

A quick introduction to higher-order automata

Vincent Moreau

ASV day 2022

October 21, 2022

IRIF, Université Paris Cité, Inria Paris

Who am I?

PhD student since September 2021, in PPS (algebra), ASV (automata) and Picube.



Paul-André Melliès

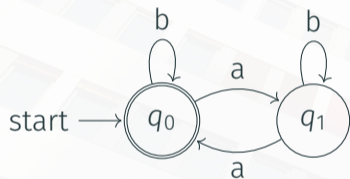


Sam van Gool

Word automata

A run of the word *aba* in this automaton

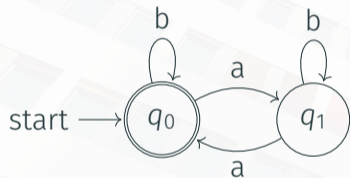
q_0



Word automata

A run of the word *aba* in this automaton

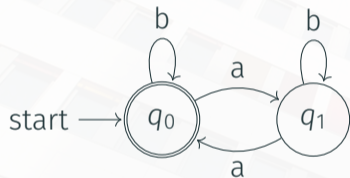
$$q_0 \xrightarrow{a} q_1$$



Word automata

A run of the word *aba* in this automaton

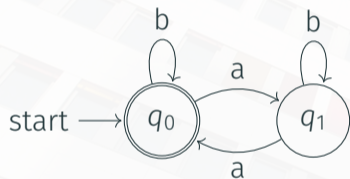
$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1$$



Word automata

A run of the word *aba* in this automaton

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0$$

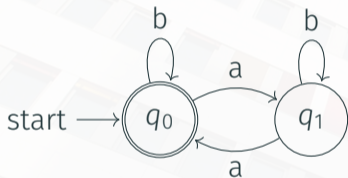


Word automata

A run of the word *aba* in this automaton

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0$$

...can be rewritten as a derivation



$$\frac{q_1 \xrightarrow{a} q_0}{\vdash aba \mid q_0} \quad \frac{q_1 \xrightarrow{b} q_1 \quad \frac{q_0 \xrightarrow{a} q_1 \quad \frac{q_0 \text{ is initial}}{\vdash \epsilon \mid q_0}}{\vdash a \mid q_1}}{\vdash ba \mid q_1}$$

Runs of words through the lens of types

If w is a word and q is a state of an automaton, then the statement

$$\vdash w \mid q$$

means that, when running w from the initial state, we arrive at the state q .

This statement is inductively defined with the two following rules:

$$\frac{q \xrightarrow{a} q' \quad \vdash w \mid q}{\vdash aw \mid q'} \qquad \frac{q \text{ is initial}}{\vdash \epsilon \mid q}$$

Proposition. A run of a word w in an automaton is the same thing as a derivation of the judgment $\vdash w \mid q_f$, with q_f a final state.

Idea: an automaton is a machine that **tries to type its input**.

Words are trees

If we have a word

$$w = a_1 \dots a_n$$

over the alphabet Σ , it can be seen as a tree



over the ranked alphabet $\{a : 1, a \in \Sigma\} \cup \{z : 0\}$.

Tree automata

Ranked alphabet $\{\vee : 2, \wedge : 2, \neg : 1, \top : 0, \perp : 0\}$
with the automaton made of

- $Q = \{q_{\top}, q_{\perp}\}$
- $Q_f = \{q_{\top}\}$
- the transitions in the set

$$\begin{aligned} & \{b \rightarrow q_b : b \in Q\} \\ & \cup \{\neg q_b \rightarrow q_{\neg b} : b \in Q\} \\ & \cup \{f q_a q_b \rightarrow q_{f a b} : a, b \in Q, f \in \{\vee, \wedge\}\} \end{aligned}$$



Tree automata

Ranked alphabet $\{\vee : 2, \wedge : 2, \neg : 1, \top : 0, \perp : 0\}$
with the automaton made of

