \\
\text{Syntactically regular} \arrow[ur] & & \text{Regular for } \textbf{FinSet} \arrow[ll, "\text{We will see this!}"']
\end{tikzcd}$$

where fintary = locally finite and well-pointed.

To achieve this, we will crucially use a new technique called **squeezing**.

Languages of λ-terms

Cartesian closed categories

A CCC is a category $\mathbf{C}$ which has cartesian products, i.e. objects $a \times b$ such that

$$\text{a morphism } c \to a \times b$$
$$\cong$$
$$\text{two morphisms } c \to a \text{ and } c \to b$$

with a terminal object 1, and which has exponentials, i.e. objects $a \Rightarrow b$ such that

$$\text{a morphism } a \times c \to b$$
$$\cong$$
$$\text{a morphism } c \to a \Rightarrow b.$$

$\to$ **We can interpret the simply typed λ-calculus.**

Interpretation: the universal property of λ-terms

The category **Lam**, whose objects are types and where

$$\text{a morphism } A \to B \qquad \text{is} \qquad \text{a } \lambda\text{-terms } t \in \Lambda(A \Rightarrow B)$$

is the free CCC on one object $\mathbb{O}$.

This means that, for every object c of $\mathbf{C}$, there exists a unique CCC functor

$$\begin{tikzcd} \mathbf{Lam} \arrow[dr, dashed, "{[\![-]\!]_c}"] & \\ 1 \arrow[u, "\mathbb{O}"] \arrow[r, "c"'] & \mathbf{C} \end{tikzcd} \quad .$$

Its action on morphisms restricts to a function on closed λ-terms

$$[\![-]\!]_c \quad : \quad \underbrace{\mathbf{Lam}(1, A)}_{\cong\, \Lambda(A)} \longrightarrow \mathbf{C}(1, [\![A]\!]_c) \ .$$

Semantic languages of λ-terms

For **words**, a morphism $\varphi : \Sigma^* \to M$ into a finite monoid and $F \subseteq M$ induce

$$L_F \quad := \quad \{w \in \Sigma^* \mid \varphi(w) \in F\} \ .$$

The notion of regular language of λ-terms has been introduced by Salvati.

For λ-**terms** of type A, any object c of $\mathbf{C}$ and $F \subseteq \mathbf{C}(1, [\![A]\!]_c)$ induce

$$L_F \quad := \quad \{M \in \Lambda(A) \mid [\![M]\!]_c \in F\} \ .$$

Definition. A language of λ-terms is **recognized by C** if it is of the form L_F.

Interpreting in the CCC of finite sets yields the deterministic automata semantics.

Syntactic languages of λ-terms

When we take $\mathbf{C} = \mathbf{Lam}$ itself, any type B gives a CCC functor

$$\begin{tikzcd} \mathbf{Lam} \arrow[dr, dashed, "(-)[B]"] & \\ 1 \arrow[u, "\mathbb{o}"] \arrow[r, "B"'] & \mathbf{Lam} \end{tikzcd}$$

For example, it assigns:

$$\begin{array}{ccc} (\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o} \Rightarrow \mathbb{o} & \rightsquigarrow & (B \Rightarrow B) \Rightarrow B \Rightarrow B \\ \lambda(f : \mathbb{o} \Rightarrow \mathbb{o}).\lambda(x : \mathbb{o}).f\,(f\,x) & \rightsquigarrow & \lambda(f : B \Rightarrow B).\lambda(x : B).f\,(f\,x) \end{array}$$

Any λ-term $R \in \Lambda(A[B] \Rightarrow \mathrm{Bool})$ induces the language of type A defined as

$$L_r \quad := \quad \{M \in \Lambda(A) \mid R\ M[B] \ =_{\beta\eta}\ \mathrm{true}\}\ .$$

Definition. A language of λ-terms is **syntactically regular** if it is of the form L_r.

Squeezing and logical relations

Squeezing structure

Definition. A squeezing structure on a CCC $\mathbf{C}$ is the data of two wide subcategories $\mathbf{C}_{\text{left}}$ and $\mathbf{C}_{\text{right}}$ of $\mathbf{C}$, with associated notations $\xrightarrow{\text{l}}$ and $\xrightarrow{\text{r}}$, such that

- $\mathbf{C}_{\text{left}}$ and $\mathbf{C}_{\text{right}}$ are stable under finite cartesian products and exponentials[1]
- for every object c of $\mathbf{C}$, there exists two objects L_c and R_c of $\mathbf{C}$ together with morphisms

$$L_1 \xrightarrow{\text{l}} 1 \qquad L_{c\times c'} \xrightarrow{\text{l}} L_c \times L_{c'} \qquad L_{c\Rightarrow c'} \xrightarrow{\text{l}} R_c \Rightarrow L_{c'}$$
$$1 \xrightarrow{\text{r}} R_1 \qquad R_c \times R_{c'} \xrightarrow{\text{r}} R_{c\times c'} \qquad L_c \Rightarrow R_{c'} \xrightarrow{\text{r}} R_{c\Rightarrow c'} \ .$$

