

Comparison-free polyregular functions a.k.a. “polyblind functions”

Lê Thành Dũng (Tito) Nguyễn — n1td@nguyentito.eu — joint work with Cécilia Pradic

Updated version of a talk given at Trends in Transformations
(December 14th, 2021 — Online FSTTCS satellite workshop)

Reminder: deterministic two-way transducers (2DFT)

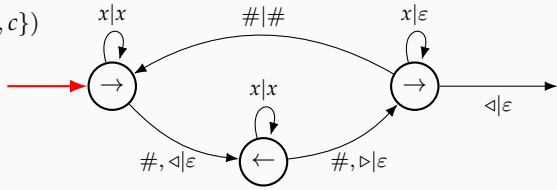
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

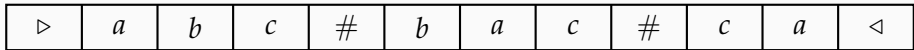
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:



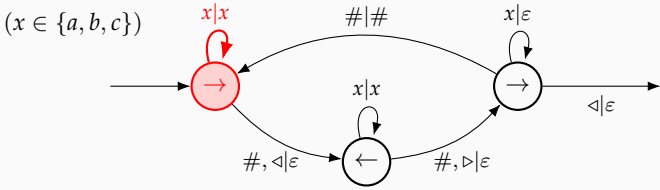
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

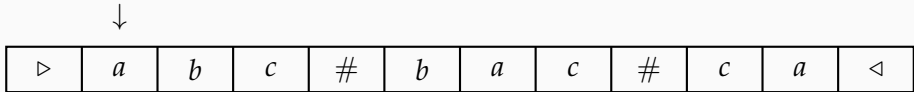
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:



Reminder: deterministic two-way transducers (2DFT)

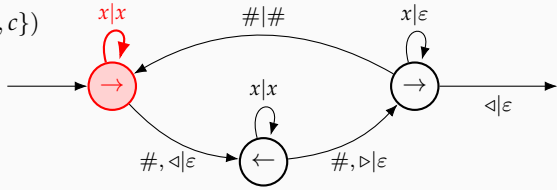
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

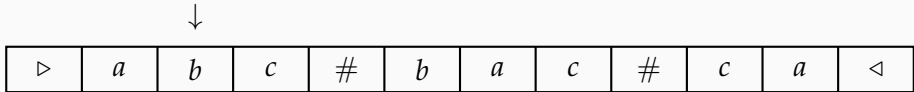
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
a



Reminder: deterministic two-way transducers (2DFT)

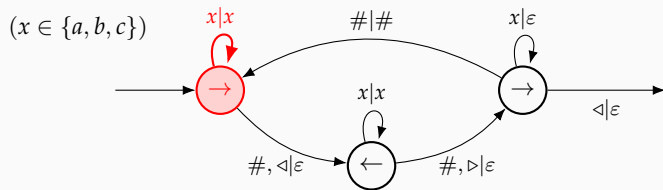
Two-way transducers: executive summary

(recall Sarah Winter's talk)

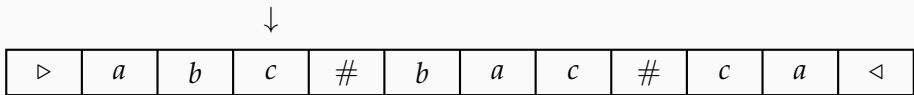
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
ab



Reminder: deterministic two-way transducers (2DFT)

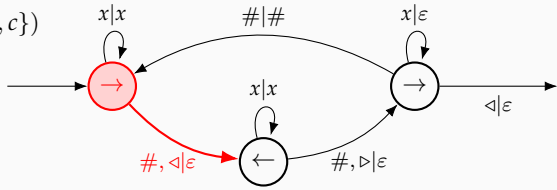
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

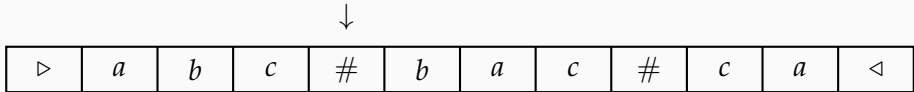
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abc



Reminder: deterministic two-way transducers (2DFT)

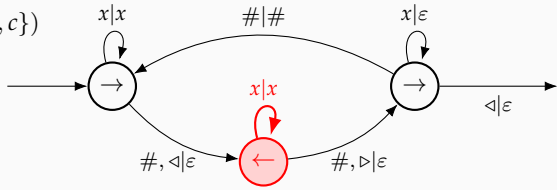
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

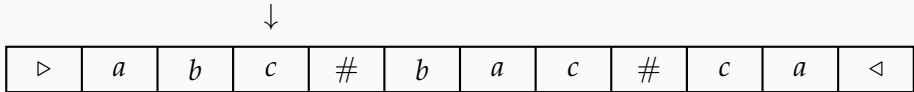
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abc



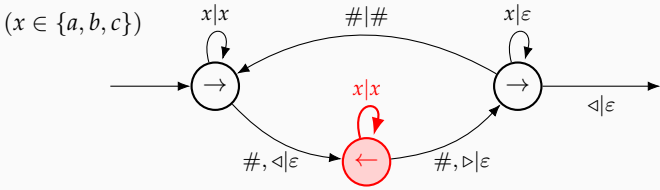
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

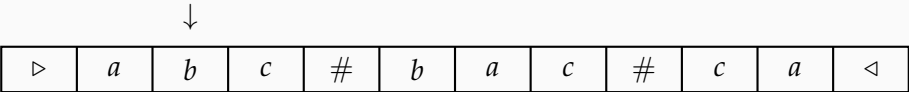
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abcc



Reminder: deterministic two-way transducers (2DFT)

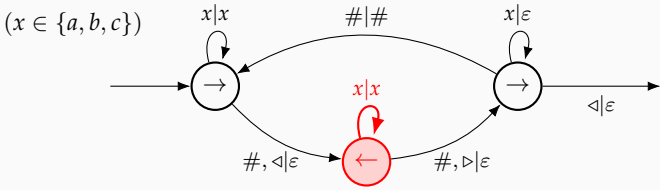
Two-way transducers: executive summary

(recall Sarah Winter's talk)

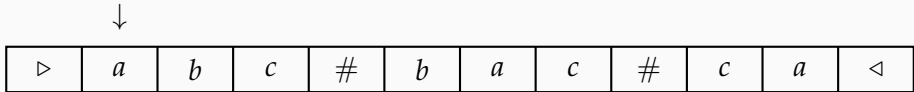
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccb



Reminder: deterministic two-way transducers (2DFT)

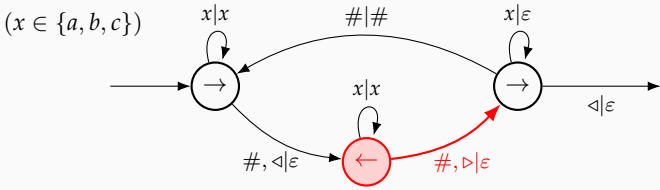
Two-way transducers: executive summary

(recall Sarah Winter's talk)

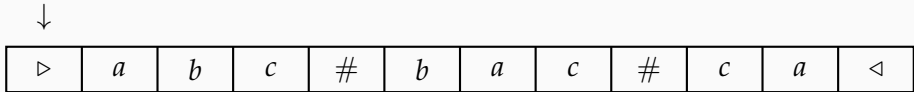
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba



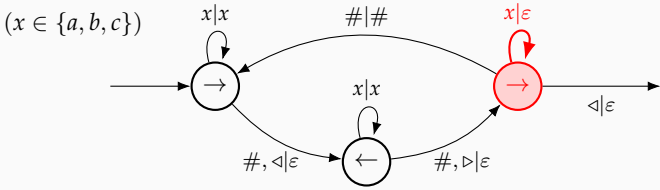
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

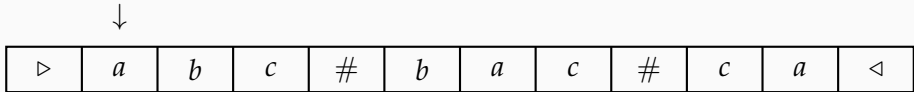
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba



Reminder: deterministic two-way transducers (2DFT)

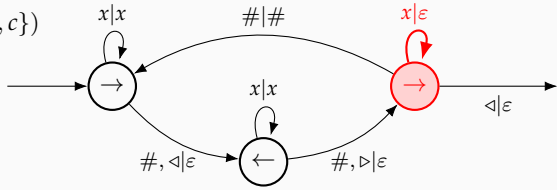
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

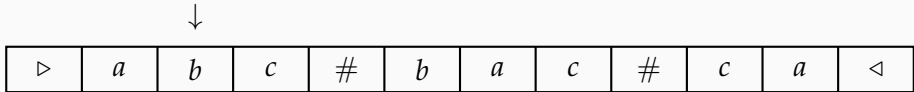
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba



Reminder: deterministic two-way transducers (2DFT)

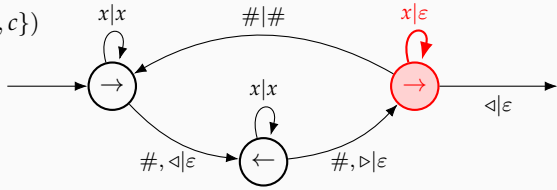
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

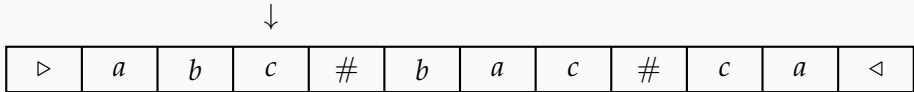
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba



Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

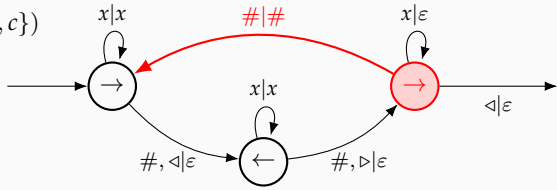
(recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

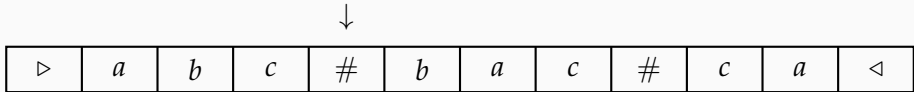
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba



Reminder: deterministic two-way transducers (2DFT)

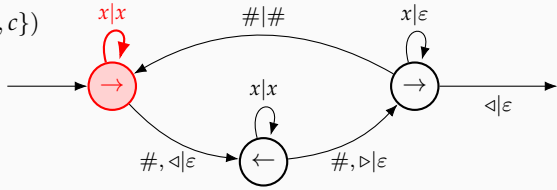
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

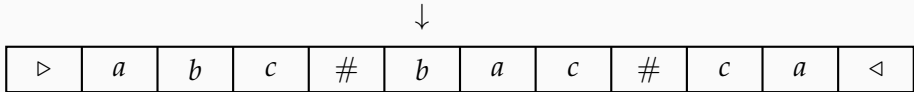
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#