- $Q = \{q_{\top}, q_{\perp}\}$
- $Q_f = \{q_{\top}\}$
- the transitions in the set

$$\begin{aligned} & \{b \rightarrow q_b : b \in Q\} \\ & \cup \{\neg q_b \rightarrow q_{\neg b} : b \in Q\} \\ & \cup \{f q_a q_b \rightarrow q_{f a b} : a, b \in Q, f \in \{\vee, \wedge\}\} \end{aligned}$$

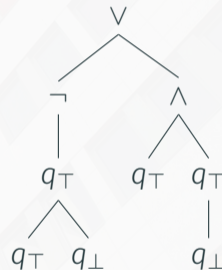


Tree automata

Ranked alphabet $\{\vee : 2, \wedge : 2, \neg : 1, \top : 0, \perp : 0\}$
with the automaton made of

- $Q = \{q_{\top}, q_{\perp}\}$
- $Q_f = \{q_{\top}\}$
- the transitions in the set

$$\begin{aligned} & \{b \rightarrow q_b : b \in Q\} \\ & \cup \{\neg q_b \rightarrow q_{\neg b} : b \in Q\} \\ & \cup \{f q_a q_b \rightarrow q_{f a b} : a, b \in Q, f \in \{\vee, \wedge\}\} \end{aligned}$$

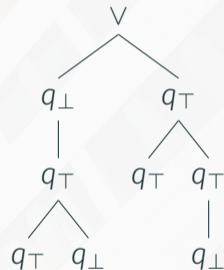


Tree automata

Ranked alphabet $\{\vee : 2, \wedge : 2, \neg : 1, \top : 0, \perp : 0\}$
with the automaton made of

- $Q = \{q_{\top}, q_{\perp}\}$
- $Q_f = \{q_{\top}\}$
- the transitions in the set

$$\begin{aligned} & \{b \rightarrow q_b : b \in Q\} \\ & \cup \{\neg q_b \rightarrow q_{\neg b} : b \in Q\} \\ & \cup \{f q_a q_b \rightarrow q_{f a b} : a, b \in Q, f \in \{\vee, \wedge\}\} \end{aligned}$$

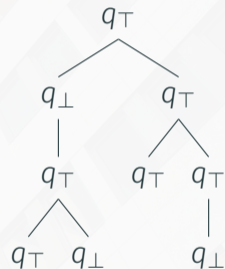


Tree automata

Ranked alphabet $\{\vee : 2, \wedge : 2, \neg : 1, \top : 0, \perp : 0\}$
with the automaton made of

- $Q = \{q_{\top}, q_{\perp}\}$
- $Q_f = \{q_{\top}\}$
- the transitions in the set

$$\begin{aligned} & \{b \rightarrow q_b : b \in Q\} \\ & \cup \{\neg q_b \rightarrow q_{\neg b} : b \in Q\} \\ & \cup \{f q_a q_b \rightarrow q_{f a b} : a, b \in Q, f \in \{\vee, \wedge\}\} \end{aligned}$$



Runs of trees through the lens of types

We consider the same tree now written, with prefix notation:

$$\vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp))$$

and show that its run in the automaton can be rephrased as the derivation

$$\frac{\dots \quad \dots}{\vdash \vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp)) \mid q_{\top}}$$

Runs of trees through the lens of types

We consider the same tree now written, with prefix notation:

$$\vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp))$$

and show that its run in the automaton can be rephrased as the derivation

$$\frac{\frac{\dots \quad \dots}{\vdash \vee (\neg (\vee \top \perp)) \mid q_{\top} \multimap q_{\top}} \quad \frac{\dots \quad \dots}{\vdash \wedge \top (\neg \perp) \mid q_{\top}}}{\vdash \vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp)) \mid q_{\top}}$$

