Minimization of subsequential transducers

Daniela Petrișan
joint work with Thomas Colcombet
ANR Delta
March 28 2018, Paris
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

\[
1 \rightarrow Q \rightarrow 2 \quad \text{in Set}
\]
We see a pattern emerging.

**Deterministic Automata**

\[ 1 \rightarrow Q \rightarrow 2 \]

in Set

\[ Q \rightarrow a \rightarrow Q \]

**Non-deterministic Automata**

\[ 1 \rightarrow Q \rightarrow 1 \]

in Rel
Word automata

- Deterministic automata
  - Transition: $1 \rightarrow Q \rightarrow 2$ in Set

- Non-deterministic automata
  - Transition: $1 \rightarrow Q \rightarrow 1$ in Rel

- Weighted automata
  - Transition: $K \rightarrow Q \rightarrow K$ in Vec$_K$
Word automata

- Deterministic automata
  - State transition: $1 \rightarrow Q \rightarrow 2$ in Set

- Non-deterministic automata
  - State transition: $1 \rightarrow Q \rightarrow 1$ in Rel

- Weighted automata
  - State transition: $K \rightarrow Q \rightarrow K$ in $\text{Vec}_K$

- Subsequence transducers
  - State transition: $1 \rightarrow Q \rightarrow 1$ in $\text{Kl}(\mathcal{T})$
Word automata

- Deterministic automata: $1 \rightarrow Q \rightarrow 2$ in Set
- Non-deterministic automata: $1 \rightarrow Q \rightarrow 1$ in Rel
- Weighted automata: $K \rightarrow Q \rightarrow K$ in $\text{Vec}_K$
- Subseq. transducers: $1 \rightarrow Q \rightarrow 1$ in $\text{Kl}(\mathcal{T})$

We see a pattern emerging!
## Languages accepted by word automata

<table>
<thead>
<tr>
<th>Automata Type</th>
<th>Transition Function</th>
<th>Domain</th>
<th>Codomain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic automata</td>
<td>$f \circ \delta \circ i$</td>
<td>1 → 2</td>
<td>in Set</td>
</tr>
<tr>
<td>Non-deterministic automata</td>
<td>$f \circ \delta \circ i$</td>
<td>1 → 1</td>
<td>in Rel</td>
</tr>
<tr>
<td>Weighted automata</td>
<td>$f \circ \delta \circ i$</td>
<td>$K \rightarrow K$</td>
<td>in $\text{Vec}_K$</td>
</tr>
<tr>
<td>Subseq. transducers</td>
<td>$f \circ \delta \circ i$</td>
<td>1 → 1</td>
<td>in $\text{Kl}(\mathcal{T})$</td>
</tr>
</tbody>
</table>
For objects $I$ and $F$ in a category $C$, a $(C, I, F)$-automaton is a tuple $\mathcal{A} = \langle Q, i, f, (\delta_a)_{a \in A} \rangle$, where

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

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

$$I \xrightarrow{i} Q \xrightarrow{\delta_{a_1}} Q \xrightarrow{\delta_{a_2}} \ldots \xrightarrow{\delta_{a_n}} Q \xrightarrow{f} F$$
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

$$I \xrightarrow{i} Q \xrightarrow{\delta_{a_1}} Q \xrightarrow{\delta_{a_2}} \ldots \xrightarrow{\delta_{a_n}} Q \xrightarrow{f} F$$

Example

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

Subsequential transducers in a category
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
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 $Q \rightarrow B^* \times Q + 1$
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 $Q \rightarrow B^* \times Q + 1$
- 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 $\text{Kl}(\mathcal{T})$ with

- **objects**: sets $X, Y, Z, \ldots$
- **arrows**: $f: X \to Y$, where $f: X \to B^* \times Y + 1$ is a function
The output category for subgrsequential transducers

We consider partial actions for the free monoid $B^*$. We consider a category $\text{Kl}(\mathcal{T})$ with

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

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

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

We consider partial actions for the free monoid $B^*$. We consider a category $\text{Kl}(\mathcal{T})$ with

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

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

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

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

This is the Kleisli category for the monad $\mathcal{T} : \text{Set} \rightarrow \text{Set}$ 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

Interpreting the arrows

\[
\begin{align*}
delta_a \\
1 & \xrightarrow{i} Q & f & \xrightarrow{} 1 \quad \text{in } Kl(\mathcal{T})
\end{align*}
\]

ammounts to give
Interpreting the arrows

\[ 1 \xrightarrow{i} Q \xrightarrow{f} 1 \]

in \( \text{Kl}(\mathcal{T}) \)

amounts to give

- a function \( i: 1 \to B^* \times Q + 1 \), 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

\[ 1 \xrightarrow{i} Q \xrightarrow{f} 1 \quad \text{in } Kl(T) \]

amounts to give

- a function \( i: 1 \to B^* \times Q + 1 \), i.e. an initial state with an initial output in \( B^* \), or an undefined initial state
- for each \( a \in A \) a function \( \delta_a: Q \to B^* \times Q + 1 \)
The output category for subsequential transducers

Interpretting the arrows

\[ \begin{array}{ccc}
1 & \xrightarrow{i} & Q \\
\delta_a & \circ & \xrightarrow{f} & 1
\end{array} \]

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

ammounts to give

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

\[ 1 \xrightarrow{i} Q \xrightarrow{f} 1 \quad \text{in } \text{Kl}(\mathcal{T}) \]

amounts to give a **subsequential transducer**!
Interpretting the arrows

\[ 1 \xrightarrow{i} Q \xrightarrow{\delta_a} 1 \xrightarrow{f} 1 \]

in \( \text{Kl}(\mathcal{T}) \)

amounts 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
Minimization of \((C, I, F)\)-automata

- What does it mean for a \((C, I, F)\)-automaton to be minimal?
- What are sufficient conditions on \(C\) so that a minimal automaton for a language exists?
Minimization of \((C, I, F)\)-automata

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

A DFA is **minimal** when it **divides** any other automaton accepting the same language.
Minimization of \((C, I, F)\)-automata

• What does it mean for a \((C, I, F)\)-automaton to be minimal?
• What are sufficient conditions on \(C\) 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»
Minimization of \((C, I, F)\)-automata

• What does it mean for a \((C, I, F)\)-automaton to be minimal?
• What are sufficient conditions on \(C\) 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?
When does a ‘minimal’ automaton accepting a language $\mathcal{L}$ exist?

