Minimization of subsequential transducers

Daniela Petrişan
joint work with Thomas Colcombet

ANR Delta
March 28 2018, Paris

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

In this talk

- Word automata in a category
- Subsequential transducers in a category
- Minimization in this setting: sufficient conditions
- A unifying framework for canonical recognizers
    - minimization of deterministic and weighted automata
    - minimization of subsequential transducers à la Choffrut
    - syntactic monoid (more generally syntactic algebra)
    - syntactic Boolean space with an internal monoid

[Colcombet, P., CALCO 2017]
Automata Minimization: a Functorial Approach

Word automata in a category

Word automata

deterministic automata $$\begin{tikzcd} 1 \arrow[r] & Q \arrow[loop above, "a"] \arrow[r] & 2 \end{tikzcd}$$ in Set

Word automata

deterministic automata
$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 2 \end{tikzcd}$$
in Set

non-deterministic automata
$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$
in Rel

Word automata

deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 2 \end{tikzcd}$$

in Set

non-deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$

in Rel

weighted automata

$$\begin{tikzcd} K \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & K \end{tikzcd}$$

in Vec_K

Word automata

deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 2 \end{tikzcd}$$

in Set

non-deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$

in Rel

weighted automata

$$\begin{tikzcd} K \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & K \end{tikzcd}$$

in Vec_K

Subseq. transducers

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$

in $\mathrm{Kl}(\mathcal{T})$

Word automata

deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 2 \end{tikzcd}$$

in Set

non-deterministic automata

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$

in Rel

weighted automata

$$\begin{tikzcd} K \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & K \end{tikzcd}$$

in Vec_K

Subseq. transducers

$$\begin{tikzcd} 1 \arrow[r] & Q \arrow[r] \arrow[loop above, "a"] & 1 \end{tikzcd}$$

in $\mathrm{Kl}(\mathcal{T})$

We see a pattern emerging!

Languages accepted by word automata

deterministic automata	$1 \xrightarrow{f\circ\delta_w\circ i} 2$	in Set
non-deterministic automata	$1 \xrightarrow{f\circ\delta_w\circ i} 1$	in Rel
weighted automata	$K \xrightarrow{f\circ\delta_w\circ i} K$	in Vec$_K$
Subseq. transducers	$1 \xrightarrow{f\circ\delta_w\circ i} 1$	in Kl($\mathcal{T}$)

Word automata in a category

For objects I and F in a category $\mathcal{C}$, a $(\mathcal{C}, I, F)$-automaton is a tuple $\mathcal{A} = \langle Q, i, f, (\delta_a)_{a \in A} \rangle$, where

- Q is an object of $\mathcal{C}$.
- $i \colon I \to Q$ is the «initial» arrow
- $f \colon Q \to F$ is the «final» arrow
- $\delta_a \colon Q \to Q$ is the «transition» arrow for each $a \in A$

Word automata in a category

For objects I and F in a category $\mathcal{C}$, a $(\mathcal{C}, I, F)$-automaton is a tuple $\mathcal{A} = \langle Q, i, f, (\delta_a)_{a \in A} \rangle$, where

- Q is an object of $\mathcal{C}$.
- $i: I \to Q$ is the «initial» arrow
- $f: Q \to F$ is the «final» arrow
- $\delta_a: Q \to Q$ is the «transition» arrow for each $a \in A$

The language accepted by $\mathcal{A}$ is a map $L_{\mathcal{A}}: A^* \to \mathcal{C}(I, F)$ that associates to a word $w = a_1 \dots a_n$ the composite morphism

$$\begin{tikzcd} I \arrow[r, "i"] & Q \arrow[r, "\delta_{a_1}"] & Q \arrow[r, "\delta_{a_2}"] & \dots & {} \arrow[r, "\delta_{a_n}"] & Q \arrow[r, "f"] & F \end{tikzcd}$$

Word automata in a category

For objects I and F in a category $\mathcal{C}$, a $(\mathcal{C}, I, F)$-automaton is a tuple $\mathcal{A} = \langle Q, i, f, (\delta_a)_{a \in A} \rangle$, where

- Q is an object of $\mathcal{C}$.
- $i: I \to Q$ is the «initial» arrow
- $f: Q \to F$ is the «final» arrow
- $\delta_a: Q \to Q$ is the «transition» arrow for each $a \in A$