Runs of trees through the lens of types

We consider the same tree now written, with prefix notation:

$$\vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp))$$

and show that its run in the automaton can be rephrased as the derivation

$$\begin{array}{c}
 \frac{q_{\perp} \multimap q_{\top} \multimap q_{\top} \in \delta(\vee)}{\vdash \vee \mid q_{\perp} \multimap q_{\top} \multimap q_{\top}} \quad \frac{\dots \quad \dots}{\vdash \neg (\vee \top \perp) \mid q_{\perp}} \quad \frac{\dots \quad \dots}{\vdash \wedge \top \mid q_{\top} \multimap q_{\top}} \quad \frac{\dots \quad \dots}{\vdash \neg \perp \mid q_{\top}} \\
 \hline
 \frac{\vdash \vee (\neg (\vee \top \perp)) \mid q_{\top} \multimap q_{\top} \quad \vdash \wedge \top (\neg \perp) \mid q_{\top}}{\vdash \vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp)) \mid q_{\top}}
 \end{array}$$

Runs of trees through the lens of types

We consider the same tree now written, with prefix notation:

$$\vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp))$$

and show that its run in the automaton can be rephrased as the derivation

$$\begin{array}{c}
 \frac{q_{\perp} \multimap q_{\top} \multimap q_{\top} \in \delta(\vee)}{\vdash \vee \mid q_{\perp} \multimap q_{\top} \multimap q_{\top}} \quad \frac{\frac{q_{\top} \multimap q_{\perp} \in \delta(\neg)}{\vdash \neg \mid q_{\top} \multimap q_{\perp}} \quad \frac{\dots \quad \dots}{\vdash \vee \top \perp \mid q_{\top}}}{\vdash \neg (\vee \top \perp) \mid q_{\perp}} \quad \frac{\frac{q_{\top} \multimap q_{\top} \multimap q_{\top} \in \delta(\wedge)}{\vdash \wedge \mid q_{\top} \multimap q_{\top} \multimap q_{\top}} \quad \frac{q_{\top} \in \delta(\top)}{\vdash \top \mid q_{\top}}}{\vdash \wedge \top \mid q_{\top} \multimap q_{\top}} \quad \frac{\frac{q_{\perp} \multimap q_{\top} \in \delta(\neg)}{\vdash \neg \mid q_{\perp} \multimap q_{\top}} \quad \frac{q_{\perp} \in \delta(\perp)}{\vdash \perp \mid q_{\perp}}}{\vdash \neg \perp \mid q_{\top}} \\
 \hline
 \frac{\vdash \vee (\neg (\vee \top \perp)) \mid q_{\top} \multimap q_{\top} \quad \vdash \wedge \top (\neg \perp) \mid q_{\top}}{\vdash \vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp)) \mid q_{\top}}
 \end{array}$$

Runs of trees through the lens of types

We consider the same tree now written, with prefix notation:

$$\vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp))$$

and show that its run in the automaton can be rephrased as the derivation

$$\begin{array}{c}
 \frac{q_{\top} \multimap q_{\perp} \multimap q_{\top} \in \delta(\vee) \quad q_{\top} \in \delta(\top)}{\vdash \vee \mid q_{\top} \multimap q_{\perp} \multimap q_{\top}} \quad \frac{q_{\perp} \in \delta(\perp)}{\vdash \perp \mid q_{\perp}} \\
 \frac{q_{\perp} \multimap q_{\top} \multimap q_{\top} \in \delta(\neg) \quad \frac{\vdash \neg \mid q_{\top} \multimap q_{\perp}}{\vdash \neg (\vee \top \perp) \mid q_{\perp}}}{\vdash \vee (\neg (\vee \top \perp)) \mid q_{\top} \multimap q_{\top}} \quad \frac{\frac{q_{\top} \multimap q_{\top} \multimap q_{\top} \in \delta(\wedge) \quad q_{\top} \in \delta(\top)}{\vdash \wedge \mid q_{\top} \multimap q_{\top} \multimap q_{\top}} \quad \frac{q_{\perp} \multimap q_{\top} \in \delta(\neg) \quad q_{\perp} \in \delta(\perp)}{\vdash \neg \perp \mid q_{\top}}}{\vdash \wedge \top (\neg \perp) \mid q_{\top}} \\
 \hline
 \vdash \vee (\neg (\vee \top \perp)) (\wedge \top (\neg \perp)) \mid q_{\top}
 \end{array}$$

An automaton is a machine that tries to type its input.