Reminder: deterministic two-way transducers (2DFT)

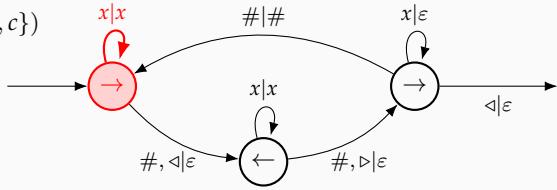
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

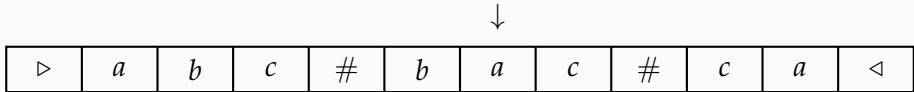
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#b



Reminder: deterministic two-way transducers (2DFT)

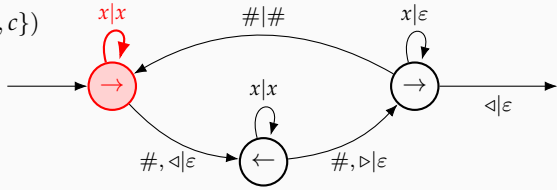
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

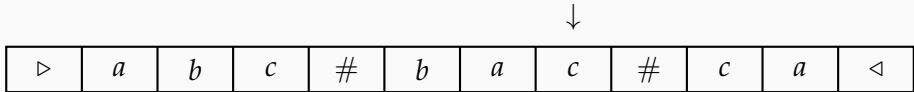
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#ba



Reminder: deterministic two-way transducers (2DFT)

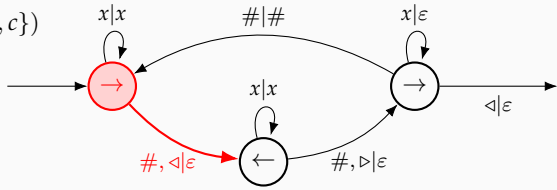
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

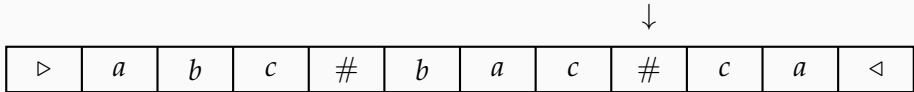
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bac



Reminder: deterministic two-way transducers (2DFT)

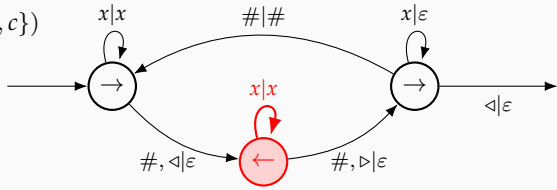
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

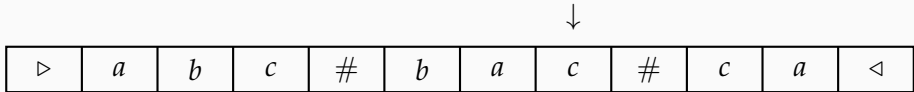
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bac



Reminder: deterministic two-way transducers (2DFT)

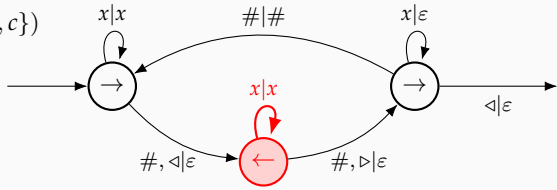
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

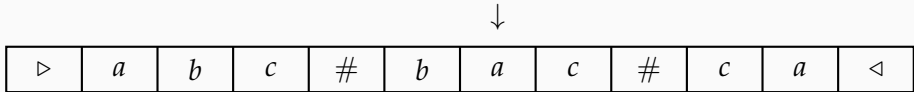
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bacc



Reminder: deterministic two-way transducers (2DFT)

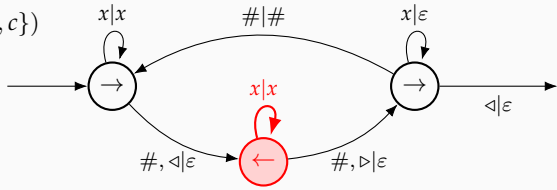
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

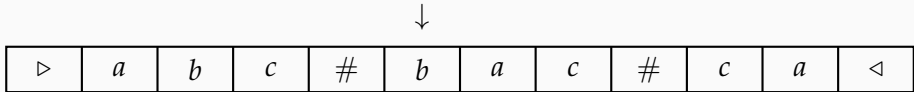
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bacca



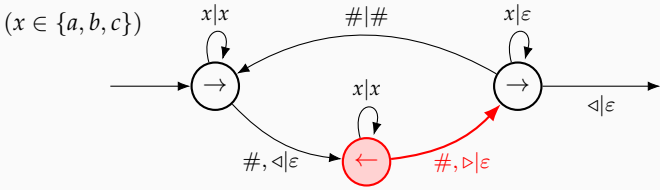
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

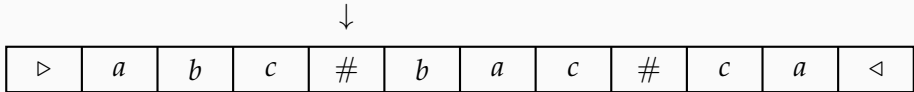
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#bccab



Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

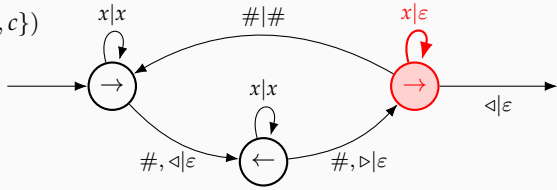
(recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

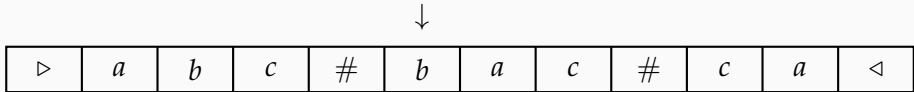
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bccab



Reminder: deterministic two-way transducers (2DFT)

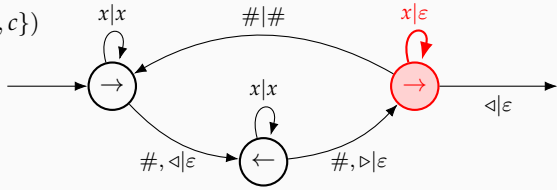
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

Example:

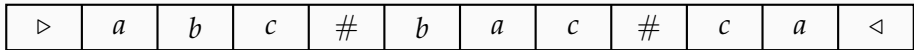
$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bccab

↓



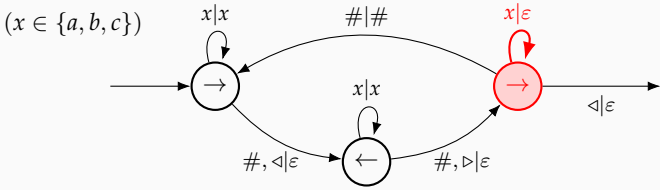
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

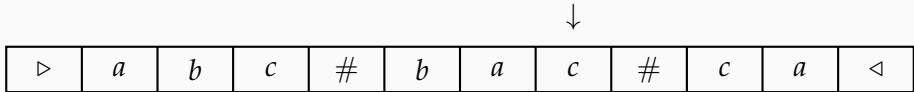
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#bccab



Reminder: deterministic two-way transducers (2DFT)

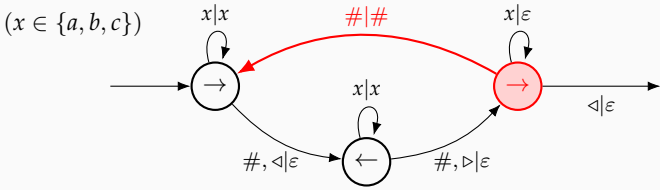
Two-way transducers: executive summary

(recall Sarah Winter's talk)

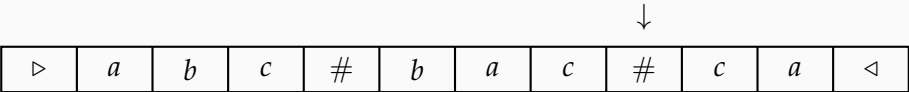
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#bccab



Reminder: deterministic two-way transducers (2DFT)

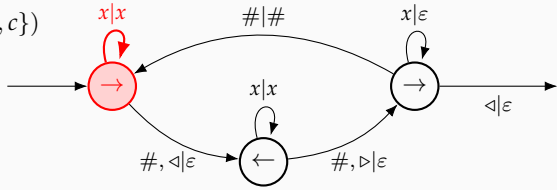
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

Example:

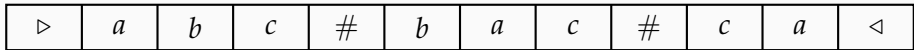
$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#baccab#

↓



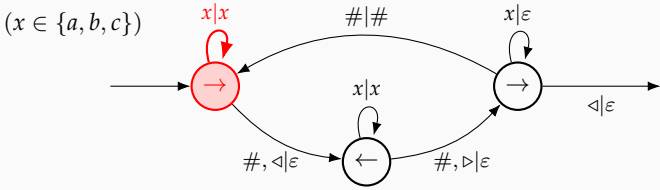
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

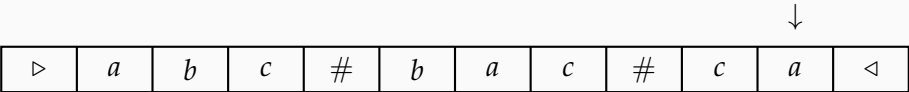
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#bccab#c



Reminder: deterministic two-way transducers (2DFT)

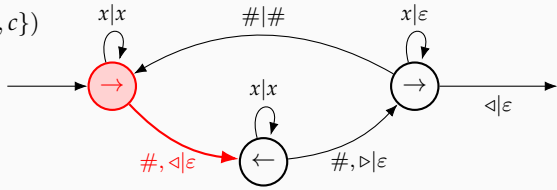
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

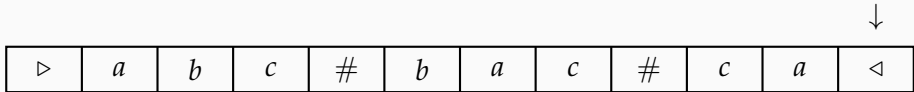
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#baccab#ca



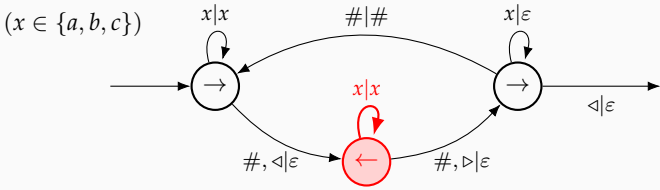
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

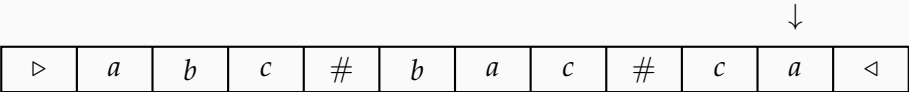
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#bccab#ca



