Tree automata and two-way automata as functors

Minimisation and unidirectionalisation

Victor Iwaniack

I2M, Université Aix-Marseille

Séminaire LDP, January 8th, 2026

Deterministic automata

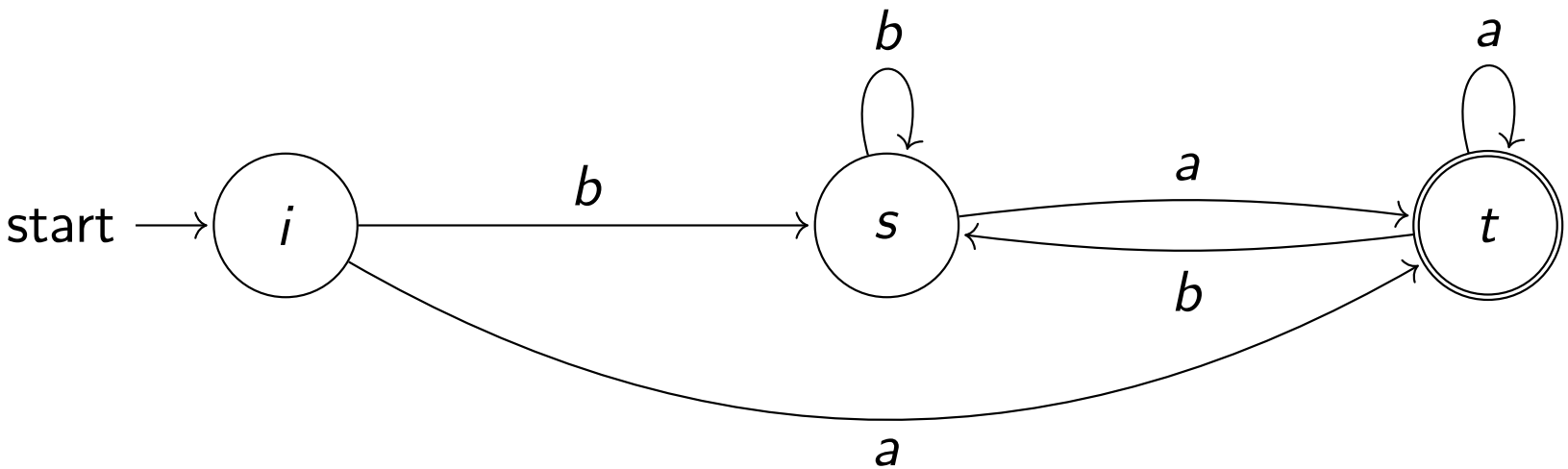
A *word* over the *alphabet* Σ is an element of the free monoid Σ^*.
A *(one-way, complete,) deterministic automaton* (Q, i, F, δ) is given by a set of *states* Q, an *initial state* $i \in Q$, a subset of *accepting states* $F \subset Q$ and a *transition function* $Q \times \Sigma \to Q$.

Deterministic automata

A *word* over the *alphabet* Σ is an element of the free monoid Σ^*.
A *(one-way, complete,) deterministic automaton* (Q, i, F, δ) is given by a set of *states* Q, an *initial state* $i \in Q$, a subset of *accepting states* $F \subset Q$ and a *transition function* $Q \times \Sigma \to Q$.

