Finitary semantics and regular languages of λ-terms

Vincent Moreau, joint work with Tito Nguyễn

GT SCALP

November the 27^{th}, 2023

IRIF, Université Paris Cité, Inria Paris

Introduction & motivations

The Church encoding

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

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

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

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

More generally, any finite ranked tree can be encoded as a λ-term:

$$\begin{tikzcd} & e & e \\ a \arrow[r] \arrow[ur] & a \arrow[r] \arrow[ur] & e \end{tikzcd} \quad \rightsquigarrow \quad \lambda(a : \obar \Rightarrow \obar \Rightarrow \obar).\lambda(e : \obar).a\,e\,(a\,e\,e)$$

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

Languages of λ-terms: the semantic side

If Q is a finite set, then any $t \in \Lambda_{\beta\eta}\langle \mathrm{Church}_{\{a,b\}} \rangle$ can be interpreted as

$$[\![t]\!]_Q \quad \in \quad (Q \Rightarrow Q) \Rightarrow (Q \Rightarrow Q) \Rightarrow Q \Rightarrow Q \ .$$

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

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

so interpreting the encoding of a word amounts to running an automaton over it. The same observation holds for finite ranked trees.

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

Language λ-terms: the syntactic side

We consider the type

$$\mathsf{Bool} \quad := \quad o \Rightarrow o \Rightarrow o$$

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

$$\mathsf{true} \ := \ \lambda(x : o).\lambda(y : o).x \quad \text{and} \quad \mathsf{false} \ := \ \lambda(x : o).\lambda(y : o).y \ .$$

An automaton can be encoded as a λ-term

$$r \quad \in \quad \Lambda_{\beta\eta}\langle((B \Rightarrow B) \Rightarrow (B \Rightarrow B) \Rightarrow B \Rightarrow B) \Rightarrow \mathsf{Bool}\rangle$$

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

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

This work

Both semantic and syntactic languages of λ-terms yield back regular languages for the simple type of finite words.

$\rightarrow$ **We show that semantic and syntactic languages of λ-terms coincide at every simple type, demonstrating its robustness.**

We can reason by proving the three following implications:

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

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

Languages of λ-terms

The universal property of λ-terms

The category **Lam** has as objects the simple types built on the base type ϕ and as morphisms from A to B the λ-terms of type $A \Rightarrow B$. It is the free CCC on one object:

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

for any CCC $\mathbf{C}$ and object c.

This unique CCC functor $[\![-]\!]_c : \mathbf{Lam} \to \mathbf{C}$ verifies the following equalities:

$$\begin{aligned} [\![A \Rightarrow B]\!]_c &= [\![A]\!]_c \Rightarrow [\![B]\!]_c & \qquad [\![\phi]\!]_c &= c \\ [\![A \times B]\!]_c &= [\![A]\!]_c \times [\![B]\!]_c & \qquad [\![1]\!]_c &= 1 \end{aligned}$$

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

$$[\![-]\!]_c \quad : \quad \underbrace{\Lambda_{\beta\eta}\langle A\rangle}_{=\mathbf{Lam}(1,A)} \quad \longrightarrow \quad \mathbf{C}(1, [\![A]\!]_c) \ .$$

Semantic languages of λ-terms

In the case of words, any homomorphism $\varphi : \Sigma^* \to M$ into a finite monoid, together with a subset $F \subseteq M$, induces the regular language of finite words

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

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

Let A be a simple type. For any object c and any subset $F \subseteq \mathbf{C}(1, [\![A]\!]_c)$, we define

$$L_F \quad := \quad \{t \in \Lambda_{\beta\eta}\langle A\rangle \mid [\![t]\!]_c \in F\} \ .$$

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

Interpreting in **FinSet** yields the deterministic automata semantics.