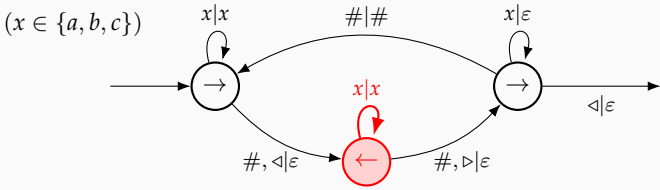
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

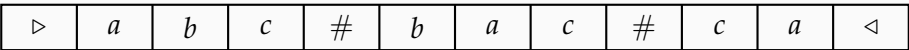
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#baccab#caa

↓



Reminder: deterministic two-way transducers (2DFT)

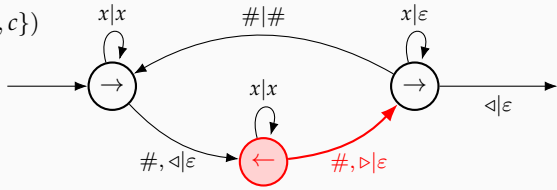
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

Example:

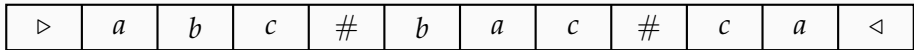
$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#baccab#caac

↓



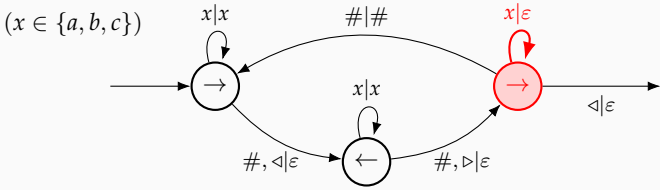
Reminder: deterministic two-way transducers (2DFT)

Two-way transducers: executive summary (recall Sarah Winter's talk)

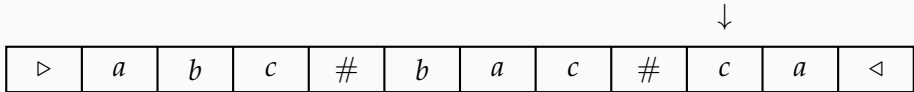
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$



Output:
abccba#baccab#caac



Reminder: deterministic two-way transducers (2DFT)

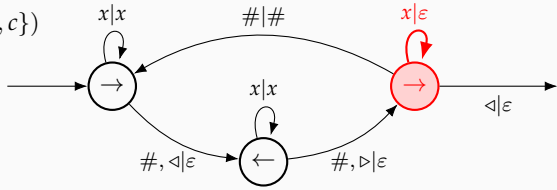
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

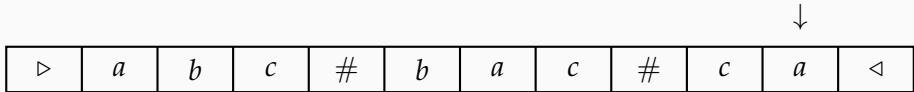
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#bccab#caac



Reminder: deterministic two-way transducers (2DFT)

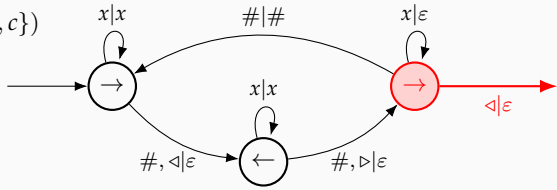
Two-way transducers: executive summary (recall Sarah Winter's talk)

Finite set of states + bidirectional reading head + output produced from left to right

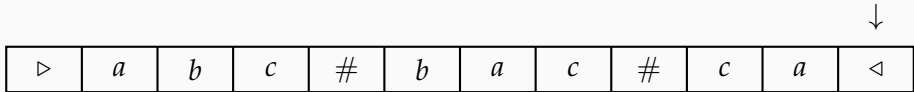
Example:

$$\begin{aligned} \text{mapPalin} : \{a, b, c, \#\}^* &\longrightarrow \{a, b, c, \#\}^* \\ w_1\# \dots \#w_n &\longmapsto w_1 \cdot \text{reverse}(w_1)\# \dots \#w_n \cdot \text{reverse}(w_n) \end{aligned}$$

$(x \in \{a, b, c\})$



Output:
abccba#baccab#caac



Regular functions

Functions $\Sigma^* \rightarrow \Gamma^*$ definable by 2DFTs are called **regular (string) functions**

(\supseteq rational functions, cf. Sarah's talk)

Properties of regular functions

- Linear growth: $|f(w)| = O(|w|)$
- Closed under composition (if $f: \Gamma^* \rightarrow \Sigma^*$ and $g: \Sigma^* \rightarrow \Pi^*$ are regular then so is $g \circ f$)
- L regular $\implies f^{-1}(L)$ regular

Functions $\Sigma^* \rightarrow \Gamma^*$ definable by 2DFTs are called **regular (string) functions**

(\supseteq rational functions, cf. Sarah's talk)

Properties of regular functions

- Linear growth: $|f(w)| = O(|w|)$
- Closed under composition (if $f: \Gamma^* \rightarrow \Sigma^*$ and $g: \Sigma^* \rightarrow \Pi^*$ are regular then so is $g \circ f$)
- L regular $\implies f^{-1}(L)$ regular

Alternative characterizations

- Via Monadic Second-Order logic (MSO transductions *à la* Courcelle)
- Copyless streaming string transducers (one-way machine model)
- Various functional programming or regexp-like (declarative) formalisms,
e.g. “regular list functions” [Bojańczyk, Daviaud & Krishna 2018]
- (recent work of ours) Minimal linear λ -calculus and Church encodings
→ see my PhD thesis *Implicit automata in linear logic and categorical transducer theory*
<https://nguyentito.eu/thesis.pdf>

Polyregular functions

- A larger class of string-to-string transductions (regular \subsetneq polyregular)
- Garnered significant attention recently, starting with [Bojańczyk 2018]

Properties of polyregular functions

- *Polynomial* growth: $|f(w)| = O(|w|^k)$
- L regular $\implies f^{-1}(L)$ regular; closed under composition

Polyregular functions

- A larger class of string-to-string transductions (regular \subsetneq polyregular)
- Garnered significant attention recently, starting with [Bojańczyk 2018]

Properties of polyregular functions

- *Polynomial* growth: $|f(w)| = O(|w|^k)$
- L regular $\implies f^{-1}(L)$ regular; closed under composition

Characterizations [Bojańczyk 2018; Bojańczyk, Kiefer & Lhote 2019]

- Multidimensional MSO interpretations
- Imperative nested loop programs
- “Polynomial list functions” (extending regular list functions):
 simply typed λ -calculus + primitive list type w/ some list manipulation primitives
- Composition closure of [regular functions \cup “squaring with underlining”]

Polyregular functions

- A larger class of string-to-string transductions (regular \subsetneq polyregular)
- Garnered significant attention recently, starting with [Bojańczyk 2018]

Properties of polyregular functions

- *Polynomial growth*: $|f(w)| = O(|w|^k)$
- L regular $\implies f^{-1}(L)$ regular; closed under composition

Characterizations [Bojańczyk 2018; Bojańczyk, Kiefer & Lhote 2019]

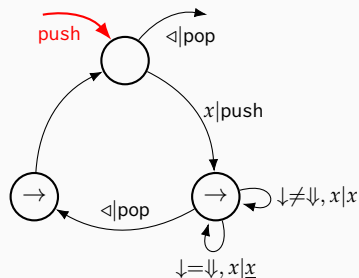
- Multidimensional MSO interpretations
- Imperative nested loop programs
- “Polynomial list functions” (extending regular list functions):
 simply typed λ -calculus + primitive list type w/ some list manipulation primitives
- Composition closure of [regular functions \cup “squaring with underlining”]
- k -pebble string-to-string transducers [Milo, Suciu & Vianu 2000]

k -pebble transducers: executive summary

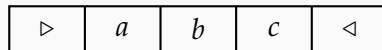
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$



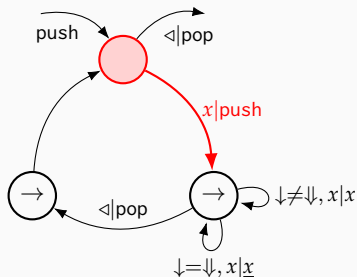
output =

k -pebble transducers: executive summary

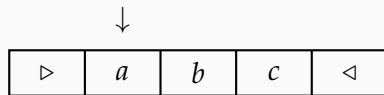
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$



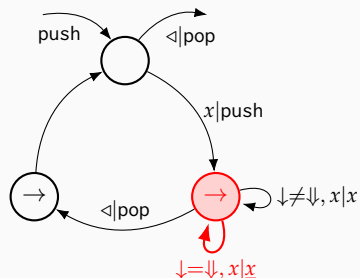
output =

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

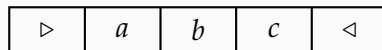
- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a}a\underline{b}a\underline{a}b$

\Downarrow
 \downarrow



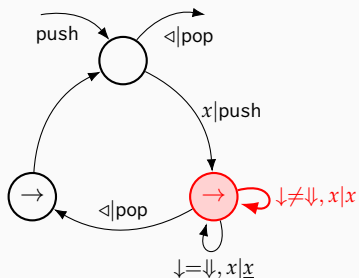
output =

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)

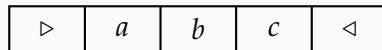


squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$

$aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$

\Downarrow

\downarrow



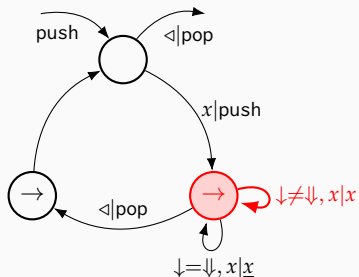
output = \underline{a}

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

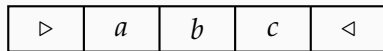
Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$

\Downarrow

\downarrow



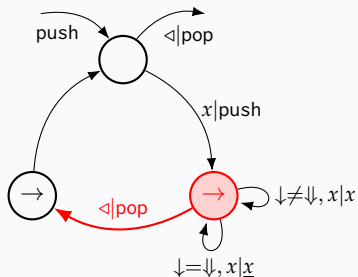
output = \underline{ab}

k -pebble transducers: executive summary

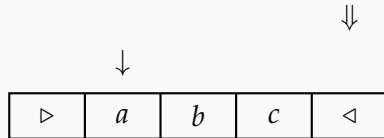
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$



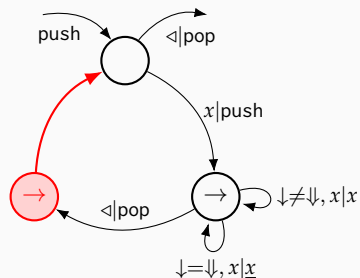
output = \underline{abc}

k -pebble transducers: executive summary

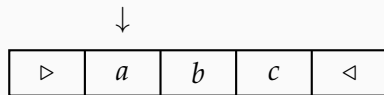
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$