The language accepted by $\mathcal{A}$ is a map $L_{\mathcal{A}}: A^* \to \mathcal{C}(I, F)$ that associates to a word $w = a_1 \ldots a_n$ the composite morphism

$$\begin{tikzcd} I \arrow[r, "i"] & Q \arrow[r, "\delta_{a_1}"] & Q \arrow[r, "\delta_{a_2}"] & \ldots \arrow[r, "\delta_{a_n}"] & Q \arrow[r, "f"] & F \end{tikzcd}$$

Example

A DFA is a $(\mathrm{Set}, 1, 2)$-automaton. It accepts a language $L: A^* \to \mathrm{Set}(1, 2) \cong 2$.

Subsequential transducers in a category

Subsequential transducers

A subsequential transducer with input alphabet A and output alphabet B consists of:

- a finite set of states Q
- an initial state with an initial output in B^*, or an undefined initial state

Subsequential transducers

A subsequential transducer with input alphabet A and output alphabet B consists of:

- a finite set of states Q
- an initial state with an initial output in B^*, or an undefined initial state
- for each $a \in A$ a transition function $Q \to B^* \times Q + 1$

Subsequential transducers

A subsequential transducer with input alphabet A and output alphabet B consists of:

- a finite set of states Q
- an initial state with an initial output in B^*, or an undefined initial state
- for each $a \in A$ a transition function $Q \to B^* \times Q + 1$
- for each state in Q, either an output word in B^* or undefined.

The output category for subsequential transducers

We consider partial actions for the free monoid B^*.

The output category for subsequential transducers

We consider partial actions for the free monoid B^*.

We consider a category $\mathsf{Kl}(\mathcal{T})$ with

- objects: sets $X, Y, Z, \ldots$
- arrows: $f\colon X \nrightarrow Y$, where $f\colon X \to B^* \times Y + 1$ is a function

The output category for subsequential transducers

We consider partial actions for the free monoid B^*.

We consider a category $\mathrm{Kl}(\mathcal{T})$ with

- objects: sets $X, Y, Z, \ldots$
- arrows: $f : X \nrightarrow Y$, where $f : X \to B^* \times Y + 1$ is a function

Composition of arrows in $\mathrm{Kl}(\mathcal{T})$ is defined using the monoid multiplication in B^*.

If $f : X \nrightarrow Y$ and $g : Y \nrightarrow Z$ then $g \circ f : X \nrightarrow Z$ (i.e. $g \circ f : X \to B^* \times Z + 1$) is given by $g \circ f(x) = \begin{cases} (uv, z) & \text{if } f(x) = (u, y) \text{ and } g(y) = (v, z) \\ \perp & \text{otherwise.} \end{cases}$

The output category for subsequential transducers

We consider partial actions for the free monoid B^*.

We consider a category $\mathsf{Kl}(\mathcal{T})$ with

- objects: sets $X, Y, Z, \ldots$
- arrows: $f\colon X \nrightarrow Y$, where $f\colon X \to B^* \times Y + 1$ is a function

Composition of arrows in $\mathsf{Kl}(\mathcal{T})$ is defined using the monoid multiplication in B^*.

If $f\colon X \nrightarrow Y$ and $g\colon Y \nrightarrow Z$ then $g \circ f\colon X \nrightarrow Z$ (i.e. $g \circ f\colon X \to B^* \times Z + 1$) is given by

$$g \circ f(x) = \begin{cases} (uv, z) & \text{if } f(x) = (u, y) \text{ and } g(y) = (v, z) \\ \perp & \text{otherwise.} \end{cases}$$

This is the Kleisli category for the monad $\mathcal{T}\colon \mathsf{Set} \to \mathsf{Set}$ given by $\mathcal{T}(X) = B^* \times X + 1$, which associates to each set X the free partial action of B^* on X. Notice that we can replace B^* with any other monoid.