Trees are λ -terms

If we have a ranked tree, for instance



over the ranked alphabet $\{a : 2, b : 1, c : 0\}$, then it can be seen as a λ -term

$$a : \circ \Rightarrow \circ \Rightarrow \circ, b : \circ \Rightarrow \circ, c : \circ \vdash a(a c c)(b c) : \circ$$

which induces a closed term by binding all the free variables:

$$\lambda a. \lambda b. \lambda c. a(a c c)(b c) : (\circ \Rightarrow \circ \Rightarrow \circ) \Rightarrow (\circ \Rightarrow \circ) \Rightarrow \circ \Rightarrow \circ$$

In particular, this gives an encoding of words as λ -terms of the type

$$\text{Church}_\Sigma \quad := \quad (\circ \Rightarrow \circ) \Rightarrow \dots \Rightarrow (\circ \Rightarrow \circ) \Rightarrow (\circ \Rightarrow \circ) .$$

About recognizability in the λ -calculus

Recognizability of λ -terms is a generalization of recognizability for words and trees. Runs, defined as derivations, are of the form

$$\Sigma \vdash M : A \mid \delta, q .$$

Theorem. For any λ -term M of type A , the singleton language

$$\{N : A \mid M =_{\beta\eta} N\}$$

is a regular language of λ -terms ([Sal10]).

Question:

Can we lift other tools and results to the more general case of λ -terms?

We give the example of **profinite methods**.

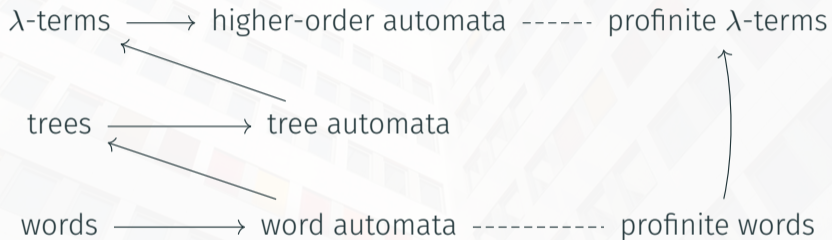
λ -terms \longrightarrow higher-order automata



trees \longrightarrow tree automata



words \longrightarrow word automata



Profinite words

A profinite word u can be described as a family of maps

$$u_M : [\Sigma, M] \longrightarrow M \quad \text{where } M \text{ ranges over all finite monoids}$$

such that the following condition is verified: for any finite monoids M and N ,

$$\forall p \in [\Sigma, M], \forall \varphi \in \mathbf{Hom}(M, N), \quad u_N(\varphi \circ p) = \varphi(u_M(p)) .$$

Any finite word $a_1 \dots a_n$ is a profinite word with components

$$u_M : p \mapsto p(a_1) \dots p(a_n) \quad \text{where } M \text{ ranges over all finite monoids}$$

but there are a lot of non-finite ones.

$$\text{regular languages} \overset{\text{Stone duality}}{\longleftrightarrow} \text{profinite words}$$

Profinite λ -terms

We have defined a notion of profinite λ -term of any type A . In the case of

$$\mathbf{Church}_\Sigma \quad := \quad \underbrace{(\mathbb{O} \Rightarrow \mathbb{O}) \Rightarrow \dots \Rightarrow (\mathbb{O} \Rightarrow \mathbb{O})}_{|\Sigma| \text{ times}} \Rightarrow (\mathbb{O} \Rightarrow \mathbb{O}) ,$$

a profinite λ -term amounts to a family of maps

$$\theta_Q : [\Sigma, [Q, Q]] \longrightarrow [Q, Q] \quad \text{where } Q \text{ ranges over all finite sets,}$$

which verifies a condition of parametricity.