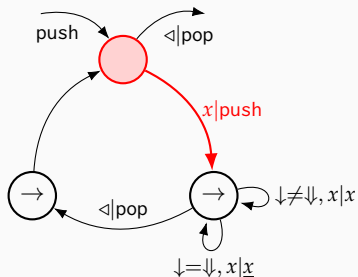
output = abc

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

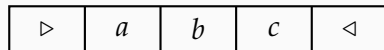
- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$

↓



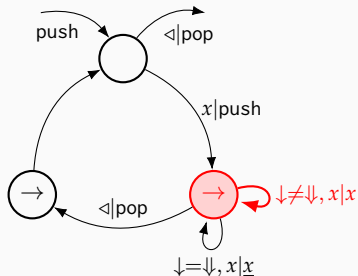
output = abc

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

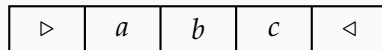
Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b}$

\Downarrow

\downarrow



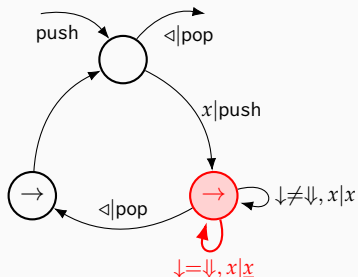
output = \underline{abc}

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)

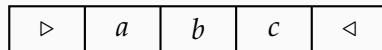


squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$

$aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b}$

↓

↓



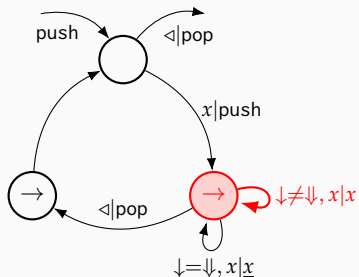
output = abca

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

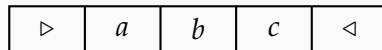
Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$

\Downarrow

\downarrow



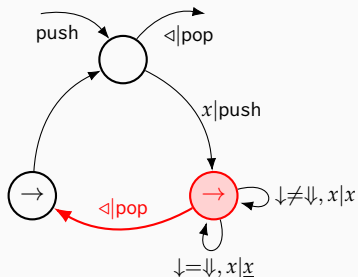
output = $\underline{a} \underline{b} \underline{c} \underline{a} \underline{b}$

k -pebble transducers: executive summary

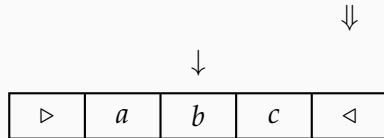
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$



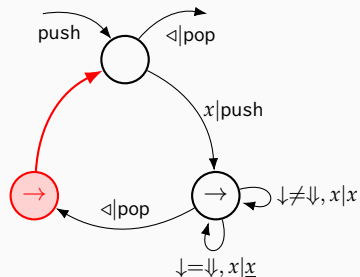
output = abcab

k -pebble transducers: executive summary

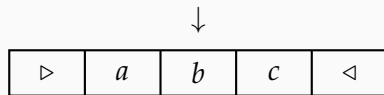
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$



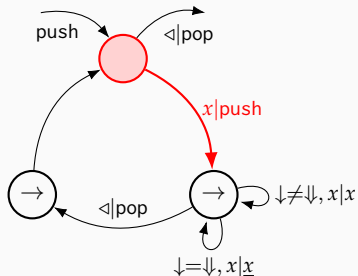
output = $\underline{a} \underline{b} c \underline{a} \underline{b} c$

k -pebble transducers: executive summary

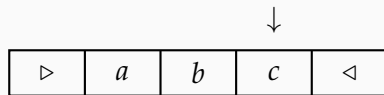
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$



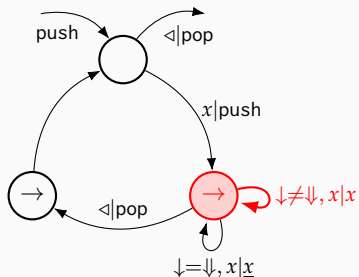
output = abcabc

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

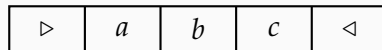
Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$

\Downarrow

\downarrow



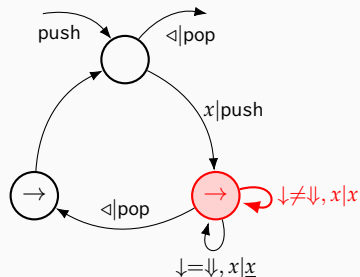
output = $\underline{a} \underline{b} c \underline{a} \underline{b}$

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)

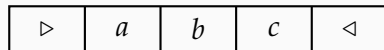


squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$

$aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b}$

\Downarrow

\downarrow



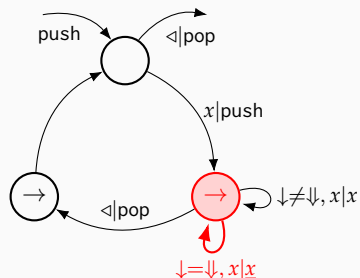
output = $\underline{a} \underline{b} \underline{c} \underline{a} \underline{b}$

k -pebble transducers: executive summary

Finite set of states + a stack of two-way reading heads of height $\leq k$

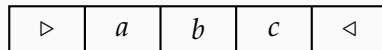
- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b}$

\Downarrow
 \downarrow



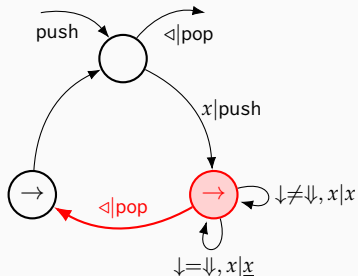
output = $\underline{a} \underline{b} \underline{c} \underline{a} \underline{b} \underline{c}$

k -pebble transducers: executive summary

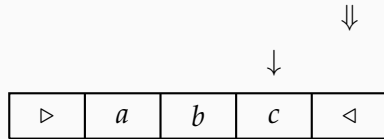
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} b \underline{a} \underline{a} b$



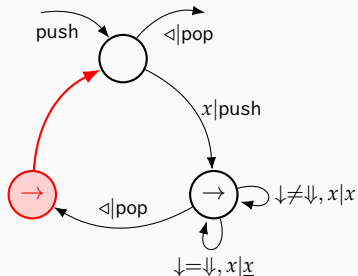
output = $\underline{a} \underline{b} \underline{c} \underline{a} \underline{b} \underline{c}$

k -pebble transducers: executive summary

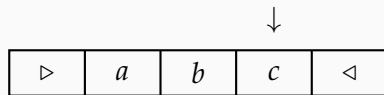
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \underline{a} \underline{b} \end{aligned}$$



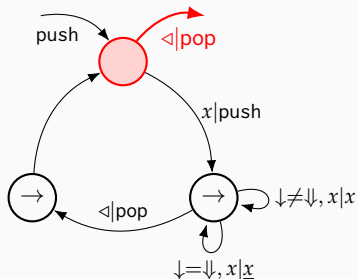
output = abcabcabc

k -pebble transducers: executive summary

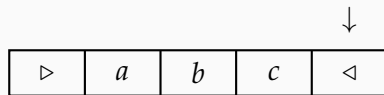
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



$$\begin{aligned} \text{squaring : } \Sigma^* &\rightarrow (\Sigma \cup \underline{\Sigma})^* \\ aab &\mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \end{aligned}$$



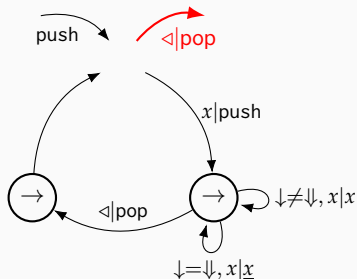
output = abcabc

k -pebble transducers: executive summary

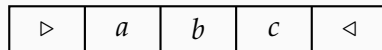
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack \rightarrow **what if we remove them?**
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$)



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b}$



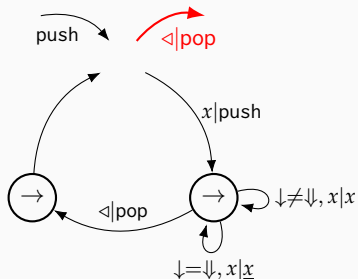
output = abcabc

k -pebble transducers: executive summary

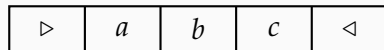
Finite set of states + a stack of two-way reading heads of height $\leq k$

- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack \rightarrow **what if we remove them?**
- 1-pebble transducers \cong 2DFTs

Example: “squaring with underlining” ($k = 2$) **seems to require comparisons**



squaring : $\Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$
 $aab \mapsto \underline{a} \underline{a} \underline{b} \underline{a} \underline{a} \underline{b} \underline{a} \underline{b}$



output = abcabc

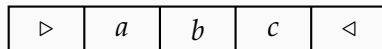
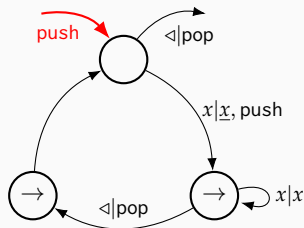
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output =

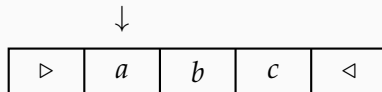
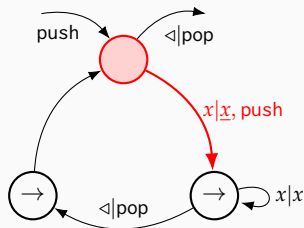
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output =

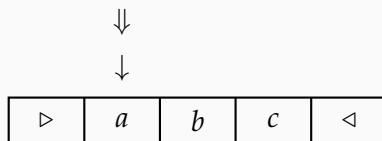
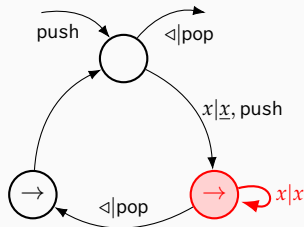
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = a

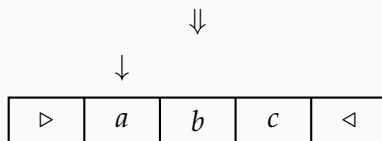
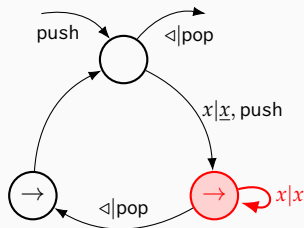
Blind transducers (a.k.a. "comparison-free pebble transducers")

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: "comparison-free squaring"

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aa

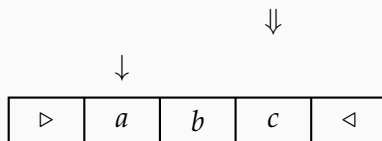
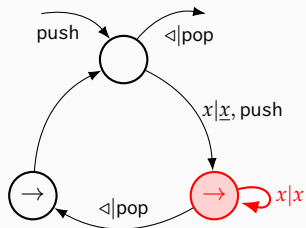
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aab