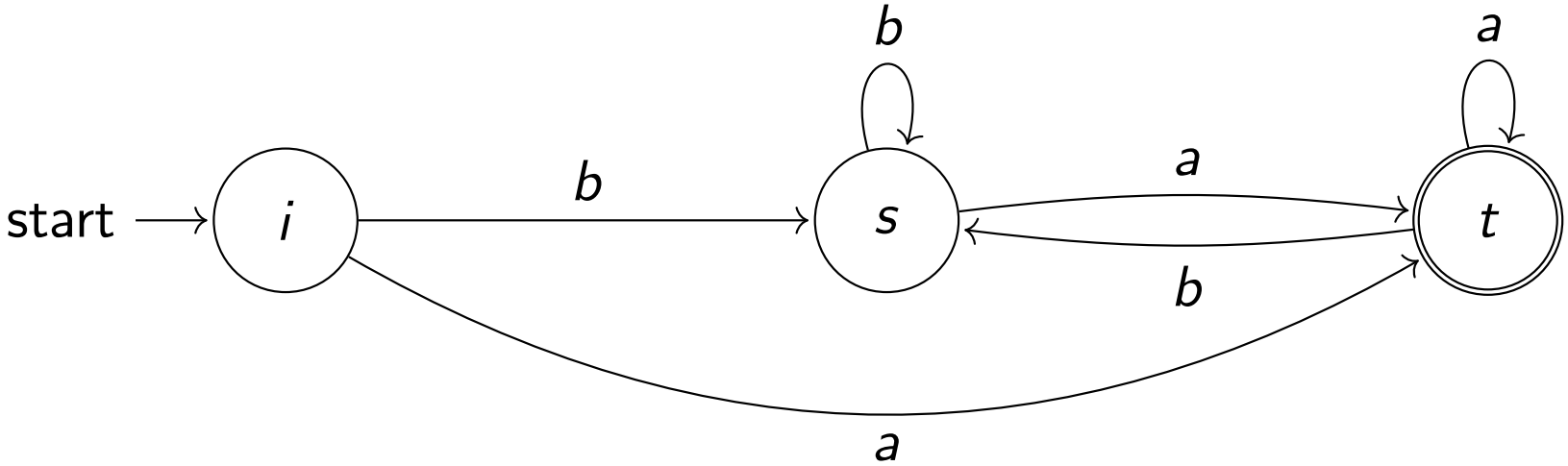
Example

For $\Sigma = \{a, b\}$



Example

The automaton

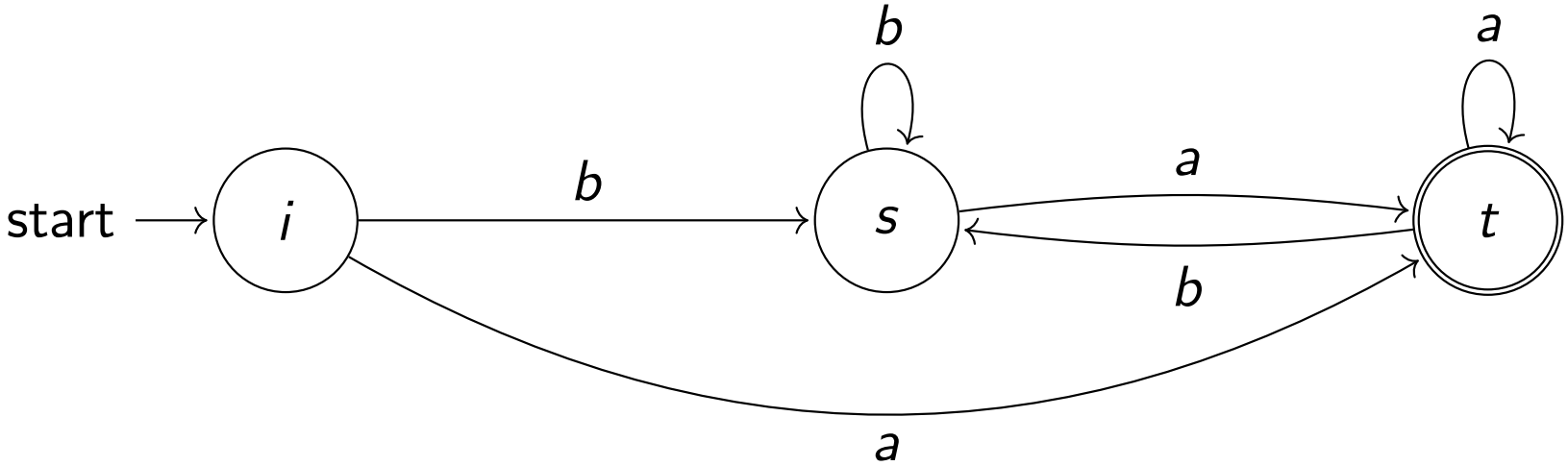


recognises the word *baba* because

$$i \xrightarrow{b} s \xrightarrow{a} t \xrightarrow{b} s \xrightarrow{a} t$$

Example

The automaton



recognises the word *baba* because

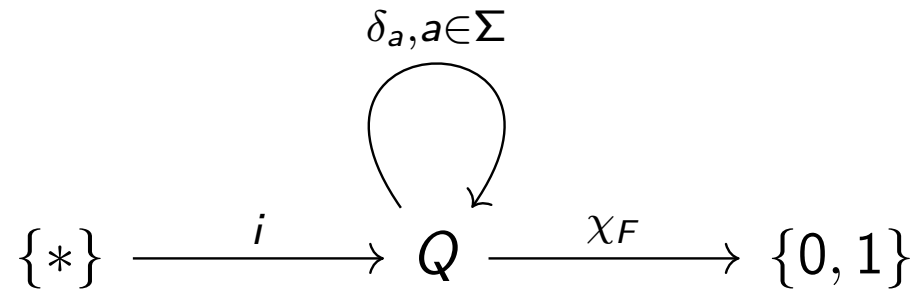
$$i \xrightarrow{b} s \xrightarrow{a} t \xrightarrow{b} s \xrightarrow{a} t$$

but not the word *bab* because

$$i \xrightarrow{b} t \xrightarrow{a} f \xrightarrow{b} s$$

The Colcombet and Petrișan [CP20] point of view

An automaton is therefore an action of the monoid Σ^* on Q. The functorial point of view of Colcombet and Petrișan [CP20] extends the monoid-action-as-a-functor to encompass $i \in Q$ and $F \subset Q$:



(where $\delta_a = \delta(-, a)$).

The Colcombet and Petrișan [CP20] point of view

An automaton is therefore an action of the monoid Σ^* on Q. The functorial point of view of Colcombet and Petrișan [CP20] extends the monoid-action-as-a-functor to encompass $i \in Q$ and $F \subset Q$:

$$\mathbb{1} = \{*\} \xrightarrow{i} Q \xrightarrow{\chi_F} \{0,1\} = \Omega \qquad (\delta_a, a \in \Sigma)$$

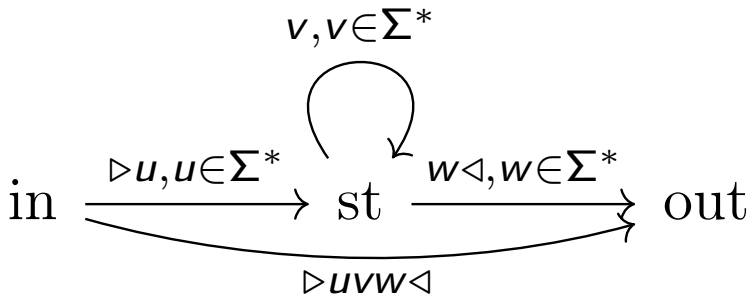
(where $\delta_a = \delta(-, a)$).
The data of an automaton is therefore equivalent to the data of a *bipointed* functor

$$\left(\mathcal{F} \left\{ \mathrm{in} \xrightarrow{\triangleright} \mathrm{st} \xrightarrow{\triangleleft} \mathrm{out} \right\}, \mathrm{in}, \mathrm{out} \right) \longrightarrow (\mathrm{Set}, \mathbb{1}, \Omega) \qquad (a \in \Sigma)$$

Denote by $\mathcal{I}_{\Sigma^*}$ the (free) source category.

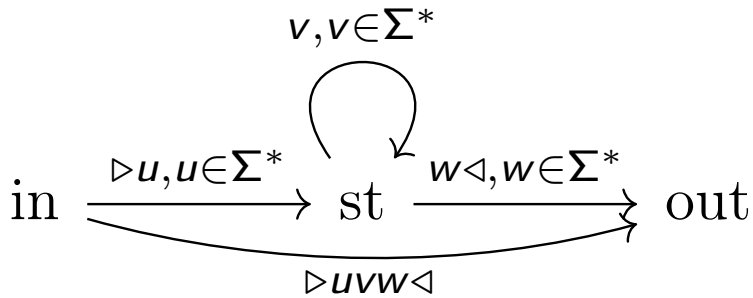
Languages as functors

The full subcategory $\mathcal{O}_{\Sigma^*}$ of $\mathcal{I}_{\Sigma^*}$ generated by in and out has only arrows $\triangleright w \triangleleft \in \mathcal{O}_{\Sigma^*}(\mathrm{in}, \mathrm{out}) \cong \Sigma^*$:



Languages as functors

The full subcategory $\mathcal{O}_{\Sigma^*}$ of $\mathcal{I}_{\Sigma^*}$ generated by in and out has only arrows $\triangleright w \triangleleft \in \mathcal{O}_{\Sigma^*}(\mathrm{in}, \mathrm{out}) \cong \Sigma^*$:



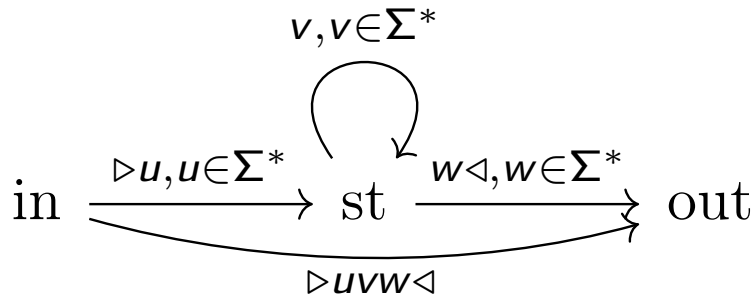
Therefore, the restriction of an automaton

$$\mathcal{A} : (\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

to $\mathcal{O}_{\Sigma^*}$ sends $\triangleright w \triangleleft : \mathrm{in} \to \mathrm{out}$ to $\mathbb{1} \to \Omega$ which is the truth value of whether w is recognised or not.