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd} 1 \arrow[r, "i"] & Q \arrow[loop above, "\delta_a"] \arrow[r, "f"] & 1 \end{tikzcd} \qquad \text{in } \mathsf{Kl}(\mathcal{T})$$

ammounts to give

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd}
1 \arrow[r, "i"] & Q \arrow[r, "f"] \arrow[loop above, "\delta_a"] & 1 & & \text{in } \mathsf{Kl}(\mathcal{T})
\end{tikzcd}$$

ammounts to give

- a function $i \colon 1 \to B^* \times Q + 1$, i.e. an initial state with an initial output in B^*, or an undefined initial state

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd} 1 \arrow[r, "i"] & Q \arrow[r, "f"] \arrow[loop above, "\delta_a"] & 1 \end{tikzcd} \qquad \text{in } \mathrm{Kl}(\mathcal{T})$$

ammounts to give

- a function $i\colon 1 \to B^* \times Q + 1$, i.e. an initial state with an initial output in B^*, or an undefined initial state
- for each $a \in A$ a function $\delta_a\colon Q \to B^* \times Q + 1$

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd}
1 \arrow[r, "i"] & Q \arrow[r, "f"] \arrow["\delta_a", loop above] & 1
\end{tikzcd}$$

in $\mathsf{Kl}(\mathcal{T})$

ammounts to give

- a function $i\colon 1 \to B^* \times Q + 1$, i.e. an initial state with an initial output in B^*, or an undefined initial state
- for each $a \in A$ a function $\delta_a\colon Q \to B^* \times Q + 1$
- a final map $f\colon Q \to B^* \times 1 + 1$, i.e. for each state in Q either an output word in B^* or undefined.

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd} 1 \arrow[r, "i"] & Q \arrow[loop above, "\delta_a"] \arrow[r, "f"] & 1 \end{tikzcd} \qquad \text{in } \mathrm{Kl}(\mathcal{T})$$

ammounts to give a subsequential transducer!

The output category for subsequential transducers

Interpretting the arrows

$$\begin{tikzcd} 1 \arrow[r, "i"] & Q \arrow[r, "f"] \arrow[loop above, "\delta_a"] & 1 \end{tikzcd} \qquad \text{in } \mathrm{Kl}(\mathcal{T})$$

ammounts to give a subsequential transducer!

Furthermore, the partial function realized by the corresponding subsequential transducer applied to a word $w \in A^*$ is exactly $f \circ \delta_w \circ i(w)$.

Automata in a category: minimization

Minimzation of $(\mathcal{C}, I, F)$-automata

- What does it mean for a $(\mathcal{C}, I, F)$-automaton to be minimal?
- What are sufficient conditions on $\mathcal{C}$ so that a minimal automaton for a language exists?

Minimzation of $(\mathcal{C}, I, F)$-automata

- What does it mean for a $(\mathcal{C}, I, F)$-automaton to be minimal?
- What are sufficient conditions on $\mathcal{C}$ so that a minimal automaton for a language exists?

A DFA is minimal when it divides any other automaton accepting the same language.

Minimzation of $(\mathcal{C}, I, F)$-automata

- What does it mean for a $(\mathcal{C}, I, F)$-automaton to be minimal?
- What are sufficient conditions on $\mathcal{C}$ so that a minimal automaton for a language exists?

A DFA is minimal when it divides any other automaton accepting the same language. Here divides = «is a quotient of a sub-automaton of»

Minimzation of $(\mathcal{C}, I, F)$-automata

- What does it mean for a $(\mathcal{C}, I, F)$-automaton to be minimal?
- What are sufficient conditions on $\mathcal{C}$ so that a minimal automaton for a language exists?

A DFA is minimal when it divides any other automaton accepting the same language. Here divides = «is a quotient of a sub-automaton of»

Thus we need a notion of «quotient» (surjection for sets) and «sub-object» (injection for sets), i.e. a factorization system.

The three ingredients for minimization

When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

The three ingredients for minimization

When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

If the category of automata accepting $\mathcal{L}$ has

- an initial object $\mathcal{A}_{\mathtt{init}}(\mathcal{L})$,

The three ingredients for minimization

When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

If the category of automata accepting $\mathcal{L}$ has

- an initial object $\mathcal{A}_{\mathtt{init}}(\mathcal{L})$,
- a final object $\mathcal{A}_{\mathtt{final}}(\mathcal{L})$, and,