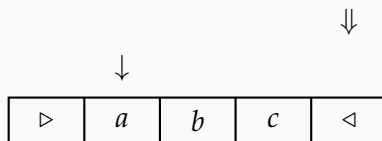
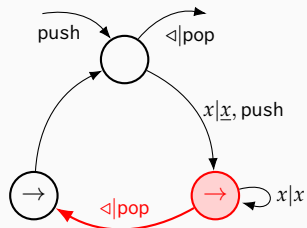
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabc

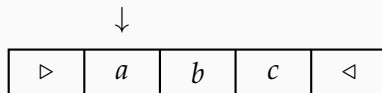
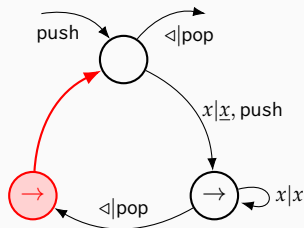
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabc

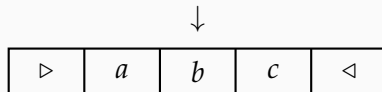
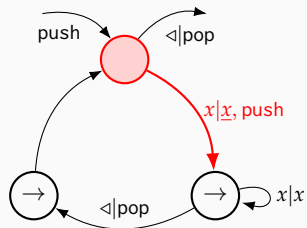
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabc

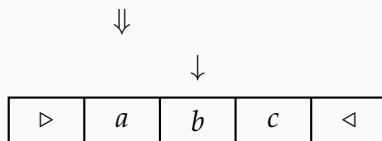
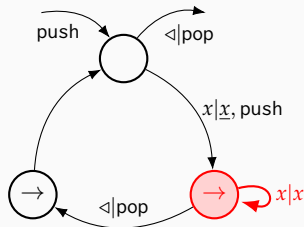
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcb

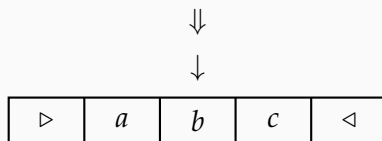
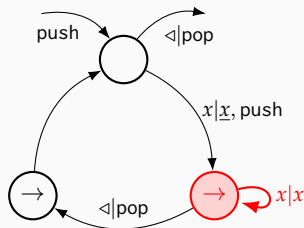
Blind transducers (a.k.a. "comparison-free pebble transducers")

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: "comparison-free squaring"

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcba

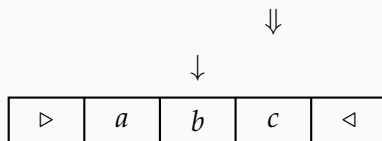
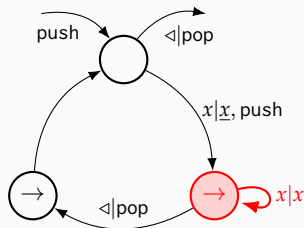
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbab

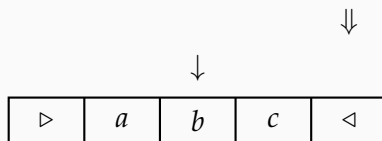
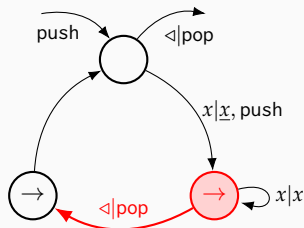
Blind transducers (a.k.a. "comparison-free pebble transducers")

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: "comparison-free squaring"

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbabc

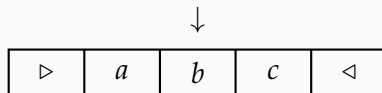
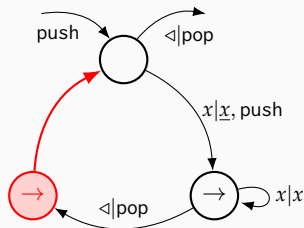
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbabc

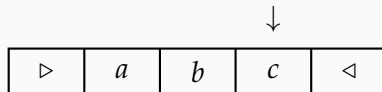
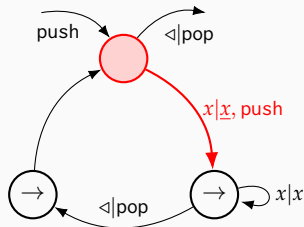
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbabc

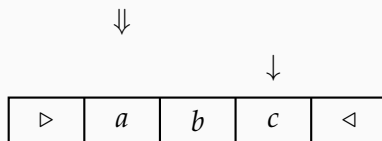
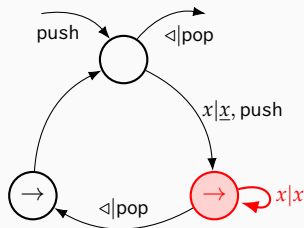
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbabcc

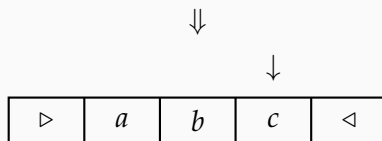
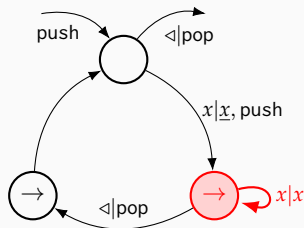
Blind transducers (a.k.a. "comparison-free pebble transducers")

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: "comparison-free squaring"

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



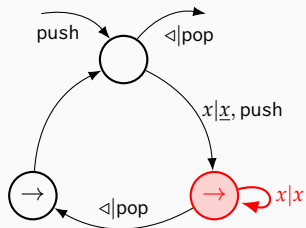
output = aabcbabcca

Blind transducers (a.k.a. “comparison-free pebble transducers”)

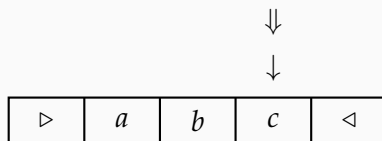
Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”



$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = $\underline{a}abc\underline{b}abcc\underline{a}b$

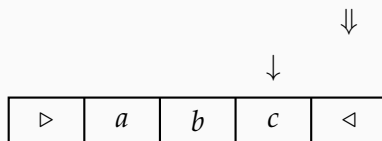
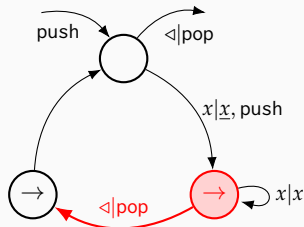
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



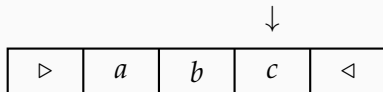
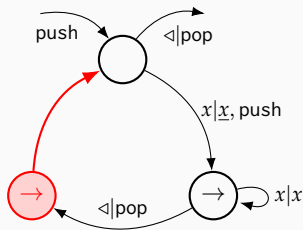
Blind transducers (a.k.a. "comparison-free pebble transducers")

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: "comparison-free squaring"

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = aabcbabccabc

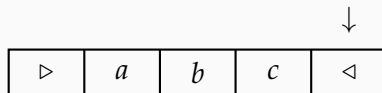
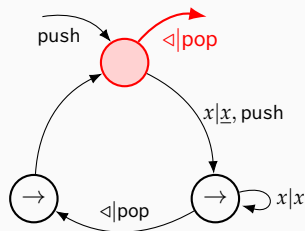
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$



output = $\underline{a}abc\underline{b}abcc\underline{a}bc$

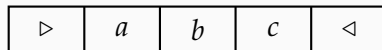
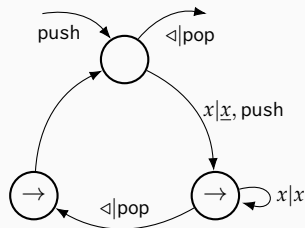
Blind transducers (a.k.a. “comparison-free pebble transducers”)

Polyblind functions = computed by blind transducers

Pebble transducers where the comparisons between reading heads are disallowed

Example: “comparison-free squaring”

$\text{cfsquaring}(abb) = \underline{a}bb\underline{b}abb\underline{b}abb$



output = abcbabccabc

Contributions covered in this talk

- Alternative characterizations → “canonicity” / “naturality” claim
(our original motivation: λ -calculus stuff, actually)
- Separation results (along the way: technical results such as a *pebble minimization* theorem)
- Discussion of special cases: unary inputs, integer sequences, aperiodicity

Alternative characterization (1/2): composition by substitutions

Definition (Composition by substitutions)

Let Γ, Σ and I be finite alphabets and $f: \Gamma^* \rightarrow I^*$, $g_i: \Gamma^* \rightarrow \Sigma^*$ and $w \in \Gamma^*$.

Define $\text{CbS}(f, (g_i)_{i \in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

$$\text{CbS}(f, (g_i)_{i \in I})(w) = g_{i_1}(w) \dots g_{i_k}(w)$$

E.g. for cfsquaring, we take $f: \Sigma^* \rightarrow (\Sigma \cup \{X\})^*$, $g_X, g_a: \Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$ (for $a \in \Sigma$) so that

$$f(abc) = aXbXcX \quad g_a(w) = \underline{a} \quad \text{and} \quad g_X(w) = w$$

- Note: both polyblind and polyregular functions are closed under CbS

Alternative characterization (1/2): composition by substitutions

Definition (Composition by substitutions)

Let Γ, Σ and I be finite alphabets and $f: \Gamma^* \rightarrow I^*$, $g_i: \Gamma^* \rightarrow \Sigma^*$ and $w \in \Gamma^*$.

Define $\text{CbS}(f, (g_i)_{i \in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

$$\text{CbS}(f, (g_i)_{i \in I})(w) = g_{i_1}(w) \dots g_{i_k}(w)$$

E.g. for cfsquaring, we take $f: \Sigma^* \rightarrow (\Sigma \cup \{X\})^*$, $g_X, g_a: \Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$ (for $a \in \Sigma$) so that

$$f(abc) = aXbXcX \quad g_a(w) = \underline{a} \quad \text{and} \quad g_X(w) = w$$

- Note: both polyblind and polyregular functions are closed under CbS

Alternative definition of polyblind functions: smallest class such that...

- Every regular function is polyblind
- If f is regular and g_i is polyblind for every $i \in I$ then $\text{CbS}(f, (g_i)_{i \in I})$ is polyblind

Alternative characterization (1/2): composition by substitutions

Definition (Composition by substitutions)

Let Γ, Σ and I be finite alphabets and $f: \Gamma^* \rightarrow I^*$, $g_i: \Gamma^* \rightarrow \Sigma^*$ and $w \in \Gamma^*$.

Define $\text{CbS}(f, (g_i)_{i \in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

$$\text{CbS}(f, (g_i)_{i \in I})(w) = g_{i_1}(w) \dots g_{i_k}(w)$$

E.g. for cfsquaring, we take $f: \Sigma^* \rightarrow (\Sigma \cup \{X\})^*$, $g_x, g_a: \Sigma^* \rightarrow (\Sigma \cup \underline{\Sigma})^*$ (for $a \in \Sigma$) so that

$$f(abc) = aXbXcX \quad g_a(w) = \underline{a} \quad \text{and} \quad g_X(w) = w$$

- Note: both polyblind and polyregular functions are closed under CbS

Alternative definition of polyblind functions: smallest class such that...