Languages as functors

The full subcategory $\mathcal{O}_{\Sigma^*}$ of $\mathcal{I}_{\Sigma^*}$ generated by in and out has only arrows $\triangleright w \triangleleft \in \mathcal{O}_{\Sigma^*}(\mathrm{in}, \mathrm{out}) \cong \Sigma^*$:



Therefore, the restriction of an automaton

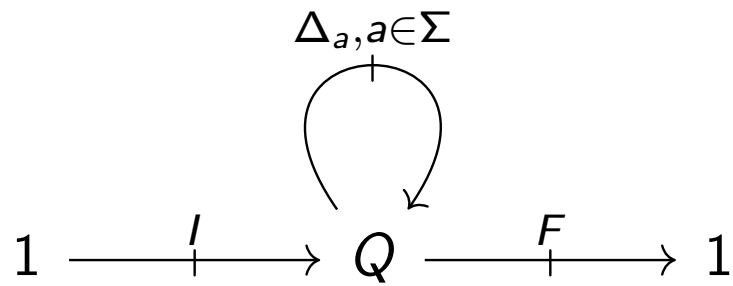
$$\mathcal{A} : (\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

to $\mathcal{O}_{\Sigma^*}$ sends $\triangleright w \triangleleft : \mathrm{in} \to \mathrm{out}$ to $\mathbb{1} \to \Omega$ which is the truth value of whether w is recognised or not.
Languages thus correspond to bipointed functors

$$L : (\mathcal{O}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

Before two-way: non-deterministic automata

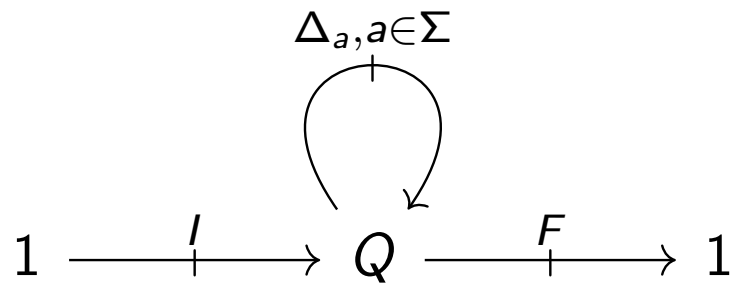
A *non-deterministic (one-way) automaton* is a tuple (Q, I, F, Δ) i.e. $i \in Q$ is replaced by $I \subset Q$ and $\delta : Q \times \Sigma \to Q$ by a relation $\Delta : Q \times \Sigma \nrightarrow Q$:



(where $\Delta_a = \{(q, r) \in Q^2 | (q, a, r) \in \Delta\}$).

Before two-way: non-deterministic automata

A *non-deterministic (one-way) automaton* is a tuple (Q, I, F, Δ)
i.e. $i \in Q$ is replaced by $I \subset Q$ and $\delta : Q \times \Sigma \to Q$ by a relation
$\Delta : Q \times \Sigma \nrightarrow Q$:

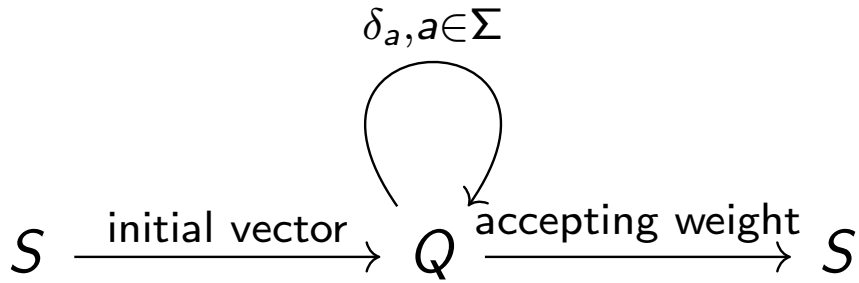


(where $\Delta_a = \{(q, r) \in Q^2 | (q, a, r) \in \Delta\}$).
Thus, non-deterministic automata are exactly bipointed functors

$$(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Rel}, 1, 1)$$

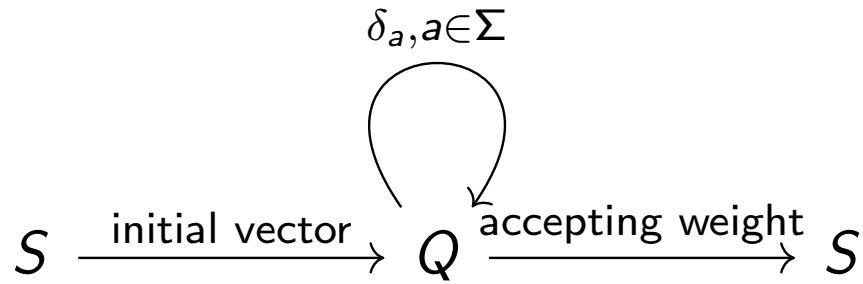
Some other types of automata

- For any semiring S, S-weighted automata are $(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (S\mathrm{Mod}, S, S)$ according to Colcombet and Petrișan [CP20].

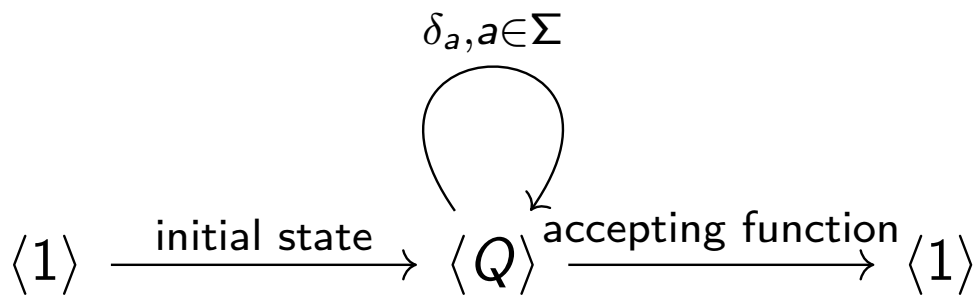


Some other types of automata

- For any semiring S, S-weighted automata are $(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (S\mathrm{Mod}, S, S)$ according to Colcombet and Petrişan [CP20].



- Transducers are $(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Kl}(M_T), \langle 1 \rangle, \langle 1 \rangle)$ where M_T is a monad depending on an *output alphabet* T (also [CP20]).



Some more abstract automata

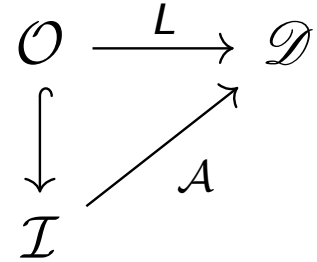
- Everything can be enriched (cf. [Iwa25]), in particular, in the category Nom of *nominal sets* (cf. [Iwa24]) so that nominal automata (generalised finite memory automata) correspond to *enriched* bipointed functors $(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Nom}, \mathbb{1}, \Omega)$.

Some more abstract automata

- Everything can be enriched (cf. [Iwa25]), in particular, in the category Nom of *nominal sets* (cf. [Iwa24]) so that nominal automata (generalised finite memory automata) correspond to *enriched* bipointed functors $(\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathrm{Nom}, \mathbb{1}, \Omega)$.
- Learning automata : classical automata and transducers by Colcombet, Petrișan, and Stabile [CPS21], generalised transducers by Aristote [Ari23], and automata in toposes by Barbé [Bar24]. Requires a new shape $\mathcal{I}_{Q,T}$ instead of $\mathcal{I}_{\Sigma^*}$.

The general functorial framework

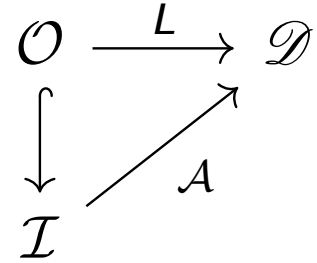
A *language* is an enriched functor $L : \mathcal{O} \to \mathscr{D}$, an L-automaton is a extension of L along $\mathcal{O} \hookrightarrow \mathcal{I}$ (the *behavioural context*).



An *automaton morphism* is an enriched natural transformation that is the identity of L when restricted to $\mathcal{O}$.

The general functorial framework

A *language* is an enriched functor $L : \mathcal{O} \to \mathscr{D}$, an L-automaton is a extension of L along $\mathcal{O} \hookrightarrow \mathcal{I}$ (the *behavioural context*).



