Finitary semantics and regular languages of λ-terms

Vincent Moreau, joint work with Tito Nguyễn

ASV day 2023

December the 6th, 2023

IRIF, Université Paris Cité, Inria Paris

Finite words

Let $\Sigma = \{a, b\}$ be a two-letter alphabet. We write the word $aab \in \Sigma^*$ as

$$a, b \vdash aab \ .$$

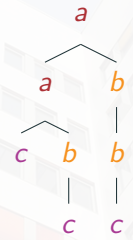
Words can be encodes as finite ranked trees: each letter of Σ has arity 1, and we add a constant of arity 0.

We write the ranked tree associated to aab as

$$a : 1, b : 1, c : 0 \vdash b(a(a(c))) \ .$$

Finite trees

Let Σ be the ranked alphabet $\{a : 2, b : 1, c : 0\}$. Then, the tree



is represented by the judgment

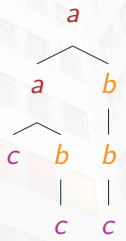
$$a : 2, b : 1, c : 0 \vdash a(a(c, b(c)), b(b(c))) .$$

Adding simple types

The ranked alphabet $\{a : 2, b : 1, c : 0\}$ can be seen as the typed context

$$a : o^2 \Rightarrow o,\ \ b : o \Rightarrow o,\ \ c : o\ .$$

With that in mind, the tree



is represented by the typed judgment

$$a : o^2 \Rightarrow o, b : o \Rightarrow o, c : o \ \vdash\ a(a(c, b(c)),\ b(b(c))) : o\ .$$

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

The λ-abstraction

Consider the typed context

$$\to : \mathfrak{o}^2 \Rightarrow \mathfrak{o},\ \forall : (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o},\ \bot : \mathfrak{o}\ .$$

The $\forall$ takes a function as an argument. They can be built using the λ-abstraction.

The following λ-terms

$$\forall\ (\lambda(\varphi : \mathfrak{o}).\ \varphi \to \varphi)$$
$$\forall\ (\lambda(\varphi : \mathfrak{o}).\ ((\varphi \to \bot) \to \bot) \to \varphi)$$
$$\forall\ (\lambda(\varphi : \mathfrak{o}).\ \forall\ (\lambda(\psi : \mathfrak{o}).\ ((\varphi \to \psi) \to \varphi) \to \varphi)$$

are all of type $\mathfrak{o}$ in the fixed context.

Closing the λ-terms

The λ-abstraction moves variables from the context into the λ-term:

$$a : A, b : B, c : C, d : D \ \vdash \ t : T$$

$$\wr$$

$$a : A, b : B, c : C \ \vdash \ \lambda(d : D).\, t : D \Rightarrow T \ .$$

Repeating this on trees, we obtain closed λ-terms like

$$a : \mathfrak{o}^2 \Rightarrow \mathfrak{o}, b : \mathfrak{o} \Rightarrow \mathfrak{o}, c : \mathfrak{o} \ \vdash \ a(a(c, b(c)),\ b(b(c))) : \mathfrak{o}$$

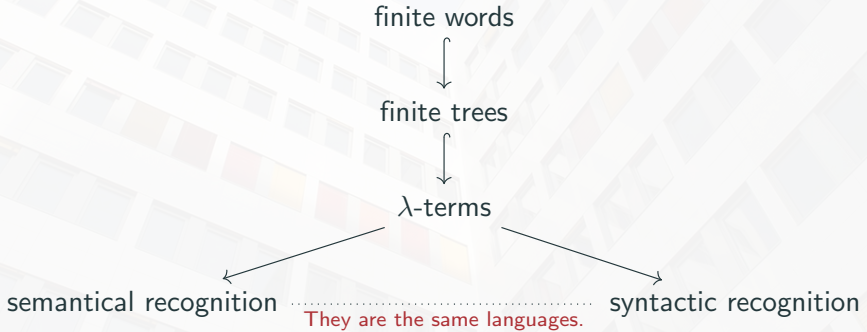
$$\wr\ \wr\ \wr$$

$$\vdash \ \lambda(a : \mathfrak{o}^2 \Rightarrow \mathfrak{o}).\, \lambda(b : \mathfrak{o} \Rightarrow \mathfrak{o}).\, \lambda(c : \mathfrak{o}).\, a(a(c, b(c)),\ b(b(c))) : \mathrm{Church}_{[2,1,0]}$$