[1] following the right polarities

The squeezing category

If $\mathbf{C}$ comes with a squeezing structure, we write $\mathbf{Sqz}(\mathbf{C})$ for the full subcategory of objects c such that there exists morphisms

$$L_c \xrightarrow{\mathsf{l}} c \qquad \text{and} \qquad c \xrightarrow{\mathsf{r}} R_c \ .$$

Then, $\mathbf{Sqz}(\mathbf{C})$ is a sub-CCC of $\mathbf{C}$, so for every type A, there exists morphisms

$$L_{[\![A]\!]_c} \xrightarrow{\mathsf{l}} [\![A]\!]_c \qquad \text{and} \qquad [\![A]\!]_c \xrightarrow{\mathsf{r}} R_{[\![A]\!]_c} \ .$$

This is related to normalization by evaluation and Tait's yoga.

Logical relations, i.e. sconing

For $\mathbf{C}$ and $\mathbf{D}$ two CCCs, the category $\mathbf{Log}(\mathbf{C}, \mathbf{D})$ of logical relations has as objects

$$\text{the tuples } (c, d, \Vdash) \qquad \text{where } \Vdash \subseteq \mathbf{C}(1, c) \times \mathbf{D}(1, d)$$

and as morphisms from $(c, d, \Vdash)$ to $(c', d', \Vdash')$ the pairs (f, g) such that for all x, y,

$$\text{if } x \Vdash y, \quad \text{then } f \circ x \Vdash' g \circ y. \ .$$

The assignement $(c, d, \Vdash) \mapsto (c, d)$ respects the CCC structure, which gives back the lemma of logical relation: for any type A and λ-term $M \in \Lambda(A)$,

$$(\, [\![M]\!]_c \,,\, [\![M]\!]_d \,) \qquad \in \qquad [\![A]\!]_{(c,d,\Vdash)} \subseteq [\![A]\!]_c \times [\![A]\!]_d \ .$$

Logical relations, like double categories (1/2)

We represent a relation $(c, d, \Vdash)$ as

$$\begin{tikzcd} c \arrow[r, "\Vdash", "\shortmid"{marking}] & d \end{tikzcd} \quad .$$

We represent a morphism (f, g) from $(c, d, \Vdash)$ to $(c', d', \Vdash')$ as a square

$$\begin{tikzcd} c \arrow[r, "\Vdash", "\shortmid"{marking}] \arrow[d, "f"'] \arrow[dr, phantom, "\Downarrow"] & d \arrow[d, "g"] \\ c' \arrow[r, "\Vdash"', "\shortmid"{marking}] & d' \end{tikzcd} \quad .$$

The categorical structure of $\mathbf{Log}(\mathbf{C}, \mathbf{D})$ amounts to compose squares vertically.

Logical relations, like double categories (2/2)

If we have a relation

$$\begin{tikzcd} c \arrow[r, "\Vdash", "\shortmid"{marking}] & d \end{tikzcd}$$

then any type A, we have the relation

$$\begin{tikzcd} \llbracket A \rrbracket_c \arrow[r, "\llbracket A \rrbracket_{\Vdash}", "\shortmid"{marking}] & \llbracket A \rrbracket_d \end{tikzcd}$$

and for every λ-term $M \in \Lambda(A)$, we have a morphism

$$\begin{tikzcd} 1 \arrow[r, "\shortmid"{marking}] \arrow[d, "\llbracket M \rrbracket_c"'] \arrow[dr, phantom, "\Downarrow"] & 1 \arrow[d, "\llbracket M \rrbracket_d"] \\ \llbracket A \rrbracket_c \arrow[r, "\llbracket A \rrbracket_{\Vdash}"', "\shortmid"{marking}] & \llbracket A \rrbracket_d \end{tikzcd}$$

which is exactly the fundamental lemma of logical relations.

Types, finite sets and their squeezing structure

We consider the category $\mathbf{Log}(\mathbf{Lam}, \mathbf{FinSet})$, whose objects are tuples $(B, Q, \Vdash)$.

For every finite set Q, there is a bijection

$$\mathbb{O}^Q \Rightarrow \mathbb{O} \xrightarrow[\vdash]{\sim_Q} Q$$

and we let $L_{(B,Q,\Vdash)}$ and $R_{(B,Q,\Vdash)}$ be this bijection.