An *automaton morphism* is an enriched natural transformation that is the identity of L when restricted to $\mathcal{O}$.
L-automata therefore form a non-full subcategory

$$\mathrm{Auto}(L) \leq [\mathcal{I}, \mathscr{D}]$$

of the category of enriched functors.

Table of Contents

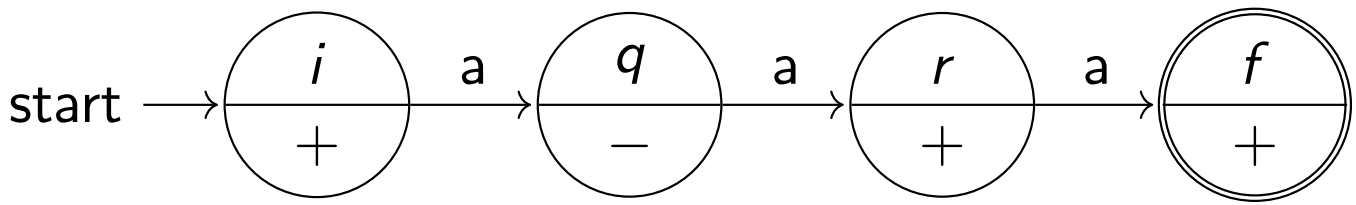
Two-way (non-deterministic) automata

A *two-way non-deterministic automaton* is a tuple (Q^+, Q^-, I, F, Δ) where $(Q^+ \oplus Q^-, I, F, \Delta)$ is a non-deterministic automaton with $I, F \subset Q^+$.

Two-way (non-deterministic) automata

A *two-way non-deterministic automaton* is a tuple (Q^+, Q^-, I, F, Δ) where $(Q^+ \oplus Q^-, I, F, \Delta)$ is a non-deterministic automaton with $I, F \subset Q^+$.

Example



as a mere non-deterministic automaton, rejects the word a, but accepts it as two-way:

$$[i]a \to a[q] \to [r]a \to a[f]$$

The category $\mathbb{I}\mathrm{ntRel}$

Let $w = a_1 a_2 \cdots a_n$ be a word.
For a one-way automaton, the composition of relations

$$\Delta_w := \Delta_{a_1} \Delta_{a_2} \cdots \Delta_{a_n}$$

gives the behaviour of the automaton while reading w i.e.

$q \Delta_w r$ iff there is a computation starting in state q and ending in state r while reading w.

The category $\mathbb{I}\mathrm{nt}\mathrm{Rel}$

Let $w = a_1 a_2 \cdots a_n$ be a word.
For a one-way automaton, the composition of relations

$$\Delta_w := \Delta_{a_1} \Delta_{a_2} \cdots \Delta_{a_n}$$

gives the behaviour of the automaton while reading w i.e.

$q \Delta_w r$ iff there is a computation starting in state q and ending in state r while reading w.

For two-way automata, this has to be done in the category $\mathbb{I}\mathrm{nt}\mathrm{Rel}$ as observed by Hines [Hin03].

The category $\mathbb{I}\mathrm{nt}\mathrm{Rel}$

Definition

The category $\mathbb{I}\mathrm{nt}\mathrm{Rel}$ has objects tuples (A^+, A^-), (B^+, B^-), etc. of sets, and morphisms from (A^+, A^-) to (B^+, B^-) are relations

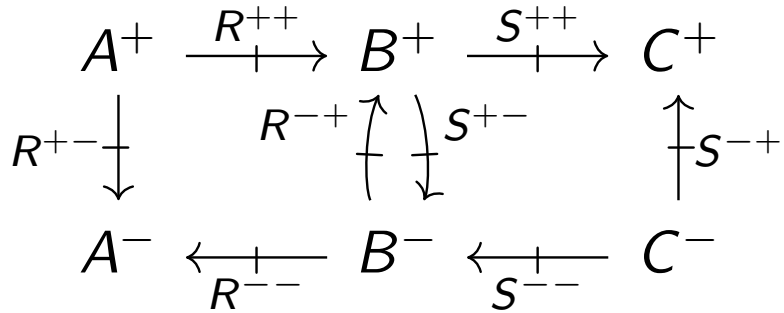
$$R : A^+ \oplus B^- \nrightarrow A^- \oplus B^+$$

equivalent to squares of relations

$$\begin{array}{ccc} A^+ & \xrightarrow{R^{++}} & B^+ \\ {\scriptstyle R^{+-}}\downarrow & & \uparrow{\scriptstyle R^{-+}} \\ A^- & \xleftarrow[R^{--}]{} & B^- \end{array}$$

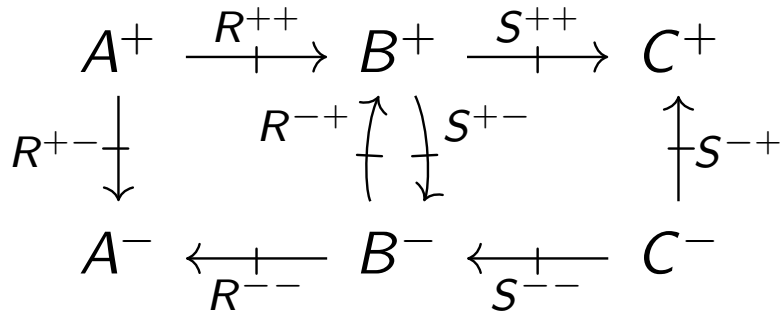
Composition in $\mathbb{I}\mathrm{nt}\mathrm{Rel}$

Take composable arrows $R : (A^+, A^-) \to (B^+, B^-)$ and $S : (B^+, B^-) \to (C^+, C^-)$

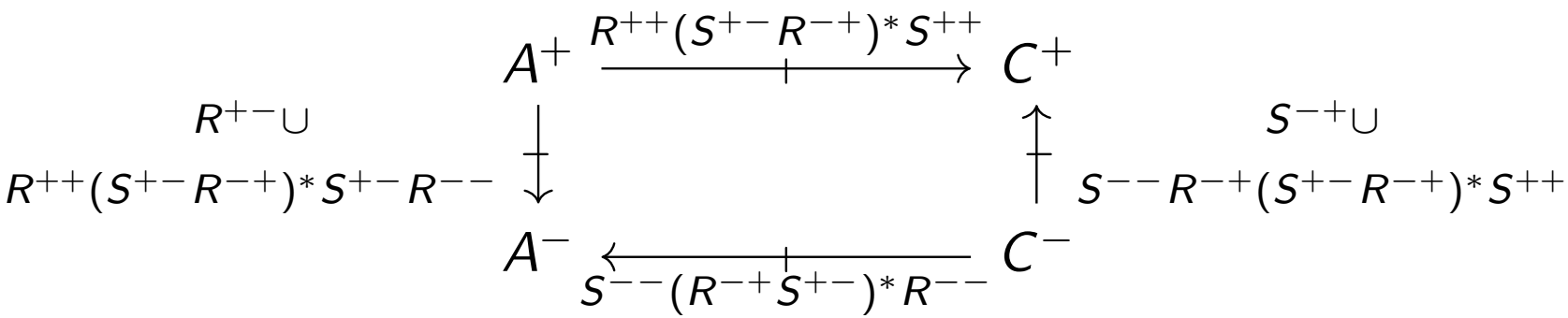


Composition in $\mathbb{I}\mathrm{nt}\mathrm{Rel}$

Take composable arrows $R : (A^+, A^-) \to (B^+, B^-)$ and $S : (B^+, B^-) \to (C^+, C^-)$



Their composite $RS : (A^+, A^-) \to (C^+, C^-)$ is defined using all the possible paths in the preceding diagram:



Birget-Hines' theorem

Let $\Delta_a^* : Q^+ \oplus Q^- \nrightarrow Q^+ \oplus Q^-$ denote the relation such that

$q \Delta_a^* r$ iff there is a computation from q to r while reading the single-letter word a.