Theorem. There is a bijection between the profinite λ -terms of type \mathbf{Church}_Σ and the profinite words on Σ .

Conclusion

Current & future work:

- find a syntax for parametric λ -terms of any type in the deterministic model;
- determine the parametric λ -terms of type \mathbf{Church}_Σ in the model associated to nondeterministic automata;
- investigate a generalization of logic on words with **MSO** to a logic on λ -terms.

Conclusion

Current & future work:

- find a syntax for parametric λ -terms of any type in the deterministic model;
- determine the parametric λ -terms of type \mathbf{Church}_Σ in the model associated to nondeterministic automata;
- investigate a generalization of logic on words with **MSO** to a logic on λ -terms.

Thank you for your attention!

Any questions?

Bibliography

- [Geh13] Mai Gehrke. *Stone duality, topological algebra, and recognition*. 2013. DOI: [10.48550/ARXIV.1309.2422](https://doi.org/10.48550/ARXIV.1309.2422).
- [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.
- [Pin] Jean-Eric Pin. “Profinite Methods in Automata Theory”. In: *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*. IBFI Schloss Dagstuhl.
- [Sal09] Sylvain Salvati. “Recognizability in the Simply Typed Lambda-Calculus”. In: Springer Berlin Heidelberg, 2009.
- [Sal10] Sylvain Salvati. “On the Membership Problem for Non-Linear Abstract Categorical Grammars”. In: *J. Log. Lang. Inf.* 19.2 (2010), pp. 163–183. DOI: [10.1007/s10849-009-9110-0](https://doi.org/10.1007/s10849-009-9110-0).

Rules of higher-order automata

$$\frac{q \leq q' \quad q' \in \delta(a)}{\langle \Sigma, a : A \vdash a : A \mid \delta, q \rangle} \text{Var}$$

$$\frac{\langle \Sigma, a : A \vdash M : B \mid \delta[a \mapsto u], q \rangle}{\langle \Sigma \vdash \lambda(a : A).M : A \Rightarrow B \mid \delta, u \multimap q \rangle} \text{Abs}$$

$$\frac{\langle \Sigma \vdash M : B \Rightarrow A \mid \delta, u \multimap q \rangle \quad \langle \langle \Sigma \vdash N : B \mid \delta, u \rangle \rangle}{\langle \Sigma \vdash MN : A \mid \delta, q \rangle} \text{App}$$

$$\frac{\langle \Sigma \vdash M : A \mid \delta, q_1 \rangle \quad \dots \quad \langle \Sigma \vdash M : A \mid \delta, q_n \rangle}{\langle \langle \Sigma \vdash M : A \mid \delta, \{q_1, \dots, q_n\} \rangle \rangle} \text{Bag}$$

The inverse bijections T and W

Pro \rightarrow **Para**. Every profinite word u induces a parafinite term with components

$$T(u)_Q : \begin{array}{ccc} \Sigma \Rightarrow (Q \Rightarrow Q) & \longrightarrow & Q \Rightarrow Q \\ p & \longmapsto & u_{Q \Rightarrow Q}(p) \end{array}$$

given the fact that $Q \Rightarrow Q$ is a monoid for the function composition.

Para \rightarrow **Pro**. Every parametric term θ induces a profinite word with components

$$W(\theta)_M : \begin{array}{ccc} \Sigma \Rightarrow M & \longrightarrow & M \\ p & \longmapsto & \theta_M(i_M \circ p)(e_M) \end{array} \quad \begin{array}{ccc} \Sigma \Rightarrow (M \Rightarrow M) & \xrightarrow{\theta_M} & M \Rightarrow M \\ i_M \circ - \uparrow & & \downarrow -(e_M) \\ \Sigma \Rightarrow M & \xrightarrow{W(\theta)_M} & M \end{array}$$

where $i_M : M \rightarrow (M \Rightarrow M)$ is the Cayley embedding.

These are bijections between profinite words and parametric λ -terms.