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

- an initial object $A_{\text{init}}(\mathcal{L})$, 

---

The three ingredients for minimization
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 $A_{\text{init}}(\mathcal{L})$,
- a final object $A_{\text{final}}(\mathcal{L})$, 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 $A_{\text{init}}(\mathcal{L})$,  
- a final object $A_{\text{final}}(\mathcal{L})$, and,  
- a factorization system

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

$$A_{\text{init}}(\mathcal{L}) \rightarrow \text{Min}(\mathcal{L}) \rightarrow A_{\text{final}}(\mathcal{L}).$$
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 \( A_{\text{init}}(\mathcal{L}) \),
- a final object \( A_{\text{final}}(\mathcal{L}) \), and,
- a factorization system

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

\[
A_{\text{init}}(\mathcal{L}) \rightarrow \text{Min}(\mathcal{L}) \rightarrow A_{\text{final}}(\mathcal{L}).
\]
deterministic automata, i.e. \((\text{Set}, 1, 2)\)-automata accepting a \((\text{Set}, 1, 2)\)-language
Trivial example

deterministic automata, i.e. \((\text{Set}, 1, 2)\)-automata accepting a \((\text{Set}, 1, 2)\)-language
Another trivial example

$\mathbb{R}$-weighted automata, i.e. $(\text{Vec}, \mathbb{R}, \mathbb{R})$-automata accepting a $(\text{Vec}, \mathbb{R}, \mathbb{R})$-language
Another trivial example

\( \mathbb{R} \)-weighted automata, i.e. \((\text{Vec}, \mathbb{R}, \mathbb{R})\)-automata accepting a \((\text{Vec}, \mathbb{R}, \mathbb{R})\)-language
The automaton $\text{Min}(\mathcal{L})$ divides any other automaton accepting $\mathcal{L}$.
The automaton $\text{Min}(L)$ divides any other automaton accepting $L$.

Thus far we identified simple sufficient conditions on $C$ so that minimization of $C$-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

\[ \begin{align*}
0 & \quad \xrightarrow{\varepsilon} \quad 1 \\
1 & \quad \xrightarrow{\varepsilon} \quad 0 \\
3 & \quad \xrightarrow{b|ab} \quad 2 \\
2 & \quad \xrightarrow{a|abb} \quad 3 \\
0 & \quad \xrightarrow{a|abba} \quad 1 \\
1 & \quad \xrightarrow{b|ba} \quad 0 \\
2 & \quad \xrightarrow{a|b} \quad 3 \\
3 & \quad \xrightarrow{b|b} \quad 2 \\
0 & \quad \xrightarrow{a} \quad 3 \\
2 & \quad \xrightarrow{b|b} \quad 1 \\
3 & \quad \xrightarrow{b|ab} \quad 0
\end{align*} \]
Minimization of subsequential transducers à la Choffrut
Minimization of subsequential transducers à la Choffrut

\[ \varepsilon \quad a|abba \quad a|abba \quad b|ba \quad a|b \quad b|b \quad a|b \quad b|b \quad b|b \quad b|b \]
Minimization of subsequential transducers à la Choffrut

![Diagram of subsequential transducers](image)
Minimization of subsequential transducers à la Choffrut
Recall that subsequential transducers are word automata interpreted in the category $\text{Kl}(\mathcal{T})$:

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

Does $\text{Kl}(\mathcal{T})$ 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 $\text{Kl}(\mathcal{T})$:

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

Does $\text{Kl}(\mathcal{T})$ 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 $\text{Kl} (\mathcal{T})$-automaton is $A^*$.
The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system

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

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

**Crucial fact:** $B^* \times \text{Irr}(A^*, B^*) \cong (A^* + \mathsf{one.osf})B^*$. 

Longest common prefixes play a fundamental role.
The ingredients for minimization

- initial automaton ✓
- final automaton ✓
- factorization system

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

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

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

- \(e: X \notRightarrow Y\) (i.e. \(e: X \rightarrow B^* \times Y + 1\)) is in \(\mathcal{E}\) iff each \(y \in Y\) is in the image of the second projection of \(e\).
- \(m: X \notRightarrow Y\) (i.e. \(m: X \rightarrow B^* \times Y + 1\)) is in \(\mathcal{M}\) iff \(m\) is everywhere defined, the second projection is injective and the first projection is constant \(\varepsilon\).
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 \(\text{Kl}(\mathcal{T})\):

• \(e: X \Rightarrow Y\) (i.e. \(e: X \rightarrow B^* \times Y + 1\)) is in \(\mathcal{E}\) iff each \(y \in Y\) is in the image of the second projection of \(e\).

• \(m: X \Rightarrow Y\) (i.e. \(m: X \rightarrow B^* \times Y + 1\)) is in \(\mathcal{M}\) iff \(m\) is everywhere defined, the second projection is injective and the first projection is constant \(\varepsilon\).

This also works if we replace \(B^*\) by a right cancellative monoid.
We obtain $\text{Min}(\mathcal{L})$ – the minimal subsequential transducer as obtained by Choffrut!
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?