The three ingredients for minimization

When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

If the category of automata accepting $\mathcal{L}$ has

- an initial object $\mathcal{A}_{\mathtt{init}}(\mathcal{L})$,
- a final object $\mathcal{A}_{\mathtt{final}}(\mathcal{L})$, and,
- a factorization system

then $\mathtt{Min}(\mathcal{L})$ is obtained as the factorization

$$\mathcal{A}_{\mathtt{init}}(\mathcal{L}) \twoheadrightarrow \mathtt{Min}(\mathcal{L}) \rightarrowtail \mathcal{A}_{\mathtt{final}}(\mathcal{L})\,.$$

The three ingredients for minimization

When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

If the category of automata accepting $\mathcal{L}$ has

- an initial object $\mathcal{A}_{\mathtt{init}}(\mathcal{L})$, ✓when $\mathcal{C}$ has copowers
- a final object $\mathcal{A}_{\mathtt{final}}(\mathcal{L})$, and, ✓when $\mathcal{C}$ has powers
- a factorization system ✓when $\mathcal{C}$ has one

then $\mathtt{Min}(\mathcal{L})$ is obtained as the factorization

$$\mathcal{A}_{\mathtt{init}}(\mathcal{L}) \twoheadrightarrow \mathtt{Min}(\mathcal{L}) \rightarrowtail \mathcal{A}_{\mathtt{final}}(\mathcal{L})\,.$$

Trivial example

deterministic automata, i.e. $(\mathsf{Set}, 1, 2)$-automata
accepting a $(\mathsf{Set}, 1, 2)$-language

$$\begin{tikzcd}
 & A^* \arrow[d, "\text{reachedState}" description] \arrow[dr, bend left, "\mathcal{L}?"] & \\
1 \arrow[ur, bend left, "\varepsilon"] \arrow[r, "i"] \arrow[dr, bend right, "\mathcal{L}"'] & Q \arrow[r, "f"] \arrow[d, "\text{acceptedLanguage}" description] & 2 \\
 & 2^{A^*} \arrow[ur, bend right, "\varepsilon?"'] &
\end{tikzcd}$$

Trivial example

deterministic automata, i.e. $(\mathsf{Set}, 1, 2)$-automata
accepting a $(\mathsf{Set}, 1, 2)$-language

$$\begin{tikzcd}
 & A^* \arrow[d, two heads] \arrow[dr, bend left, "\mathcal{L}?"] & \\
1 \arrow[ur, bend left, "\varepsilon"] \arrow[r, "i"] \arrow[dr, bend right, "\mathcal{L}"'] & \mathtt{Min}(\mathcal{L}) \arrow[r, "f"] \arrow[d, tail] & 2 \\
 & 2^{A^*} \arrow[ur, bend right, "\varepsilon?"'] &
\end{tikzcd}$$

Another trivial example

$\mathbb{R}$-weighted automata, i.e. $(\mathsf{Vec}, \mathbb{R}, \mathbb{R})$-automata
accepting a $(\mathsf{Vec}, \mathbb{R}, \mathbb{R})$-language

$$\begin{tikzcd}
 & \bigoplus\limits_{u \in A^*} \mathbb{R} \arrow[dd, "\mathsf{reachedState}" description] \arrow[dr, bend left, "L?"] & \\
\mathbb{R} \arrow[ur, bend left, "\varepsilon"] \arrow[r, "i"] \arrow[dr, bend right, "L"'] & Q \arrow[r, "F"] \arrow[d, "\mathsf{acceptedLanguage}" description] & \mathbb{R} \\
 & \prod\limits_{u \in A^*} \mathbb{R} \arrow[ur, bend right, "\varepsilon?"'] &
\end{tikzcd}$$

Another trivial example

$\mathbb{R}$-weighted automata, i.e. $(\mathsf{Vec}, \mathbb{R}, \mathbb{R})$-automata
accepting a $(\mathsf{Vec}, \mathbb{R}, \mathbb{R})$-language

$$\begin{tikzcd}
 & \bigoplus_{u \in A^*} \mathbb{R} \arrow[d, two heads] \arrow[dr, bend left, "L?"] & \\
\mathbb{R} \arrow[ur, bend left, "\varepsilon"] \arrow[r, "i"] \arrow[dr, bend right, "L"'] & \mathtt{Min}(\mathcal{L}) \arrow[r, "F"] \arrow[d, tail] & \mathbb{R} \\
 & \prod_{u \in A^*} \mathbb{R} \arrow[ur, bend right, "\varepsilon?"'] &
\end{tikzcd}$$