Δ_a^* is a morphism $(Q^+, Q^-) \to (Q^+, Q^-)$ of $\mathbb{I}\mathrm{ntRel}$, and we let $\Delta_{a_1 a_2 \cdots a_n}^* := \Delta_{a_1}^* \Delta_{a_2}^* \cdots \Delta_{a_n}^*$ with composition in $\mathbb{I}\mathrm{ntRel}$.

Birget-Hines' theorem

Let $\Delta_a^* : Q^+ \oplus Q^- \nrightarrow Q^+ \oplus Q^-$ denote the relation such that

$q\Delta_a^* r$ iff there is a computation from q to r while reading the single-letter word a.

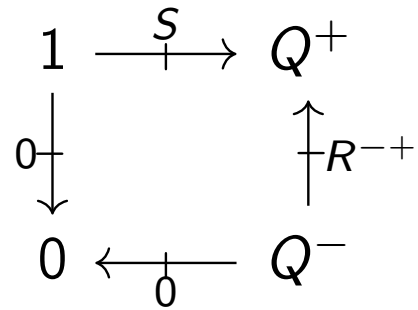
Δ_a^* is a morphism $(Q^+, Q^-) \to (Q^+, Q^-)$ of $\mathbb{I}\mathrm{ntRel}$, and we let $\Delta_{a_1 a_2 \cdots a_n}^* := \Delta_{a_1}^* \Delta_{a_2}^* \cdots \Delta_{a_n}^*$ with composition in $\mathbb{I}\mathrm{ntRel}$.

Theorem (Hines [Hin03] based on Birget [Bir89])

$q\Delta_w^* r$ iff there is a computation in the two-way automaton (Q^+, Q^-, I, F, Δ) while reading the word w, starting in state q and ending in state r.

Two-way automata as functors

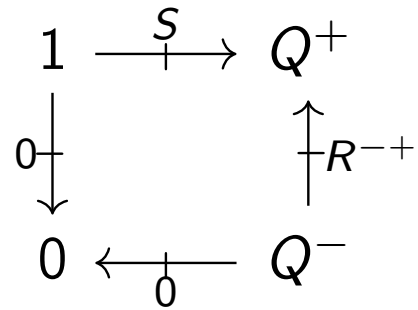
In $\mathbb{I}\mathrm{nt}\mathrm{Rel}$, a morphism $(1,0) \to (Q^+, Q^-)$ corresponds to a square of relations



i.e. is given by a subset $S \subset A^+$ *and* a relation $R^{-+} : Q^- \nrightarrow Q^+$.

Two-way automata as functors

In $\mathbb{I}\mathrm{nt}\mathrm{Rel}$, a morphism $(1,0) \to (Q^+, Q^-)$ corresponds to a square of relations



i.e. is given by a subset $S \subset A^+$ *and* a relation $R^{-+} : Q^- \nrightarrow Q^+$. If R^{-+} is empty, it gives a subset of initial states; similarly,

$$R : (Q^+, Q^-) \to (1,0)$$

such that $R^{+-} = 0$ gives a subset of accepting states.

Two-way automata as functors

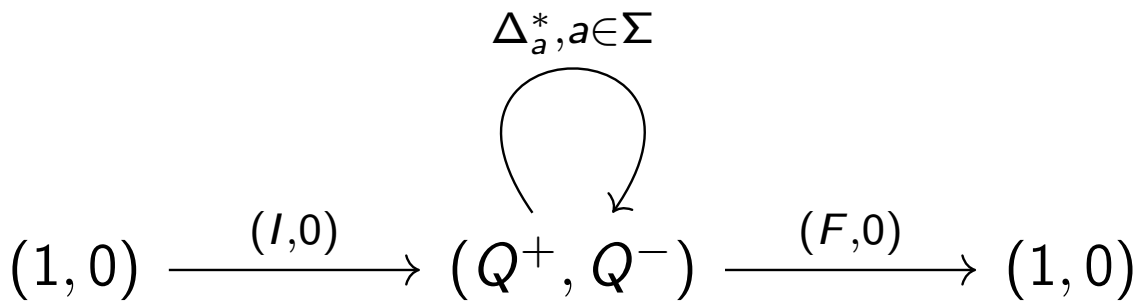
Theorem

Two-way automata correspond surjectively to bipointed functors

$$\mathcal{A} : (\mathcal{I}_{\Sigma^*}, \mathrm{in}, \mathrm{out}) \to (\mathbb{I}\mathrm{ntRel}, (1,0), (1,0))$$

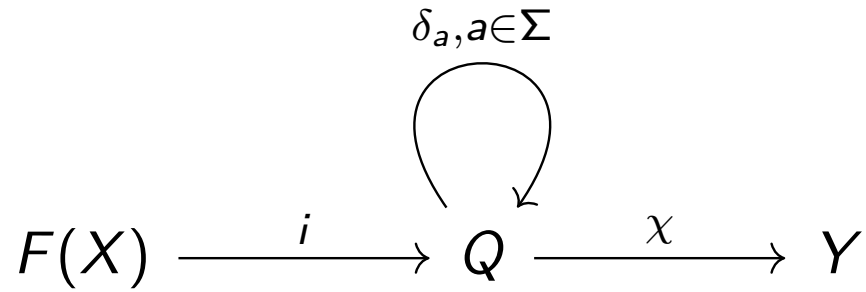
such that $\mathcal{A}(\triangleright)^{-+} = 0$ *and* $\mathcal{A}(\triangleleft)^{+-} = 0$.

The correspondence sends (Q^+, Q^-, I, F, Δ) to

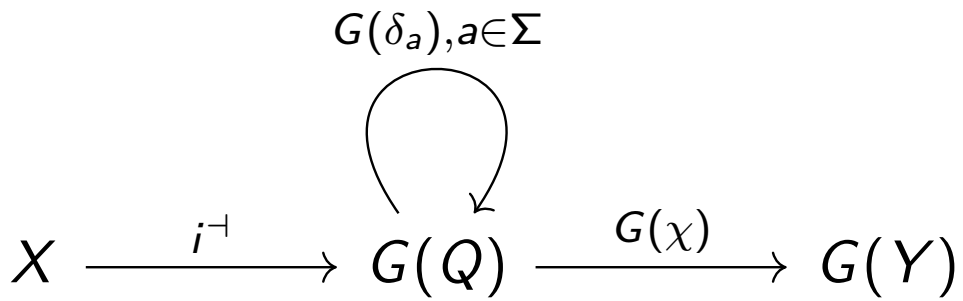


Lifting some pointwise adjunctions

If $F \dashv G : \mathscr{D} \to \mathscr{E}$ is an adjunction, sending an automaton