- Every regular function is polyblind
- If f is regular and g_i is polyblind for every $i \in I$ then $\text{CbS}(f, (g_i)_{i \in I})$ is polyblind
- Straightforward reformulation of blind transducers; more convenient to manipulate formally
- Number of pebbles = nesting of the CbS operator + 1

Alternative characterization (2/2): composition and basic combinators

Polyblind functions can be characterized using a linear λ -calculus (again, see my PhD thesis)

→ Hints at the following non-trivial theorem

(reproven with automata-theoretic tools in our ICALP'21 paper with no references to the λ -calculus)

Closure under composition

If $f: \Sigma^* \rightarrow \Gamma^*$ and $g: \Gamma^* \rightarrow \Delta^*$ are both polyblind, so is $g \circ f$.

Alternative characterization (2/2): composition and basic combinators

Polyblind functions can be characterized using a linear λ -calculus (again, see my PhD thesis)

→ Hints at the following non-trivial theorem

(reproven with automata-theoretic tools in our ICALP'21 paper with no references to the λ -calculus)

Closure under composition

If $f: \Sigma^* \rightarrow \Gamma^*$ and $g: \Gamma^* \rightarrow \Delta^*$ are both polyblind, so is $g \circ f$.

Leads to a combinator-based definition:

Alternative definition of polyblind functions

Least class containing the regular functions, `cfsquaring` and closed under composition.

- Analogous to the case of general polyregular functions
`cfsquaring` replaced by “squaring with underlining” in the above → all polyregular functions

Alternative characterization (2/2): composition and basic combinators

Polyblind functions can be characterized using a linear λ -calculus (again, see my PhD thesis)

→ Hints at the following non-trivial theorem

(reproven with automata-theoretic tools in our ICALP'21 paper with no references to the λ -calculus)

Closure under composition

If $f: \Sigma^* \rightarrow \Gamma^*$ and $g: \Gamma^* \rightarrow \Delta^*$ are both polyblind, so is $g \circ f$.

Leads to a combinator-based definition:

Alternative definition of polyblind functions

Least class containing the regular functions, `cfsquaring` and closed under composition.

- Analogous to the case of general polyregular functions
 `cfsquaring` replaced by “squaring with underlining” in the above → all polyregular functions
- Note: regular functions can also themselves be obtained by composing:
 - sequential functions (that can themselves be decomposed by [Krohn & Rhodes 1965]);
 - $\text{mapReverse}_\Sigma, \text{mapDuplicate}_\Sigma : (\Sigma \cup \{\#\})^* \rightarrow (\Sigma \cup \{\#\})^*$ for $\# \notin \Sigma$(a theorem of Bojańczyk et al., see e.g. [Bojańczyk & Stefański 2020])

Separation results (or: what about counterexamples?)

Theorem

The function $f : a^n \in \{a\}^* \mapsto a\#aa\#\dots\#a^n$
is polyregular but not polyblind.

Corollary

“Squaring with underlining” is not polyblind.

Theorem

$g : a^{n_1}\#\dots\#a^{n_k} \in \{a,\#\}^* \mapsto a^{n_1 \times n_1}\#\dots\#a^{n_k \times n_k}$
is polyregular but not polyblind.

Note: stronger result in [Douéneau-Tabot 2021]
(cf. next talk) on $a^{n_1}\#\dots\#a^{n_k} \mapsto a^{n_1 \times n_1 + \dots + n_k \times n_k}$

Separation results (or: what about counterexamples?)

Theorem

The function $f : a^n \in \{a\}^* \mapsto a\#aa\#\dots\#a^n$ is polyregular but not polyblind.

Corollary

“Squaring with underlining” is not polyblind.

f is also an **HDT0L transduction**

(\iff computable by a marble transducer /
copyful streaming string transducer;
more in Gaëtan Douéneau-Tabot’s talk next)

\longrightarrow HDT0L $\not\subset$ polyblind; conversely:

Theorem

$(w \in \Gamma^* \mapsto w^{|w|}) \in \text{polyblind} \setminus \text{HDT0L}$ for $|\Gamma| \geq 2$.

Note: polynomially growing HDT0L \subset polyreg

Theorem

$g : a^{n_1}\# \dots \# a^{n_k} \in \{a, \#\}^* \mapsto a^{n_1 \times n_1}\# \dots \# a^{n_k \times n_k}$ is polyregular but not polyblind.

Note: stronger result in [Douéneau-Tabot 2021]
(cf. next talk) on $a^{n_1}\# \dots \# a^{n_k} \mapsto a^{n_1 \times n_1 + \dots + n_k \times n_k}$

Separation results (or: what about counterexamples?)

Theorem

The function $f : a^n \in \{a\}^* \mapsto a\#aa\#\dots\#a^n$ is polyregular but not polyblind.

Corollary

“Squaring with underlining” is not polyblind.

f is also an **HDT0L transduction**

(\iff computable by a marble transducer /
copyful streaming string transducer;
more in Gaëtan Douéneau-Tabot’s talk next)

\rightarrow HDT0L $\not\subset$ polyblind; conversely:

Theorem

$(w \in \Gamma^* \mapsto w^{|w|}) \in \text{polyblind} \setminus \text{HDT0L}$ for $|\Gamma| \geq 2$.

Note: polynomially growing HDT0L \subset polyreg

Theorem

$g : a^{n_1}\#\dots\#a^{n_k} \in \{a, \#\}^* \mapsto a^{n_1 \times n_1}\#\dots\#a^{n_k \times n_k}$ is polyregular but not polyblind.

Note: stronger result in [Douéneau-Tabot 2021]
(cf. next talk) on $a^{n_1}\#\dots\#a^{n_k} \mapsto a^{n_1 \times n_1 + \dots + n_k \times n_k}$

Definition

For $h : \Gamma^* \rightarrow \Sigma^*$, $w_1, \dots, w_n \in \Gamma^*$ with $\# \notin \Gamma$,
 $\mathbf{map}(h)(w_1\#\dots\#w_n) = f(w_1)\#\dots\#f(w_n)$.

Since $g = \mathbf{map}(w \mapsto w^{|w|})$:

Corollary

Polyblind functions are not closed under **map**, unlike regular and polyregular functions.

\rightarrow obstruction to characterizing polyblind fn
by list-processing functional programs
à la regular/polynomial list functions

Theorem

$$g(a^{n_1} \# \dots \# a^{n_k}) = a^{n_1 \times n_1} \# \dots \# a^{n_k \times n_k}$$

is not polyblind.

Proof by contradiction: assume g is polyblind.

First, $|g(w)| = O(|w|^2)$ therefore g is computed by some 2-head blind transducer.

Theorem (Blind pebble minimization)

If f is polyblind and $|f(w)| = O(|w|^k)$ then some blind transducer with k heads computes f .

More on this kind of result in Gaëtan’s talk...

Spoiler: elitist conferences such as LICS are terrible, don’t trust their peer review

Theorem

$g(a^{n_1} \# \dots \# a^{n_k}) = a^{n_1 \times n_1} \# \dots \# a^{n_k \times n_k}$

is not polyblind.

Proof by contradiction: assume g is polyblind.

First, $|g(w)| = O(|w|^2)$ therefore g is computed by some 2-head blind transducer.

Theorem (Blind pebble minimization)

If f is polyblind and $|f(w)| = O(|w|^k)$ then some blind transducer with k heads computes f .

More on this kind of result in Gaëtan’s talk...

Spoiler: elitist conferences such as LICS are terrible, don’t trust their peer review

Theorem

$g(a^{n_1} \# \dots \# a^{n_k}) = a^{n_1 \times n_1} \# \dots \# a^{n_k \times n_k}$

is not polyblind.

Proof by contradiction: assume g is polyblind.

First, $|g(w)| = O(|w|^2)$ therefore g is computed by some 2-head blind transducer. Equivalently, for some finite I and regular functions f and h_i ,

$$g = \text{CbS}(f, (h_i)_{i \in I}) \quad (\text{composition by substitution})$$

$$\text{i.e. } f(w) = i_1 \dots i_m \implies g(w) = h_{i_1}(w) \dots h_{i_m}(w)$$

Separation proof for “map unary square” + blind pebble minimization

Theorem (Blind pebble minimization)

If f is polyblind and $|f(w)| = O(|w|^k)$ then some blind transducer with k heads computes f .

More on this kind of result in Gaëtan’s talk...

Spoiler: elitist conferences such as LICS are terrible, don’t trust their peer review

Theorem

$g(a^{n_1} \# \dots \# a^{n_k}) = a^{n_1 \times n_1} \# \dots \# a^{n_k \times n_k}$

is not polyblind.

Proof by contradiction: assume g is polyblind.

First, $|g(w)| = O(|w|^2)$ therefore g is computed by some 2-head blind transducer. Equivalently, for some finite I and regular functions f and h_i ,

$$g = \text{CbS}(f, (h_i)_{i \in I}) \quad (\text{composition by substitution})$$

$$\text{i.e. } f(w) = i_1 \dots i_m \implies g(w) = h_{i_1}(w) \dots h_{i_m}(w)$$

(Oversimplified) idea: consider $g(w)$ where

$$w = \underbrace{aa \dots a \# \dots \# aa \dots a}_{|I| \text{ times } \# \text{ i.e. } |I| + 1 \text{ blocks of } a}$$

Separation proof for “map unary square” + blind pebble minimization

Theorem (Blind pebble minimization)

If f is polyblind and $|f(w)| = O(|w|^k)$ then some blind transducer with k heads computes f .

More on this kind of result in Gaëtan’s talk...

Spoiler: elitist conferences such as LICS are terrible, don’t trust their peer review

Theorem

$g(a^{n_1} \# \dots \# a^{n_k}) = a^{n_1 \times n_1} \# \dots \# a^{n_k \times n_k}$

is not polyblind.

Proof by contradiction: assume g is polyblind.

First, $|g(w)| = O(|w|^2)$ therefore g is computed by some 2-head blind transducer. Equivalently, for some finite I and regular functions f and h_i ,

$$g = \text{CbS}(f, (h_i)_{i \in I}) \quad (\text{composition by substitution})$$

$$\text{i.e. } f(w) = i_1 \dots i_m \implies g(w) = h_{i_1}(w) \dots h_{i_m}(w)$$

(Oversimplified) idea: consider $g(w)$ where

$$w = \underbrace{aa \dots a \# \dots \# aa \dots a}_{|I| \text{ times } \# \text{ i.e. } |I| + 1 \text{ blocks of } aa}$$

$|I|$ times $\#$ i.e. $|I| + 1$ blocks of aa

“pump” p -th factor $aa \dots a$ ($1 \leq p \leq |I| + 1$) in w

→ p -th factor in $g(w)$ grows as $\Theta(n^2)$

→ \exists corresponding $j[p] \in I$ such that

$|f(w)|_{j[p]}$ and $|h_{j[p]}(w)|$ grow as $\Theta(n)$

Pigeonhole principle: $j[p] = j[q]$ for some $p \neq q$

→ “pumping” p -th factor of w makes q -th factor of $g(w)$ grow, contradiction

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

In fact, $\exists \mathcal{S} \subseteq \mathbb{Q}[X]$ finite such that $\{P(n) \mid P \in \mathcal{S}\}$ contains $\{\text{lengths of maximal } a\text{-factors of } g(a^n)\}$, by structural induction on *poly-pumping sequences*.