Minimial Automaton $\texttt{Min}(\mathcal{L})$ for a Language

The automaton $\texttt{Min}(\mathcal{L})$ divides any other automaton accepting $\mathcal{L}$.

$$\begin{tikzcd}
& & & \mathcal{A} \arrow[rrrd, bend left] & & & \\
\mathcal{A}_{\texttt{init}}(L) \arrow[rrru, bend left] \arrow[rr, two heads] \arrow[rrrd, bend right, two heads] & & \texttt{reach}(\mathcal{A}) \arrow[ru, tail] \arrow[rr, two heads] & & \texttt{obs}(\texttt{reach}(\mathcal{A})) \arrow[rr, tail] \arrow[ld, dashed, no head] & & \mathcal{A}_{\texttt{final}}(L) \\
& & & \texttt{Min}(L) \arrow[rrru, tail, bend right] & & &
\end{tikzcd}$$

Minimial Automaton $\mathtt{Min}(\mathcal{L})$ for a Language

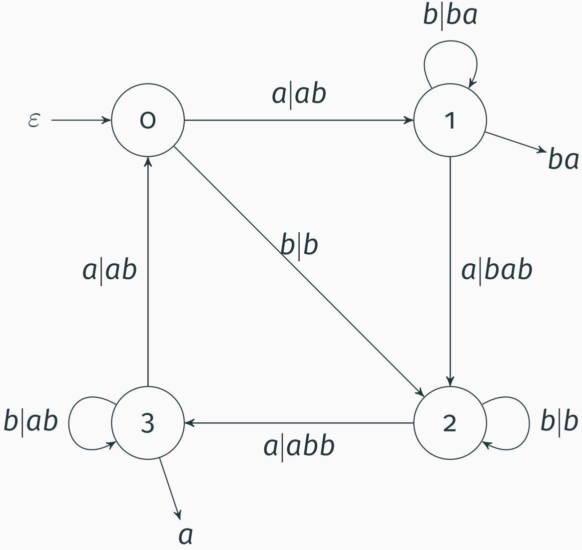
The automaton $\mathtt{Min}(\mathcal{L})$ divides any other automaton accepting $\mathcal{L}$.

$$\begin{tikzcd}
 & & \mathcal{A} & & \\
\mathcal{A}_{\mathtt{init}}(L) \arrow[r, two heads] \arrow[urr, bend left] \arrow[drr, two heads, bend right] & \mathtt{reach}(\mathcal{A}) \arrow[ur, tail] \arrow[rr, two heads] & & \mathtt{obs}(\mathtt{reach}(\mathcal{A})) \arrow[r, tail] \arrow[dl, dashed, no head] & \mathcal{A}_{\mathtt{final}}(L) \\
 & & \mathtt{Min}(L) \arrow[urr, tail, bend right] & &
\end{tikzcd}$$

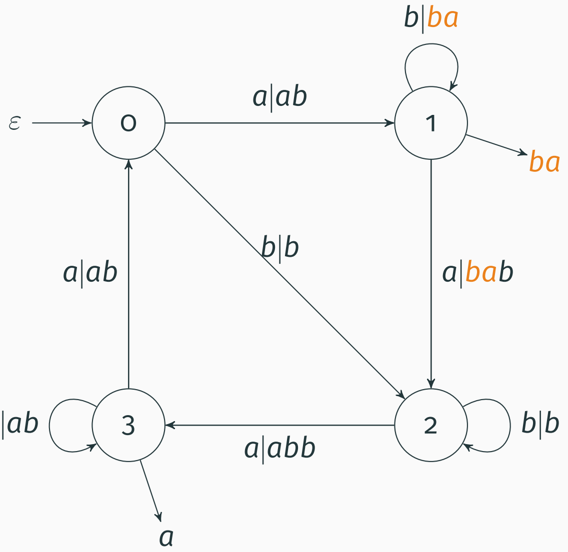
Thus far we identified simple sufficient conditions on $\mathcal{C}$ so that minimization of $\mathcal{C}$-automata is guaranteed!