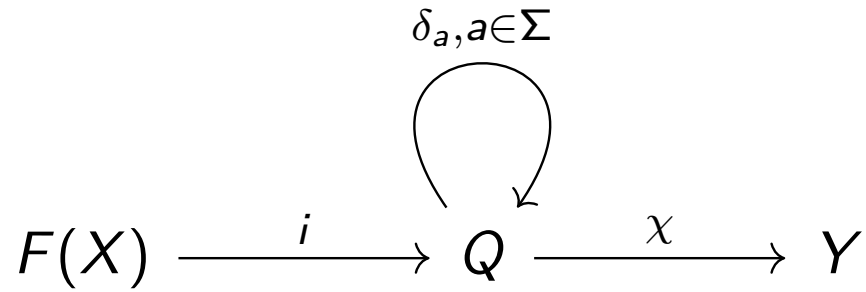
to the automaton



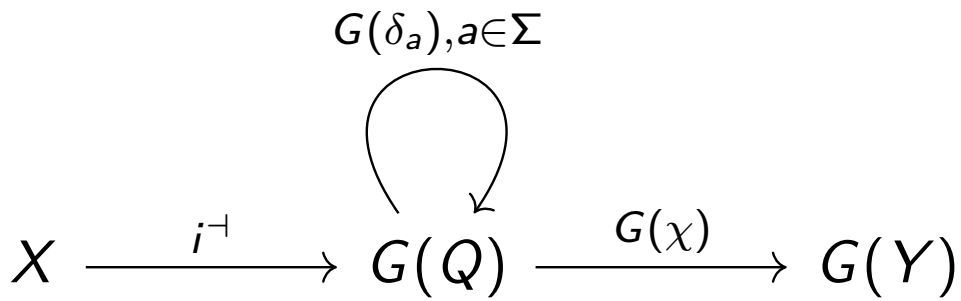
defines a functor $\mathrm{Auto}(L) \to \mathrm{Auto}(L^\dashv)$ which admits a left adjoint.

Lifting some pointwise adjunctions

If $F \dashv G : \mathscr{D} \to \mathscr{E}$ is an adjunction, sending an automaton



to the automaton



defines a functor $\mathrm{Auto}(L) \to \mathrm{Auto}(L^\dashv)$ which admits a left adjoint. In fact, the definition of this functor $\mathrm{Auto}(L) \to \mathrm{Auto}(L^\dashv)$ only requires the unit at X i.e. only an object X' and an isomorphism

$$\mathscr{D}(X', Q) \cong \mathscr{E}(X, G(Q))$$

natural in Q.

Application: the Shepherdson construction

We have

$$\mathbb{I}\mathrm{ntRel}((1,0),-) \cong \mathrm{Set}(\mathbb{1}, \mathbb{I}\mathrm{ntRel}((1,0),-))$$

so we obtain[1] a functor sending a two-way automaton

$$(1,0) \xrightarrow{(I,0)} (Q^+, Q^-) \xrightarrow{(F,0)} (1,0) \qquad \Delta_a^*, a \in \Sigma$$

to the deterministic automaton

$$\mathbb{1} \xrightarrow{(I,0)^\dashv} \mathrm{Rel}(1 \oplus Q^-, Q^+) \xrightarrow{\mathbb{I}\mathrm{ntRel}((1,0),(F,0))} \Omega \qquad \mathbb{I}\mathrm{ntRel}((1,0),\Delta_a^*), a \in \Sigma$$

[1] Idea of Tito Nguyen !

Table of Contents

Definition

The alphabet is now graded i.e. Σ is endowed with an arity function $|\cdot| : \Sigma \to \mathbb{N}$ so that letters are now *functional symbols*. Words are replaced by trees or terms over Σ.

Definition

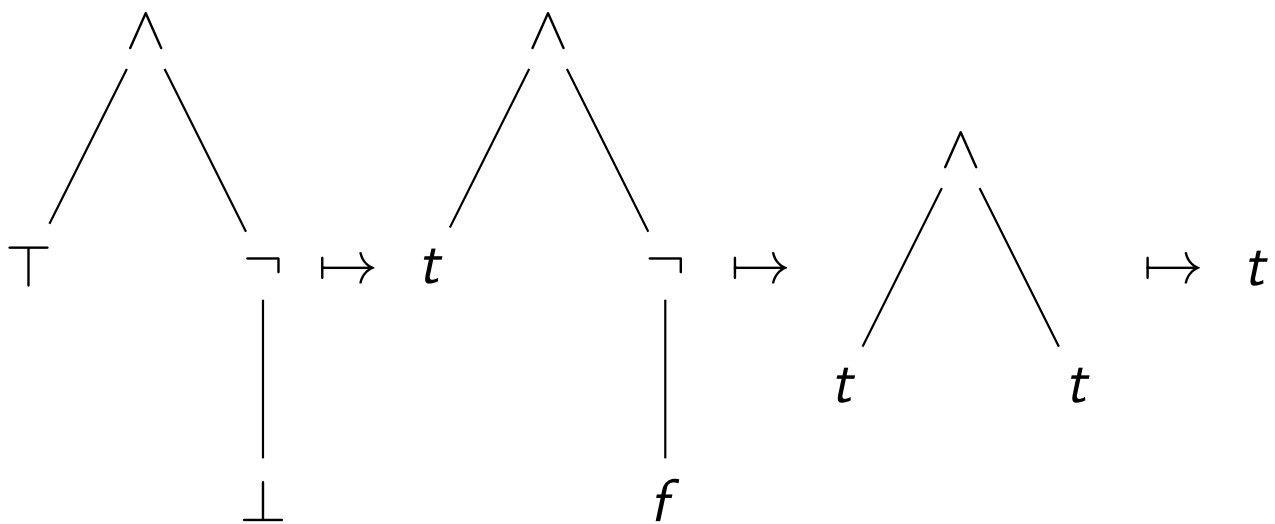
The alphabet is now graded i.e. Σ is endowed with an arity function $|\cdot| : \Sigma \to \mathbb{N}$ so that letters are now *functional symbols*. Words are replaced by trees or terms over Σ.
A *deterministic, bottom-up tree automaton* is given by a set of states Q, for each symbol $f \in \Sigma$ a "rewriting rule" function $\delta_f : Q^{|f|} \to Q$, and a subset of accepting states $F \subset Q$.

Definition

The alphabet is now graded i.e. Σ is endowed with an arity function $|\cdot| : \Sigma \to \mathbb{N}$ so that letters are now *functional symbols*. Words are replaced by trees or terms over Σ.
A *deterministic, bottom-up tree automaton* is given by a set of states Q, for each symbol $f \in \Sigma$ a "rewriting rule" function $\delta_f : Q^{|f|} \to Q$, and a subset of accepting states $F \subset Q$.

Example

Consider $\Sigma := \{\top : 0, \bot : 0, \vee : 2, \wedge : 2, \neg : 1\}$. Let $Q = \{t, f\}$ with the usual Boolean algebra structure, and $F = \{t\} \subset Q$.



Lawvere theories and models

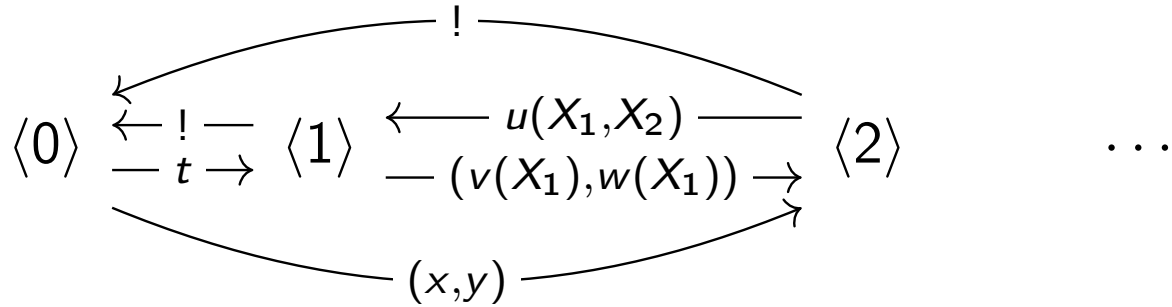
A *tree automaton* is exactly a Σ-algebra with a distinguished subset F.