We define left and right morphisms to be the squares with identities on the side of sets. By squeezing, we thus have that, for every type A, there exists morphisms

$$\begin{tikzcd}
\mathbb{O}^{[\![A]\!]_Q} \Rightarrow \mathbb{O} \arrow[r, "\sim_{[\![A]\!]_Q}", "\vdash"'] \arrow[d] \arrow[dr, phantom, "\Downarrow"] & {[\![A]\!]_Q} \arrow[d, equal, "\mathrm{Id}"] \\
A[\mathbb{O}^Q \Rightarrow \mathbb{O}] \arrow[r, "{[\![A]\!]_{\sim_Q}}", "\vdash"'] & {[\![A]\!]_Q}
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
A[\mathbb{O}^Q \Rightarrow \mathbb{O}] \arrow[r, "{[\![A]\!]_{\sim_Q}}", "\vdash"'] \arrow[d] \arrow[dr, phantom, "\Downarrow"] & {[\![A]\!]_Q} \arrow[d, equal, "\mathrm{Id}"] \\
\mathbb{O}^{[\![A]\!]_Q} \Rightarrow \mathbb{O} \arrow[r, "\sim_{[\![A]\!]_Q}", "\vdash"'] & {[\![A]\!]_Q}
\end{tikzcd}$$

Proving the theorem

Let $F \subseteq [\![A]\!]_Q$ represented by $\chi : [\![A]\!]_Q \to 2$. For every $M \in \Lambda(A)$, we have

$$\begin{tikzcd}
1 \arrow[r, "|"] \arrow[d, "M[\circ^Q \Rightarrow \circ]"'] \arrow[dr, phantom, "\Downarrow"] & 1 \arrow[d, "{[\![M]\!]_Q}"] \\
A[\circ^Q \Rightarrow \circ] \arrow[r, "|", "{[\![A]\!]_{\sim_Q}}"'] & {[\![A]\!]_Q}
\end{tikzcd}
\qquad
\begin{tikzcd}
A[\circ^Q \Rightarrow \circ] \arrow[r, "|", "{[\![A]\!]_{\sim_Q}}"] \arrow[d] \arrow[dr, phantom, "\Downarrow"] & {[\![A]\!]_Q} \arrow[d, equal, "\mathsf{Id}"] \\
\circ^{[\![A]\!]_Q} \Rightarrow \circ \arrow[r, "|", "{\sim_{[\![A]\!]_Q}}"'] & {[\![A]\!]_Q}
\end{tikzcd}
\qquad
\begin{tikzcd}
\circ^{[\![A]\!]_Q} \Rightarrow \circ \arrow[r, "|", "{\sim_{[\![A]\!]_Q}}"] \arrow[d, "\circ^\chi \Rightarrow \circ"'] \arrow[dr, phantom, "\Downarrow"] & {[\![A]\!]_Q} \arrow[d, "\chi"] \\
\mathsf{Bool} \arrow[r, "|", "\sim_2"'] & 2
\end{tikzcd}$$

which, when composed together, yield R such that

$$\begin{tikzcd}
1 \arrow[r, "|"] \arrow[d, "R\ M[\circ^Q \Rightarrow \circ]"'] \arrow[dr, phantom, "\Downarrow"] & 1 \arrow[d, "\chi([\![M]\!]_Q)"] \\
\mathsf{Bool} \arrow[r, "|", "\sim_2"'] & 2
\end{tikzcd} \cdot$$

In other words:

$$R\ M[\circ^Q \Rightarrow \circ] \ = \ \mathsf{true} \qquad \text{if and only if} \qquad [\![M]\!]_Q \ \in \ F \qquad \square$$

Conclusion

Future work:

- Finitary intensional models of the simply typed λ-calculus, e.g. sequential algorithms, some qualitative models of linear logic.
- Study different calculi, e.g. linear, polymorphic.
- Study what kind of conditions can be encoded as regular languages of higher-order terms.

Conclusion

Future work:

- Finitary intensional models of the simply typed λ-calculus, e.g. sequential algorithms, some qualitative models of linear logic.
- Study different calculi, e.g. linear, polymorphic.
- Study what kind of conditions can be encoded as regular languages of higher-order terms.

Thank you for your attention!

Any questions?

Bibliography

[HK96] Gerd G. Hillebrand and Paris C. Kanellakis. **"On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi".** In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 253–263. DOI: 10.1109/LICS.1996.561337.

[Mel17] Paul-André Melliès. **"Higher-order parity automata".** In: *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, 2017*. 2017, pp. 1–12.

[Sal09] Sylvain Salvati. **"Recognizability in the Simply Typed Lambda-Calculus".** In: *16th Workshop on Logic, Language, Information and Computation*. Vol. 5514. Lecture Notes in Computer Science. Tokyo Japan: Springer, 2009, pp. 48–60.