Minimization of subsequential transducers

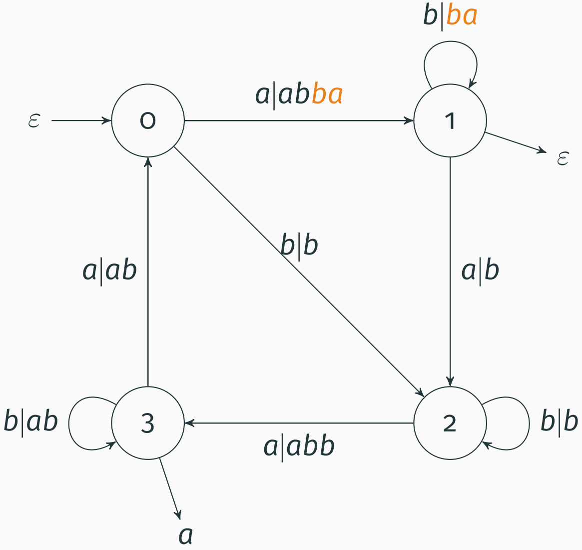
Minimization of subsequential transducers à la Choffrut



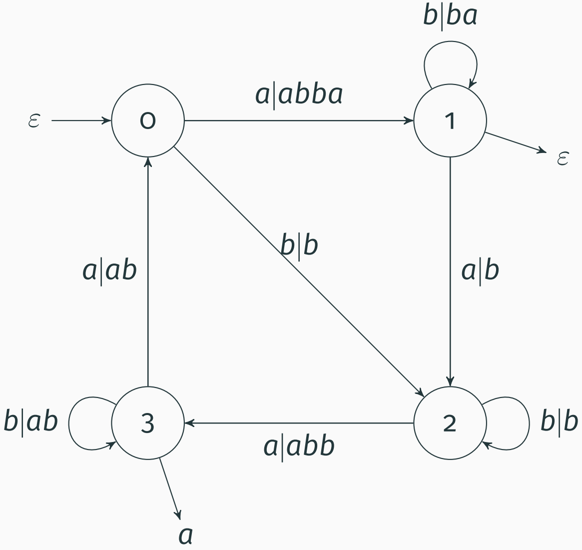
Minimization of subsequential transducers à la Choffrut



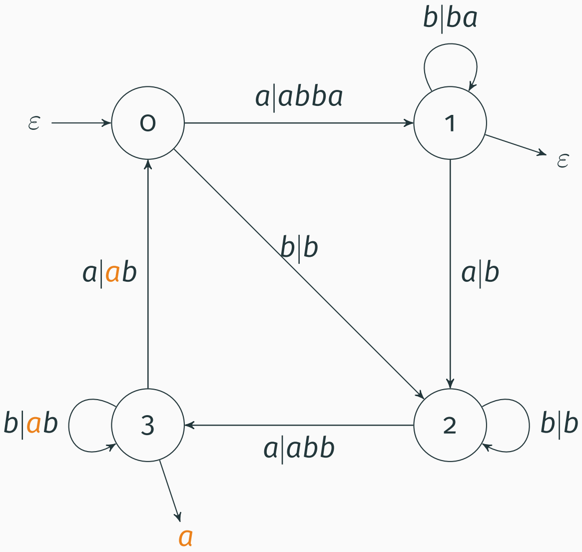
Minimization of subsequential transducers à la Choffrut



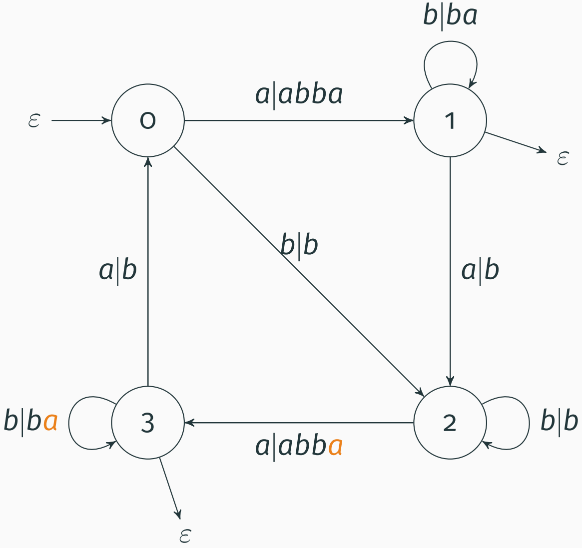
Minimization of subsequential transducers à la Choffrut