Lawvere theories and models

A *tree automaton* is exactly a Σ-algebra with a distinguished subset F.
According to Lawvere, a Σ-algebra corresponds to a monoidal functor

$$\mathscr{L}_{(\Sigma,\emptyset)} \to \mathrm{Set}$$

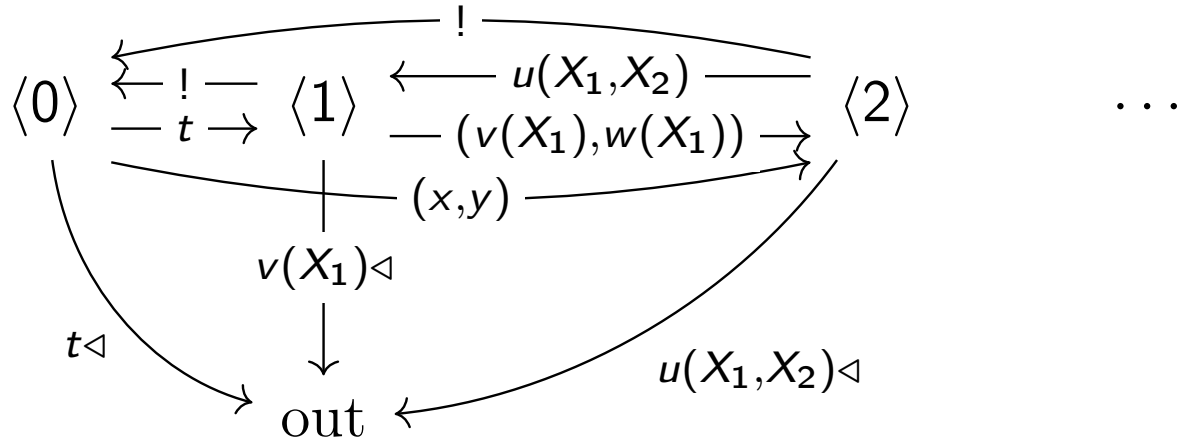
where $\mathscr{L}_{(\Sigma,\emptyset)}$ is the opposite of the category of finitely freely generated Σ-algebras:



i.e. $\mathscr{L}_{(\Sigma,\emptyset)}(\langle m\rangle, \langle n\rangle)$ is the set of n-tuples of trees/terms with m free variables. In fact, $\langle n\rangle \cong \langle 1\rangle^n$ in $\mathscr{L}_{(\Sigma,\emptyset)}$.

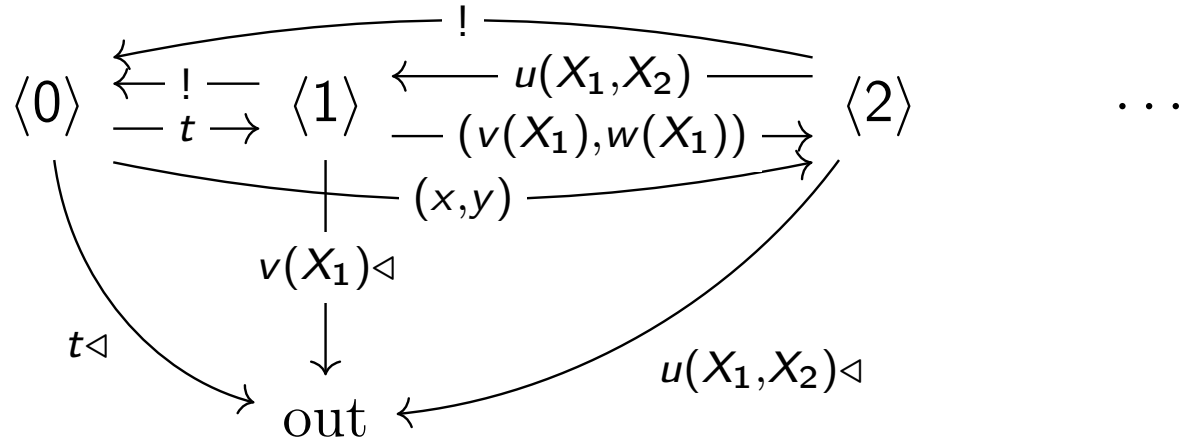
Tree automata as (some) functors

We freely add an object out to $\mathscr{L}_{(\Sigma,\emptyset)}$ to form the category $\mathcal{T}_\Sigma$ with $\mathcal{T}_\Sigma(\langle n\rangle, \mathrm{out}) = \mathscr{L}_{(\Sigma,\emptyset)}(\langle n\rangle, \langle 1\rangle)$, $\mathcal{T}_\Sigma(\mathrm{out}, \langle n\rangle) = \emptyset$:



Tree automata as (some) functors

We freely add an object out to $\mathscr{L}_{(\Sigma,\emptyset)}$ to form the category $\mathcal{T}_\Sigma$ with $\mathcal{T}_\Sigma(\langle n\rangle, \mathrm{out}) = \mathscr{L}_{(\Sigma,\emptyset)}(\langle n\rangle, \langle 1\rangle)$, $\mathcal{T}_\Sigma(\mathrm{out}, \langle n\rangle) = \emptyset$:



Theorem

Tree automata over Σ are in bijective correspondence with pointed functors

$$\mathcal{A} : (\mathcal{T}_\Sigma, \mathrm{out}) \to (\mathrm{Set}, \Omega)$$

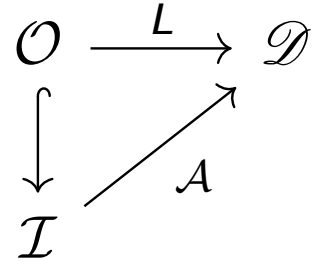
such that

$$\mathscr{L}_{(\Sigma,\emptyset)} \hookrightarrow \mathcal{T}_\Sigma \xrightarrow{\mathcal{A}} \mathrm{Set}$$

is monoidal.

General idea of minimisation in the functorial framework

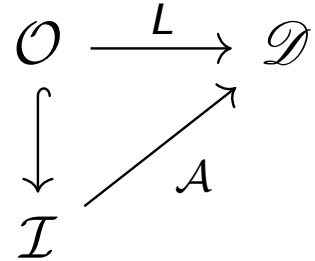
Recall the general framework



where an automaton morphism is a natural transformation that is the identity of L when restricted to $\mathcal{O}$.

General idea of minimisation in the functorial framework

Recall the general framework

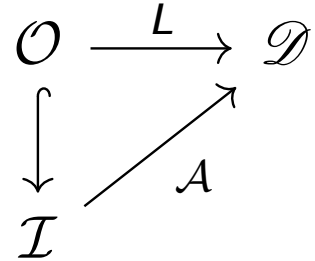


where an automaton morphism is a natural transformation that is the identity of L when restricted to $\mathcal{O}$.
To minimise we have to

1. Find the initial $\emptyset(L)$ and terminal $\mathbb{1}(L)$ automata, for instance as left resp. right Kan extensions.