with no free variable and which are of type

$$\mathrm{Church}_{[2,1,0]} \quad := \quad (\mathfrak{o}^2 \Rightarrow \mathfrak{o}) \Rightarrow (\mathfrak{o} \Rightarrow \mathfrak{o}) \Rightarrow \mathfrak{o} \Rightarrow \mathfrak{o} \ .$$

Two notions of recognition



Language of λ-terms: the semantic side

If Q is a finite set, then any tree $t : \mathrm{Church}_{[2,1,0]}$ can be interpreted as

$$[\![t]\!]_Q \quad \in \quad (Q^2 \Rightarrow Q) \Rightarrow (Q \Rightarrow Q) \Rightarrow Q \Rightarrow Q \ .$$

For all $\delta_a : Q^2 \to Q$, $\delta_b : Q \to Q$, $\delta_c \in Q$, we have

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

$\rightarrow$ **Interpreting the λ-calculus in finite sets specializes to runs in deterministic finite bottom-up tree automata.**

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 finite set Q and any subset $F \subseteq [\![A]\!]_Q$, we define

$$L_F \quad := \quad \{t : A \mid [\![t]\!]_c \in F\} \ .$$

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

Quantified boolean formulas

We have seen that λ-terms of the type

$$\mathrm{QBF} \quad := \quad \underbrace{\mathbb{o}^2 \Rightarrow \mathbb{o}}_{\to} \Rightarrow \underbrace{(\mathbb{o} \Rightarrow \mathbb{o}) \Rightarrow \mathbb{o}}_{\forall} \Rightarrow \underbrace{\mathbb{o}}_{\bot} \Rightarrow \mathbb{o}$$

By taking $Q = \{0, 1\}$ with the transitions

$$\delta_\to \in Q^2 \Rightarrow Q \qquad \delta_\forall \in (Q \Rightarrow Q) \Rightarrow Q \qquad \delta_\bot \in Q \ ,$$

for any λ-term $t : \mathrm{QBF}$, we have

$$t \text{ represents a true formula} \quad \iff \quad [\![t]\!]_Q(\delta_\to, \delta_\forall, \delta_\bot) = 1 \ .$$

which shows that true formula form a semantically recognizable language.

Language of λ-terms: the syntactic side

We consider the type

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

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

$$\mathrm{true} \;:=\; \lambda(a : o).\lambda(b : o).\, a \quad \text{and} \quad \mathrm{false} \;:=\; \lambda(a : o).\lambda(b : o).\, b\ .$$

An automaton can be encoded as a closed λ-term of type

$$r \quad : \quad ((B^2 \Rightarrow B) \Rightarrow (B \Rightarrow B) \Rightarrow B \Rightarrow B) \Rightarrow \mathrm{Bool}$$

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

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

Syntactic languages of λ-terms

If A is a simple type, then $A[B]$ is the substitution of B for $\mathfrak{o}$ in A.

If t is λ-term of type A, then $t[B]$ is a λ-term of type $t[A]$, defined inductively as

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

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

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

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

Our theorem

It was known that, at type Church_Σ and L a language of λ-terms of type Church_Σ,

$$L \text{ is semantically recognizable} \iff L \text{ is syntactically regular}$$

which is also equivalent to being regular in the usual sense.

Theorem (M., Nguyen). For every type A and L a language of λ-terms of type A,

$$L \text{ is semantically recognizable} \iff L \text{ is syntactically regular.}$$

Actually stronger: still hold when replacing finite sets by other sufficiently well-behaved finitary model of the λ-calculus! For example:

- deterministic models (finite sets and functions),
- nondeterministic models (finite Scott domains).

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 which languages of formulas are regular languages of λ-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 which languages of formulas are regular languages of λ-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.