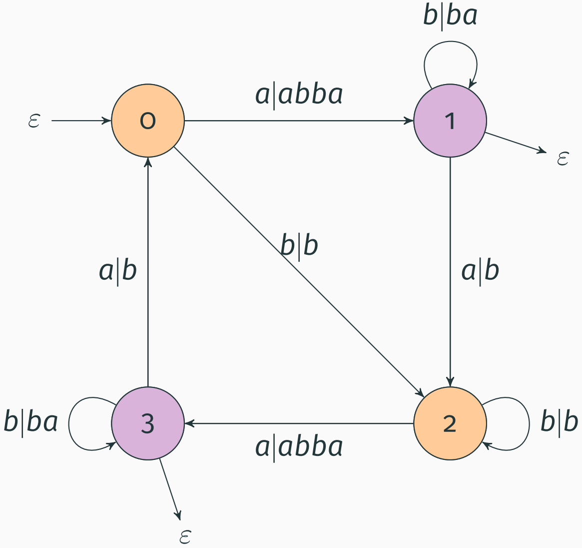
Minimization of subsequential transducers à la Choffrut



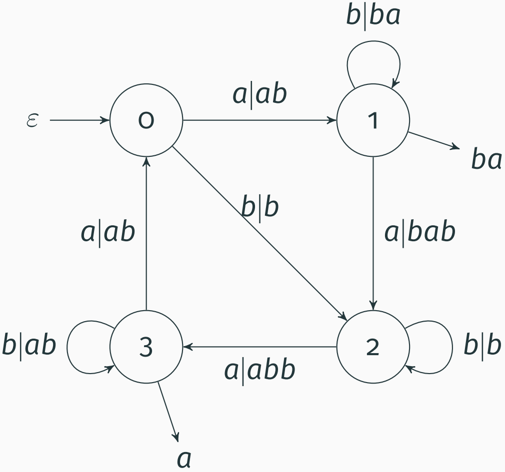
Minimization of subsequential transducers à la Choffrut

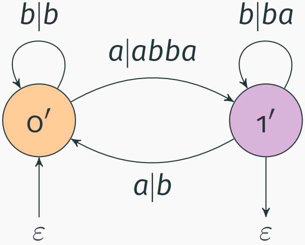


Minimization of subsequential transducers à la Choffrut



Minimization of subsequential transducers à la Choffrut





Minimization of subsequential transducers: the category-theoretic version

Recall that subsequential transducers are word automata interpreted in the category $\mathsf{Kl}(\mathcal{T})$:

- objects: sets $X, Y, Z, \ldots$
- arrows: $f \colon X \nrightarrow Y$ where $f \colon X \to B^* \times Y + 1$ is a function

Does $\mathsf{Kl}(\mathcal{T})$ satisfy the sufficient conditions for minimization?

Minimization of subsequential transducers: the category-theoretic version

Recall that subsequential transducers are word automata interpreted in the category $\mathsf{Kl}(\mathcal{T})$:

- objects: sets $X, Y, Z, \ldots$
- arrows: $f : X \nrightarrow Y$ where $f : X \to B^* \times Y + 1$ is a function

Does $\mathsf{Kl}(\mathcal{T})$ satisfy the sufficient conditions for minimization?

Not quite! It does not have products, powers... so proving the existence of the final automaton for a language is problematic. The latter exists nevertheless in this case.

The ingredients for minimization

- initial automaton
- final automaton
- factorization system

The ingredients for minimization

- initial automaton ✓
- final automaton
- factorization system

Idea: We use the lifting of the Kleisli adjunction for the monad $\mathcal{T}$. The left adjoint preserves the initial automaton.