Syntactic languages of λ-terms

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

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

If A is a simple type, then $A[B]$ is the substitution of B for o in A. On λ-terms,

$$(\lambda(x : A).t)\,[B] \;=\; \lambda(x : A[B]).t[B] \qquad (t\;u)\,[B] \;=\; t[B]\;u[B] \qquad x[B] \;=\; x$$

For any simple type B, any λ-term $r : A[B] \Rightarrow \mathrm{Bool}$ induces a language

$$L_r \quad := \quad \{t \in \Lambda_{\beta\eta}\langle A\rangle \mid r\;\,t[B] \;=_{\beta\eta}\; \mathrm{true}\}\;.$$

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

Logical relations and squeezing

Sconing in a nutshell: the unary case

Definition. Let $\mathbf{C}$ be a CCC. The category $\mathrm{P}(\mathbf{C})$ of logical predicates has

- as objects the pairs (c, X) where $X \subseteq \mathbf{C}(1, c)$,
- as morphisms from (c, X) to (d, Y) the $f \in \mathbf{C}(c, d)$ such that for all $x \in \mathbf{C}(1, c)$,

$$\text{if } x \in X, \quad \text{then } f \circ x \in Y \ .$$

Then, $\mathrm{P}(C)$ is a CCC and the forgetful functor $\mathrm{P}(\mathbf{C}) \to \mathbf{C}$ respects the CCC structure.

$$\begin{tikzcd}
\mathbf{Lam} \arrow[dr, "{[\![-]\!]_{(c,X)}}"] \arrow[drr, bend left, "{[\![-]\!]_c}"] & & \\
1 \arrow[u, "\Phi"] \arrow[r, "{(c,X)}"'] & \mathrm{P}(\mathbf{C}) \arrow[r] & \mathbf{C}
\end{tikzcd}$$

commutes for any object (c, X) of $\mathrm{P}(\mathbf{C})$.

More concretely, if A is any simple type and (c, X) is in $\mathrm{P}(\mathbf{C})$, then

$$[\![A]\!]_{(c,X)} \quad = \quad ([\![A]\!]_c, X^A) \qquad \text{for some } X^A \subseteq \mathbf{C}(1, [\![A]\!]_c) \ .$$

Sconing in a nutshell: the binary case

Property. Let $\mathbf{C}$ and $\mathbf{D}$ be CCCs. The category $\mathrm{P}(\mathbf{C} \times \mathbf{D})$ has

- as objects the triples $(c, d, \Vdash)$ where $\Vdash \subseteq \mathbf{C}(1, c) \times \mathbf{D}(1, d)$,
- as morphisms from $(c, d, \Vdash)$ to $(c', d', \Vdash')$ the pairs (f, g) which are parametric:

$$\text{for all } x \in \mathbf{C}(1, c) \text{ and } y \in \mathbf{D}(1, d), \qquad \text{if } x \Vdash y, \quad \text{then } f \circ x \Vdash' g \circ y. \; .$$

The same universal property of **Lam** gives directly that if A is a simple type, then

$$[\![A]\!]_{(c,d,\Vdash)} \quad = \quad ([\![A]\!]_c, [\![A]\!]_d, \Vdash^A) \quad \text{for some } \; \Vdash^A \subseteq \; \mathbf{C}(1, [\![A]\!]_c) \times \mathbf{D}(1, [\![A]\!]_d)$$

and we have a lemma of logical relations: for all $t \in \Lambda_{\beta\eta}\langle A \rangle$,

$$[\![t]\!]_c \quad \Vdash^A \quad [\![t]\!]_d \; .$$

Squeezing structure

Definition. A squeezing structure on a CCC $\mathbf{C}$ is the data of

- two wide subcategories $\mathbf{C}_{\mathrm{left}}$ and $\mathbf{C}_{\mathrm{right}}$ of $\mathbf{C}$ with associated notations $\xrightarrow{\mathrm{l}}$ and $\xrightarrow{\mathrm{r}}$ for morphisms, which are stable under finite cartesian products and such that for all $u : c_l \xrightarrow{\mathrm{l}} c_l'$ and $v : c_r \xrightarrow{\mathrm{r}} c_r'$,

$$v \Rightarrow u \;:\; c_r' \Rightarrow c_l \xrightarrow{\mathrm{l}} c_r \Rightarrow c_l' \qquad \text{and} \qquad u \Rightarrow v \;:\; c_l' \Rightarrow c_r \xrightarrow{\mathrm{r}} c_l \Rightarrow c_r' \;.$$

- for every object c of $\mathbf{C}$, two objects L_c and R_c of $\mathbf{C}$ such that there exists morphisms:

$$\begin{array}{ccc} L_1 \xrightarrow{\mathrm{l}} 1 & L_{c\times c'} \xrightarrow{\mathrm{l}} L_c \times L_{c'} & L_{c\Rightarrow c'} \xrightarrow{\mathrm{l}} R_c \Rightarrow L_{c'} \\ 1 \xrightarrow{\mathrm{r}} R_1 & R_c \times R_{c'} \xrightarrow{\mathrm{r}} R_{c\times c'} & L_c \Rightarrow R_{c'} \xrightarrow{\mathrm{r}} R_{c\Rightarrow c'} \;. \end{array}$$