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

In fact, $\exists \mathcal{S} \subseteq \mathbb{Q}[X]$ finite such that $\{P(n) \mid P \in \mathcal{S}\}$ contains $\{\text{lengths of maximal } a\text{-factors of } g(a^n)\}$, by structural induction on poly-pumping sequences.

Definition (poly-pumping sequence of words)

Smallest subclass of $(\Sigma^*)^{\mathbb{N}}$

- Containing the constant sequences $\alpha_n = w$
- Closed under concatenation $\alpha_n = \beta_n \cdot \gamma_n$
- Closed under "iteration" $\alpha_n = (\beta_n)^n$

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

In fact, $\exists \mathcal{S} \subseteq \mathbb{Q}[X]$ finite such that $\{P(n) \mid P \in \mathcal{S}\}$ contains $\{\text{lengths of maximal } a\text{-factors of } g(a^n)\}$, by structural induction on poly-pumping sequences.

Definition (poly-pumping sequence of words)

Smallest subclass of $(\Sigma^*)^{\mathbb{N}}$

- Containing the constant sequences $\alpha_n = w$
- Closed under concatenation $\alpha_n = \beta_n \cdot \gamma_n$
- Closed under “iteration” $\alpha_n = (\beta_n)^n$

Theorem (polyblind with unary input)

$f : \{a\}^* \rightarrow \Sigma^*$ is polyblind

if and only if $\exists p \in \mathbb{N}$ such that $(f(a^{(n+1)p+m}))_{n \in \mathbb{N}}$ is poly-pumping for every $m < p$.

→ “ultimately periodic combinations” (u.p.c.)

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

In fact, $\exists \mathcal{S} \subseteq \mathbb{Q}[X]$ finite such that $\{P(n) \mid P \in \mathcal{S}\}$ contains $\{\text{lengths of maximal } a\text{-factors of } g(a^n)\}$, by structural induction on poly-pumping sequences.

- Regular word sequences are u.p.c. of pumping sequences $(u_0(v_1)^n \dots (v_l)^n u_l)_{n \in \mathbb{N}}$ [Choffrut 2017]
Proof idea: find an idempotent in a suitable transition monoid of your favorite machine model for reg fn
- Proof for general polyblind sequences: induction on the CbS-based definition

Definition (poly-pumping sequence of words)

Smallest subclass of $(\Sigma^*)^{\mathbb{N}}$

- Containing the constant sequences $\alpha_n = w$
- Closed under concatenation $\alpha_n = \beta_n \cdot \gamma_n$
- Closed under "iteration" $\alpha_n = (\beta_n)^n$

Theorem (polyblind with unary input)

$f : \{a\}^* \rightarrow \Sigma^*$ is polyblind

if and only if $\exists p \in \mathbb{N}$ such that $(f(a^{(n+1)p+m}))_{n \in \mathbb{N}}$ is poly-pumping for every $m < p$.

→ "ultimately periodic combinations" (u.p.c.)

Separation proof idea continued: unary inputs

Theorem

$f(a^n) = a\#aa\#\dots\#a^n$ is not polyblind.

Observation: $f(a^n)$ has the n maximal a -factors

$a \quad aa \quad \dots \quad a^n$

Lemma

For any polyblind $g : \{a\}^* \rightarrow \Sigma^*$, there are $O(1)$ possible lengths for maximal a -factors in $g(a^n)$.

In fact, $\exists \mathcal{S} \subseteq \mathbb{Q}[X]$ finite such that $\{P(n) \mid P \in \mathcal{S}\}$ contains $\{\text{lengths of maximal } a\text{-factors of } g(a^n)\}$, by structural induction on *poly-pumping sequences*.

- Regular word sequences are u.p.c. of pumping sequences $(u_0(v_1)^n \dots (v_l)^n u_l)_{n \in \mathbb{N}}$ [Choffrut 2017]
Proof idea: find an idempotent in a suitable transition monoid of your favorite machine model for reg fn
- Proof for general polyblind sequences: induction on the CbS-based definition
- Work-in-progress: similar characterization of polyregular word sequences

Definition (poly-pumping sequence of words)

Smallest subclass of $(\Sigma^*)^{\mathbb{N}}$

- Containing the constant sequences $\alpha_n = w$
- Closed under concatenation $\alpha_n = \beta_n \cdot \gamma_n$
- Closed under “iteration” $\alpha_n = (\beta_n)^n$

Theorem (polyblind with unary input)

$f : \{a\}^* \rightarrow \Sigma^*$ is polyblind

if and only if $\exists p \in \mathbb{N}$ such that $(f(a^{(n+1)p+m}))_{n \in \mathbb{N}}$ is poly-pumping for every $m < p$.

→ “ultimately periodic combinations” (u.p.c.)

Poly-pumping sequences: generated from constants by concat and “iteration” $\alpha_n = (\beta_n)^n$.

Theorem

“ultimately periodic combinations” of poly-pumping sequences = all polyblind sequences

What about the *aperiodic* (\Leftrightarrow *first-order*) case?

Poly-pumping sequences: generated from constants by concat and “iteration” $\alpha_n = (\beta_n)^n$.

Theorem

“ultimately periodic combinations” of poly-pumping sequences = all polyblind sequences

What about the *aperiodic* (\Leftrightarrow *first-order*) case?

FO-regular functions: many equivalent def.

- logic: replace MSO by FO
- 2DFT/SST w/ aperiodic monoid of behaviors
- FO-regular list functions
- regexp-like [Dartois, Gastin & Krishna 2021]
- more self-advertizing:
 - 2DFT with *planar* behaviors (upcoming)
 - non-commutative λ -calculus

Poly-pumping sequences: generated from constants by concat and “iteration” $\alpha_n = (\beta_n)^n$.

Theorem

“ultimately periodic combinations” of poly-pumping sequences = all polyblind sequences

What about the *aperiodic* (\Leftrightarrow first-order) case?

FO-regular functions: many equivalent def.

- logic: replace MSO by FO
- 2DFT/SST w/ aperiodic monoid of behaviors
- FO-regular list functions
- regexp-like [Dartois, Gastin & Krishna 2021]
- more self-advertizing:
 - 2DFT with *planar* behaviors (upcoming)
 - non-commutative λ -calculus

Definition (First-order polyblind functions)

FO-polyblind = smallest class such that

- every FO-regular function is FO-polyblind
- if f is FO-regular and g_i is FO-polyblind $\forall i \in I$ then $\text{CbS}(f, (g_i)_{i \in I})$ is FO-polyblind

Poly-pumping sequences: generated from constants by concat and “iteration” $\alpha_n = (\beta_n)^n$.

Theorem

“ultimately periodic combinations” of poly-pumping sequences = all polyblind sequences

What about the *aperiodic* (\Leftrightarrow *first-order*) case?

Also, FO-polyblind \subsetneq FO-polyregular (the previous counterexamples are all FO-polyreg); and composition, pebble minimization, etc. should work *mutatis mutandis*

FO-regular functions: many equivalent def.

- logic: replace MSO by FO
- 2DFT/SST w/ aperiodic monoid of behaviors
- FO-regular list functions
- regexp-like [Dartois, Gastin & Krishna 2021]
- more self-advertizing:
 - 2DFT with *planar* behaviors (upcoming)
 - non-commutative λ -calculus – should extend to FO-polyblind characterization

Definition (First-order polyblind functions)

FO-polyblind = smallest class such that

- every FO-regular function is FO-polyblind
- if f is FO-regular and g_i is FO-polyblind $\forall i \in I$ then $\text{CbS}(f, (g_i)_{i \in I})$ is FO-polyblind

Poly-pumping sequences: generated from constants by concat and “iteration” $\alpha_n = (\beta_n)^n$.

Theorem

“ultimately periodic combinations” of poly-pumping sequences = all polyblind sequences

What about the *aperiodic* (\Leftrightarrow first-order) case?

Theorem

$f : \{a\}^* \rightarrow \Sigma^*$ is FO-polyblind if and only if $\exists p \in \mathbb{N}$ such that $n \mapsto f(a^{n+p})$ is poly-pumping.

Also, FO-polyblind \subsetneq FO-polyregular (the previous counterexamples are all FO-polyreg); and composition, pebble minimization, etc. should work *mutatis mutandis*

FO-regular functions: many equivalent def.

- logic: replace MSO by FO
- 2DFT/SST w/ aperiodic monoid of behaviors
- FO-regular list functions
- regexp-like [Dartois, Gastin & Krishna 2021]
- more self-advertizing:
 - 2DFT with *planar* behaviors (upcoming)
 - non-commutative λ -calculus – should extend to FO-polyblind characterization

Definition (First-order polyblind functions)

FO-polyblind = smallest class such that

- every FO-regular function is FO-polyblind
- if f is FO-regular and g_i is FO-polyblind $\forall i \in I$ then $\text{CbS}(f, (g_i)_{i \in I})$ is FO-polyblind

Integer sequences (unpublished)

Previously: unary *inputs*. Gaëtan's talk (next):
unary *outputs*. Let's consider both: let $f: \mathbb{N} \rightarrow \mathbb{N}$.

Integer sequences (unpublished)

Previously: unary *inputs*. Gaëtan's talk (next):
unary *outputs*. Let's consider both: let $f: \mathbb{N} \rightarrow \mathbb{N}$.

Polyregular integer sequences are eqv. to:

1. HDT0L (\Leftrightarrow linear recurrence) s.t. $f(n) = n^{O(1)}$
2. ultimately periodic combinations of $\mathbb{N}[X]$
(unary poly-pumping sequences = polynomials in $\mathbb{N}[X]$)
3. polyblind integer sequences

\rightarrow single canonical automata-theoretic class of
polynomial growth integer sequences?
polyregular \Leftrightarrow (1) \Leftrightarrow (2) holds for arbitrary input
and unary output [Douéneau–Tobot 2021];
the rest is elementary

Integer sequences (unpublished)

Previously: unary *inputs*. Gaëtan's talk (next):
unary *outputs*. Let's consider both: let $f: \mathbb{N} \rightarrow \mathbb{N}$.

Polyregular integer sequences are eqv. to:

1. HDT0L (\Leftrightarrow linear recurrence) s.t. $f(n) = n^{O(1)}$
2. ultimately periodic combinations of $\mathbb{N}[X]$
(unary poly-pumping sequences = polynomials in $\mathbb{N}[X]$)
3. polyblind integer sequences

\rightarrow single canonical automata-theoretic class of
polynomial growth integer sequences?
polyregular \Leftrightarrow (1) \Leftrightarrow (2) holds for arbitrary input
and unary output [Douéneau–Tobot 2021];
the rest is elementary

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{N}$. Then f is FO-polyblind \Leftrightarrow
 $\exists P \in \mathbb{Z}[X] : \forall n$ except finitely many, $f(n) = P(n)$

Thus, the following is not FO-polyblind for $k \geq 2$:

$$b_k(n) = \binom{n}{k} \quad \text{e.g. } b_2(n) = \frac{n(n-1)}{2}$$

Integer sequences (unpublished)

Previously: unary *inputs*. Gaëtan's talk (next):
unary *outputs*. Let's consider both: let $f: \mathbb{N} \rightarrow \mathbb{N}$.