The set of states of the initial $\mathsf{Kl}(\mathcal{T})$-automaton is A^*

$$\begin{tikzcd}
\mathtt{Auto}(\mathcal{L}_{\mathsf{Set}}) \arrow[r, bend left] \arrow[r, phantom, "\perp"] \arrow[d, "\mathsf{States}" description] & \mathtt{Auto}(\mathcal{L}_{\mathsf{Kl}(\mathcal{T})}) \arrow[l, bend left] \arrow[d, "\mathsf{States}" description] \\
\mathsf{Set} \arrow[r, bend left, "F_{\mathcal{T}}"] \arrow[r, phantom, "\perp"] & \mathsf{Kl}(\mathcal{T}) \arrow[l, bend left, "U_{\mathcal{T}}"]
\end{tikzcd}$$

The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system

Idea: There exists a $\mathsf{Kl}(\mathcal{T})$-automaton mapped by the right adjoint to the final Set-automaton.
The set of states of the final $\mathsf{Kl}(\mathcal{T})$-automaton is $\mathsf{Irr}(A^*, B^*)$ – the set of partial functions
$f : A^* \to B^* + 1$ such that

- f is defined on some word in A^* and
- the longest common prefix of $\{f(w) \mid f(w) \in B^*\}$ is ϵ.

The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system

Idea: There exists a $\mathrm{Kl}(\mathcal{T})$-automaton mapped by the right adjoint to the final Set-automaton.

The set of states of the final $\mathrm{Kl}(\mathcal{T})$-automaton is $\mathrm{Irr}(A^*, B^*)$ – the set of partial functions

$f\colon A^* \to B^* + 1$ such that

- f is defined on some word in A^* and
- the longest common prefix of $\{f(w) \mid f(w) \in B^*\}$ is ϵ.

Crucial fact: $B^* \times \mathrm{Irr}(A^*, B^*) \cong (A^* + 1)^{B^*}$.

Longest common prefixes play a fundamental role.

The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system ✓

Idea: The factorization system is inherited from a factorization system $(\mathcal{E}, \mathcal{M})$ on $\mathrm{Kl}(\mathcal{T})$:

- $e : X \nrightarrow Y$ (i.e. $e : X \to B^* \times Y + 1$) is in $\mathcal{E}$ iff each $y \in Y$ is in the image of the second projection of e.
- $m : X \nrightarrow Y$ (i.e. $m : X \to B^* \times Y + 1$) is in $\mathcal{M}$ iff m is everywhere defined, the second projection is injective and the first projection is constant ε.

The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system ✓

Idea: The factorization system is inherited from a factorization system $(\mathcal{E}, \mathcal{M})$ on $\mathsf{Kl}(\mathcal{T})$:

- $e{:}X \nrightarrow Y$ (i.e. $e{:}X \to B^* \times Y + 1$) is in $\mathcal{E}$ iff each $y \in Y$ is in the image of the second projection of e.
- $m{:}X \nrightarrow Y$ (i.e. $m{:}X \to B^* \times Y + 1$) is in $\mathcal{M}$ iff m is everywhere defined, the second projection is injective and the first projection is constant ε.

This also works if we replace B^* by a right cancellative monoid.

The minimal transducer in a picture

We obtain $\mathtt{Min}(\mathcal{L})$ – the minimal subsequential transducer as obtained by Choffrut!

$$\begin{tikzcd}
 & A^* \arrow[d, two heads] \arrow[dr, bend left, "\mathcal{L}?"] & \\
1 \arrow[ur, bend left, "\varepsilon"] \arrow[r, "i"] \arrow[dr, bend right, "\mathcal{L}"'] & \mathtt{Min}(\mathcal{L}) \arrow[r, "f"] \arrow[d, tail] & 1 \\
 & \mathrm{Irr}(A^*, B^*) \arrow[ur, bend right, "\varepsilon?"'] &
\end{tikzcd}$$

Conclusions

We put under the same umbrella concepts like

- minimal DFA,
- syntactic monoid/algebras,
- minimal subsequential transducers (à la Choffrut)?
- new forms of automata: minimal hybrid-set-vector automata (see [MFCS'17]),

Conclusions

We put under the same umbrella concepts like

- minimal DFA,
- syntactic monoid/algebras,
- minimal subsequential transducers (à la Choffrut)?
- new forms of automata: minimal hybrid-set-vector automata (see [MFCS'17]),

Conclusions

We put under the same umbrella concepts like

- minimal DFA,
- syntactic monoid/algebras,
- minimal subsequential transducers (à la Choffrut)?
- new forms of automata: minimal hybrid-set-vector automata (see [MFCS'17]),

What next

- Transducers with outputs in an arbitrary monoid? See next talk...
- Algebras for recognition beyond Set?
- Learning and minimization? Generic learning algorithms?