The squeezing category

If $\mathbf{C}$ comes with a squeezing structure, then we define $\mathbf{Sqz}(\mathbf{C})$ to be the full subcategory of $\mathbf{C}$ whose objects are the c such that there exists morphisms

$$u \quad : \quad L_c \xrightarrow{\mathsf{l}} c \qquad \text{and} \qquad v \quad : \quad c \xrightarrow{\mathsf{r}} R_c \ .$$

Theorem. $\mathbf{Sqz}(\mathbf{C})$ is a sub-CCC of $\mathbf{C}$.

Therefore, for any object c of $\mathbf{Sqz}(\mathbf{C})$ and any type A, there exists maps

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

Types, finite sets and their squeezing structure

We consider the category $\mathrm{P}(\mathbf{Lam} \times \mathbf{FinSet})$, whose objects are triples $(B, Q, \Vdash)$.

We have a functor $F_{(-)} : \mathbf{FinSet} \to \mathbf{Lam}$ defined as

$$F_Q \quad := \quad \mathrm{o}^{|Q|} \Rightarrow \mathrm{o}$$

and we note $\sim_Q$ the graph of the bijection $\Lambda_{\beta\eta}\langle F_Q \rangle \simeq Q$.

For any $(B, Q, \Vdash)$, we define $L_{(B,Q,\Vdash)}$ and $R_{(B,Q,\Vdash)}$ to be $(F_Q, Q, \sim_Q)$. We define left and right morphisms to be pairs (t, Id_Q), which gives a squeezing structure.

By taking $(F_Q, Q, \sim_Q)$ as the interpretation of o, we get for any type A two λ-terms

$$u_A \quad : \quad F_{[\![A]\!]_Q} \longrightarrow A[F_Q] \qquad \text{and} \qquad v_A \quad : \quad A[F_Q] \longrightarrow F_{[\![A]\!]_Q}$$

such that (u_A, Id_Q) and (v_A, Id_Q) are morphisms of logical relations.

A plan of the situation

$$
\begin{tikzcd}
 & \mathbf{Lam} \arrow[ddl, bend right, "(-)[B]"'] \arrow[dd, bend right, "{[\![-]\!]_{(B,Q,\Vdash)}}"'] \arrow[ddr, bend left, "{[\![-]\!]_Q}"] & \\
 & 1 \arrow[u, "\Phi"'] \arrow[d, "{(B,Q,\Vdash)}"] \arrow[ddl, "B"'] \arrow[ddr, "Q"] & \\
 & \mathbf{Sqz}(\mathrm{P}(\mathbf{Lam} \times \mathbf{FinSet})) \arrow[d] & \\
\mathbf{Lam} & \mathrm{P}(\mathbf{Lam} \times \mathbf{FinSet}) \arrow[l] \arrow[r] & \mathbf{FinSet}
\end{tikzcd}
$$

Encoding semantic languages into syntactic ones

Let $F \subseteq [\![A]\!]_Q$ represented by $\chi : [\![A]\!]_Q \to 2 = \{\bot, \top\}$. It induces a morphism

$$(F_\chi, \chi) \quad : \quad (F_{[\![A]\!]_Q}, [\![A]\!]_Q, \sim_{[\![A]\!]_Q}) \longrightarrow (F_2, 2, \sim_2)$$

By precomposing with $(v_A, \mathrm{Id}_{[\![A]\!]_Q})$, we obtain a morphism

$$(\underbrace{F_\chi \circ v_A}_{r}, \chi) \quad : \quad (A[F_Q], [\![A]\!]_Q, \sim_{[\![A]\!]_Q}) \longrightarrow (F_2, 2, \sim_2)$$

For any $t : A$, we then have that $(F_\chi \circ v_A)\ t[F_Q] \ \sim_2 \ \chi([\![t]\!]_Q)$, so

$$(F_\chi \circ v_A)\ t[F_Q] \ =_{\beta\eta} \ \text{true} \quad \Longleftrightarrow \quad [\![t]\!]_Q \ \in \ F \ .$$

Theorem. Any **FinSet**-recognizable language is syntactically regular.

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, with effects.
- 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, with effects.
- 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.