Polyregular integer sequences are eqv. to:

1. HDT0L (\Leftrightarrow linear recurrence) s.t. $f(n) = n^{O(1)}$
2. ultimately periodic combinations of $\mathbb{N}[X]$
(unary poly-pumping sequences = polynomials in $\mathbb{N}[X]$)
3. polyblind integer sequences

\rightarrow single canonical automata-theoretic class of
polynomial growth integer sequences?
polyregular \Leftrightarrow (1) \Leftrightarrow (2) holds for arbitrary input
and unary output [Douéneau–Tobot 2021];
the rest is elementary

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{N}$. Then f is FO-polyblind \Leftrightarrow
 $\exists P \in \mathbb{Z}[X] : \forall n$ except finitely many, $f(n) = P(n)$

Thus, the following is not FO-polyblind for $k \geq 2$:

$$b_k(n) = \binom{n}{k} \quad \text{e.g. } b_2(n) = \frac{n(n-1)}{2}$$

Yet all b_k are both polyblind (divide by $k!$)
and FO-polyreg (think of $a^n \mapsto a\#aa\#\dots\#a^n$)
 \rightarrow **FO-polyblind** \neq (**FO-polyreg** \cap **polyblind**)

Integer sequences (unpublished)

Previously: unary *inputs*. Gaëtan's talk (next): unary *outputs*. Let's consider both: let $f: \mathbb{N} \rightarrow \mathbb{N}$.

Polyregular integer sequences are eqv. to:

1. HDT0L (\Leftrightarrow linear recurrence) s.t. $f(n) = n^{O(1)}$
2. ultimately periodic combinations of $\mathbb{N}[X]$
(unary poly-pumping sequences = polynomials in $\mathbb{N}[X]$)
3. polyblind integer sequences

\rightarrow single canonical automata-theoretic class of polynomial growth integer sequences?
polyregular \Leftrightarrow (1) \Leftrightarrow (2) holds for arbitrary input and unary output [Douéneau–Tobot 2021];
the rest is elementary

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{N}$. Then f is FO-polyregular $\Leftrightarrow \exists P \in \mathbb{Q}[X] : \forall n$ except finitely many, $f(n) = P(n)$

Theorem

Let $f: \mathbb{N} \rightarrow \mathbb{N}$. Then f is FO-polyblind $\Leftrightarrow \exists P \in \mathbb{Z}[X] : \forall n$ except finitely many, $f(n) = P(n)$

Thus, the following is not FO-polyblind for $k \geq 2$:

$$b_k(n) = \binom{n}{k} \quad \text{e.g. } b_2(n) = \frac{n(n-1)}{2}$$

Yet all b_k are both polyblind (divide by $k!$) and FO-polyreg (think of $a^n \mapsto a\#aa\#\dots\#a^n$)
 \rightarrow **FO-polyblind** \neq (**FO-polyreg** \cap **polyblind**)

$$\forall P \in \mathbb{Q}[X], P(X+a) = \sum_{i=0}^{\deg P} \Delta^i(P)(a) \binom{X}{i}$$

\rightarrow all polynomials P s.t. $P(\mathbb{N}) \subseteq \mathbb{N}$ are FO-polyreg

Not a logical characterization of FO-polyblind functions

Let $\mathfrak{M} : \{\text{words}\} \rightarrow \{\text{finite models}\}$ be as usual.

For $\mathfrak{U} = (U, R, \dots)$, let $\mathfrak{U}^k = (U^k, R_1, \dots, R_k, \dots)$ where $R_i(x_1, \dots, x_m) :\Leftrightarrow R(\pi_i(x_1), \dots, \pi_i(x_m))$.

Reasonable-sounding conjecture

f is FO-polyblind if and only there exist $k \in \mathbb{N}$ and a FO transduction φ such that

$$\forall w. \mathfrak{M}(f(w)) \simeq \varphi\left(\mathfrak{M}(w)^k\right)$$

Idea: in $\mathfrak{M}(w)^k$, one cannot compare different coordinates \rightarrow “comparison-free” condition

Not a logical characterization of FO-polyblind functions

Let $\mathfrak{M} : \{\text{words}\} \rightarrow \{\text{finite models}\}$ be as usual.

For $\mathfrak{U} = (U, R, \dots)$, let $\mathfrak{U}^k = (U^k, R_1, \dots, R_k, \dots)$ where $R_i(x_1, \dots, x_m) :\Leftrightarrow R(\pi_i(x_1), \dots, \pi_i(x_m))$.

Reasonable-sounding conjecture

f is FO-polyblind if and only there exist $k \in \mathbb{N}$ and a FO transduction φ such that

$$\forall w. \mathfrak{M}(f(w)) \simeq \varphi\left(\mathfrak{M}(w)^k\right)$$

Idea: in $\mathfrak{M}(w)^k$, one cannot compare different coordinates \rightarrow “comparison-free” condition

- Defines a subclass of FO-polyregular functions, according to [Bojańczyk, Kiefer & Lhote 2019]
- Beware: naive replacement of FO with MSO \rightarrow non reg-preserving

Not a logical characterization of FO-polyblind functions

Let $\mathfrak{M} : \{\text{words}\} \rightarrow \{\text{finite models}\}$ be as usual.

For $\mathfrak{U} = (U, R, \dots)$, let $\mathfrak{U}^k = (U^k, R_1, \dots, R_k, \dots)$ where $R_i(x_1, \dots, x_m) :\Leftrightarrow R(\pi_i(x_1), \dots, \pi_i(x_m))$.

Reasonable-sounding conjecture

f is FO-polyblind if and only there exist $k \in \mathbb{N}$ and a FO transduction φ such that

$$\forall w. \mathfrak{M}(f(w)) \simeq \varphi\left(\mathfrak{M}(w)^k\right)$$

Idea: in $\mathfrak{M}(w)^k$, one cannot compare different coordinates \rightarrow “comparison-free” condition

- Defines a subclass of FO-polyregular functions, according to [Bojańczyk, Kiefer & Lhote 2019]
- Beware: naive replacement of FO with MSO \rightarrow non reg-preserving
- FO-polyblind \subsetneq this kind of FO-interpretation; separating example:

$$w_1 \# \dots \# w_n \mapsto (w_1)^n \dots (w_n)^n$$

- it requires a k -pebble transducer for $k \geq 3$; apply the blind pebble minimization theorem
- j.w.w. Mikołaj Bojańczyk, Gaëtan Douéneau-Tabot, Sandra Kiefer et Cécilia Pradic

Conclusion

A new(?) class of string-to-string functions: *polyblind* (or *comparison-free polyregular*) functions.

Equivalent definitions

- by blind transducers
- inductively (composition by substitution)
- **as the composition closure of regular functions** + $\text{cfsquaring}(abc) = \underline{a}abc\underline{b}abcc\underline{c}abc$
- linear λ -calculus (not covered here)

Conclusion

A new(?) class of string-to-string functions: *polyblind* (or *comparison-free polyregular*) functions.

Equivalent definitions

- by blind transducers
 - inductively (composition by substitution)
 - **as the composition closure of regular functions** + $\text{cfsquaring}(abc) = \underline{a}abc\underline{b}abcc\underline{c}abc$
 - linear λ -calculus (not covered here)
-
- L regular language $\implies f^{-1}(L)$ also regular
 - polynomial growth: $|f(w)| = O(|w|^k)$
 - **pebble minimization theorem:**
 $k = \text{number of heads necessary to compute } f$
 - strictly included in polyregular functions
 - $a^n \mapsto a\#aa\#\dots\#a^n$ / "map unary square"
 - for $\{a\}^* \rightarrow \{a\}^*$ polyblind = polyregular
 - well-behaved first-order counterpart

Conclusion

A new(?) class of string-to-string functions: *polyblind* (or *comparison-free polyregular*) functions.

Equivalent definitions

- by blind transducers
 - inductively (composition by substitution)
 - **as the composition closure of regular functions** + $\text{cfsquaring}(abc) = \underline{a}abc\underline{b}abcc\underline{c}abc$
 - linear λ -calculus (not covered here)
-
- L regular language $\implies f^{-1}(L)$ also regular
 - polynomial growth: $|f(w)| = O(|w|^k)$
 - **pebble minimization theorem:**
 $k = \text{number of heads necessary to compute } f$
 - strictly included in polyregular functions
 - $a^n \mapsto a\#aa\#\dots\#a^n$ / "map unary square"
 - for $\{a\}^* \rightarrow \{a\}^*$ polyblind = polyregular
 - well-behaved first-order counterpart

Future work: generalization to *trees*?

(The λ -calculus suggests an obvious extension)

Conclusion

A new(?) class of string-to-string functions: *polyblind* (or *comparison-free polyregular*) functions.

Equivalent definitions

- by blind transducers
 - inductively (composition by substitution)
 - **as the composition closure of regular functions** + $\text{cfsquaring}(abc) = \underline{a}abc\underline{b}abcc\underline{a}bc$
 - linear λ -calculus (not covered here)
-
- L regular language $\implies f^{-1}(L)$ also regular
 - polynomial growth: $|f(w)| = O(|w|^k)$
 - **pebble minimization theorem:**
 $k = \text{number of heads necessary to compute } f$
 - strictly included in polyregular functions
 - $a^n \mapsto a\#aa\#\dots\#a^n$ / "map unary square"
 - for $\{a\}^* \rightarrow \{a\}^*$ polyblind = polyregular
 - well-behaved first-order counterpart

Future work: generalization to *trees*?

(The λ -calculus suggests an obvious extension)

Decision problems are not well understood

- Membership: next talk!
- Equivalence: seems hard
(obstruction to naive approach: Schmude, *On polynomial grammars extended with substitution*)

A new(?) class of string-to-string functions: *polyblind* (or *comparison-free polyregular*) functions.

Equivalent definitions

- by blind transducers
 - inductively (composition by substitution)
 - **as the composition closure of regular functions** + $\text{cfsquaring}(abc) = \underline{a}abc\underline{b}abcc\underline{a}bc$
 - linear λ -calculus (not covered here)
-
- L regular language $\implies f^{-1}(L)$ also regular
 - polynomial growth: $|f(w)| = O(|w|^k)$
 - **pebble minimization theorem:**
 $k = \text{number of heads necessary to compute } f$
 - strictly included in polyregular functions
 - $a^n \mapsto a\#aa\#\dots\#a^n$ / “map unary square”
 - for $\{a\}^* \rightarrow \{a\}^*$ polyblind = polyregular
 - well-behaved first-order counterpart

Future work: generalization to *trees*?

(The λ -calculus suggests an obvious extension)

Decision problems are not well understood

- Membership: next talk!
- Equivalence: seems hard
 (obstruction to naive approach: Schmude, *On polynomial grammars extended with substitution*)

Not as robust as hoped at first:

polyblind functions are *not* equivalent to

- “comparison-free” logical interpretations
 - a restriction of polynomial list functions
- are there other “interesting” subclasses of polyregular functions?