General idea of minimisation in the functorial framework

Recall the general framework

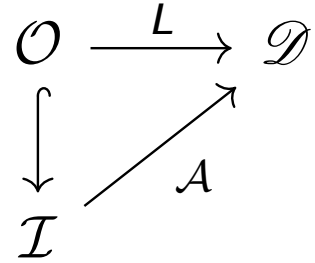


where an automaton morphism is a natural transformation that is the identity of L when restricted to $\mathcal{O}$.
To minimise we have to

1. Find the initial $\emptyset(L)$ and terminal $\mathbb{1}(L)$ automata, for instance as left resp. right Kan extensions.
2. Factorise (pointwise) the unique automaton morphism $\emptyset(L) \Rightarrow \mathbb{1}(L)$; the image-functor $\mathrm{Min}(L)$ is actually an L-automaton that is a subquotient of any other L-automaton.

General idea of minimisation in the functorial framework

Recall the general framework



where an automaton morphism is a natural transformation that is the identity of L when restricted to $\mathcal{O}$.
To minimise we have to

1. Find the initial $\emptyset(L)$ and terminal $\mathbb{1}(L)$ automata, for instance as left resp. right Kan extensions.
2. Factorise (pointwise) the unique automaton morphism $\emptyset(L) \Rightarrow \mathbb{1}(L)$; the image-functor $\mathrm{Min}(L)$ is actually an L-automaton that is a subquotient of any other L-automaton.

$\mathrm{Min}(L)$ is then called *the* minimal L-automaton.

Application: minimisation of tree automata

Fix a tree language

$$L : (\mathcal{S}_\Sigma, \langle 0 \rangle, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

where $\mathcal{S}_\Sigma$ is the full subcategory of $\mathcal{T}_\Sigma$ generated by $\langle 0 \rangle$ and out so that

$$\mathcal{S}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathcal{T}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathscr{L}_\Sigma(\langle 0 \rangle, \langle 1 \rangle) = \{(\mathrm{closed})\ \mathrm{trees}\}$$

Application: minimisation of tree automata

Fix a tree language

$$L : (\mathcal{S}_\Sigma, \langle 0 \rangle, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

where $\mathcal{S}_\Sigma$ is the full subcategory of $\mathcal{T}_\Sigma$ generated by $\langle 0 \rangle$ and out so that

$$\mathcal{S}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathcal{T}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathscr{L}_\Sigma(\langle 0 \rangle, \langle 1 \rangle) = \{(\text{closed}) \text{ trees}\}$$

Then we check that

1. Both Kan extensions exist, but only the left one is monoidal when restricted to $\mathscr{L}_{(\Sigma,\emptyset)}$. Thus, the minimal L-automaton $\mathrm{Min}(L)$ exists.

Application: minimisation of tree automata

Fix a tree language

$$L : (\mathcal{S}_\Sigma, \langle 0 \rangle, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

where $\mathcal{S}_\Sigma$ is the full subcategory of $\mathcal{T}_\Sigma$ generated by $\langle 0 \rangle$ and out so that

$$\mathcal{S}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathcal{T}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathscr{L}_\Sigma(\langle 0 \rangle, \langle 1 \rangle) = \{(\text{closed}) \text{ trees}\}$$

Then we check that

1. Both Kan extensions exist, but only the left one is monoidal when restricted to $\mathscr{L}_{(\Sigma,\emptyset)}$. Thus, the minimal L-automaton $\mathrm{Min}(L)$ exists.
2. $\mathrm{Min}(L)$ is monoidal when restricted to $\mathscr{L}_{(\Sigma,\emptyset)}$.

Application: minimisation of tree automata

Fix a tree language

$$L : (\mathcal{S}_\Sigma, \langle 0 \rangle, \mathrm{out}) \to (\mathrm{Set}, \mathbb{1}, \Omega)$$

where $\mathcal{S}_\Sigma$ is the full subcategory of $\mathcal{T}_\Sigma$ generated by $\langle 0 \rangle$ and out so that

$$\mathcal{S}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathcal{T}_\Sigma(\langle 0 \rangle, \mathrm{out}) = \mathscr{L}_\Sigma(\langle 0 \rangle, \langle 1 \rangle) = \{(\mathrm{closed})\ \mathrm{trees}\}$$

Then we check that

1. Both Kan extensions exist, but only the left one is monoidal when restricted to $\mathscr{L}_{(\Sigma,\emptyset)}$. Thus, the minimal L-automaton $\mathrm{Min}(L)$ exists.
2. $\mathrm{Min}(L)$ is monoidal when restricted to $\mathscr{L}_{(\Sigma,\emptyset)}$.

Therefore there is a tree automaton $\mathrm{Min}(L)$ which divides any other tree automaton recognising L.

References I

[Ari23] Quentin Aristote. *Functorial Approach to Minimizing and Learning Deterministic Transducers with Outputs in Arbitrary Monoids*. Nov. 2023. URL: https://ens.hal.science/hal-04172251.

[Bar24] Killian Barbé. "Learning Functorial Automata in Topoi". MA thesis. Université Paris Cité, 2024. URL: https://www.irif.fr/~kbarbe/files/msc.pdf.

[Bir89] Jean-Camille Birget. "Concatenation of Inputs in a Two-Way Automaton". In: *Theoretical Computer Science* 63.2 (Feb. 1989), pp. 141–156. ISSN: 0304-3975. DOI: 10.1016/0304-3975(89)90075-3.

[CP20] Thomas Colcombet and Daniela Petrişan. "Automata Minimization: A Functorial Approach". In: *Logical Methods in Computer Science* 16.1 (Mar. 2020), Issue 1, 18605974. ISSN: 1860-5974. DOI: 10.23638/LMCS-16(1:32)2020.

References II

[CPS21] Thomas Colcombet, Daniela Petrişan, and Riccardo Stabile. "Learning Automata and Transducers: A Categorical Approach". In: *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*. Vol. 183. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 15:1–15:17. ISBN: 978-3-95977-175-7. DOI: 10.4230/LIPIcs.CSL.2021.15.

[Hin03] Peter Hines. "A Categorical Framework for Finite State Machines". In: *Mathematical Structures in Computer Science* 13.3 (June 2003), pp. 451–480. ISSN: 1469-8072, 0960-1295. DOI: 10.1017/S0960129503003931.

References III

[Iwa24] Victor Iwaniack. "Automata in W-Toposes, and General Myhill-Nerode Theorems". In: *Coalgebraic Methods in Computer Science*. Cham: Springer Nature Switzerland, 2024, pp. 93–113. ISBN: 978-3-031-66438-0. DOI: 10.1007/978-3-031-66438-0_5.

[Iwa25] Victor Iwaniack. "Automates Topossiques". These de Doctorat. Université Côte d'Azur, June 2025. URL: https://theses.fr/2025COAZ5026.

[JSV96] André Joyal, Ross Street, and Dominic Verity. "Traced Monoidal Categories". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 119 (Apr. 1996), pp. 447–468. DOI: 10.1017/S0305004100074338.

Perspectives

- Enrich everything; in particular, study nominal two-way automata and nominal tree automata.
- For the former this means enriching the $\mathbb{I}\mathrm{nt}$ construction from Joyal, Street, and Verity [JSV96].
- For the latter it means finding a good notion of finitary signature in a topos (link with monads with arity?).