Polyblind functions

Cécilia Pradic
j.w.w. Lê Thành Dũng (Tito) Nguyễn
15/05/25 Swansea theory seminar

Meta

Talk based on a 2021 ICALP paper, but

Actual goals

- Introduce you to **some** very restricted models of computations $\{0,1\}^* \to \{0,1\}^*$.
- Motivate those **somewhat** by theoretical results.

  Some practical applications: think verification of extended regexps/sed programs
- Give you an idea of what hard problems people look at in this space.

If you have questions, please interrupt and ask!

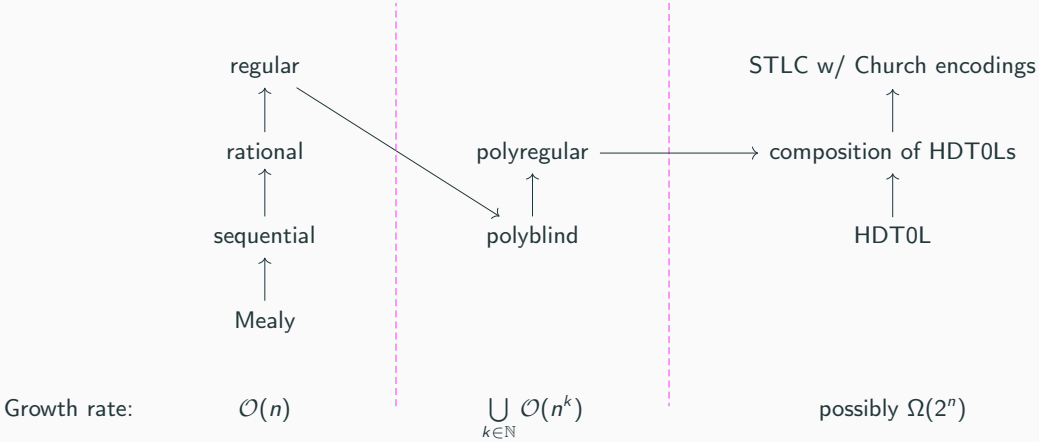
(I don't care if I don't cover pet result $\#n$)

Introduction: string-to-string transducers

Automata-theoretic classes of string-to-string functions

In contrast with languages (string to booleans), many sensible notions:

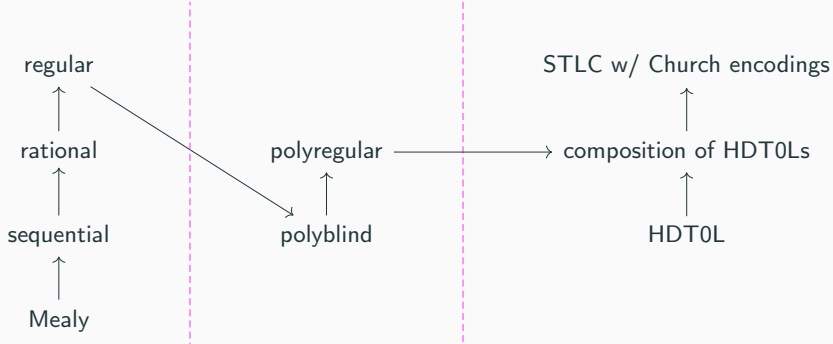
- Arrows are strict inclusions, no path = incomparable
- I can make this picture worse if you desire



Automata-theoretic classes of string-to-string functions

In contrast with languages (string to booleans), many sensible notions:

- Arrows are strict inclusions, no path = incomparable
- I can make this picture worse if you desire



Today: regular, polyregular and polyblind

Deterministic two-way transducers (2DFT)

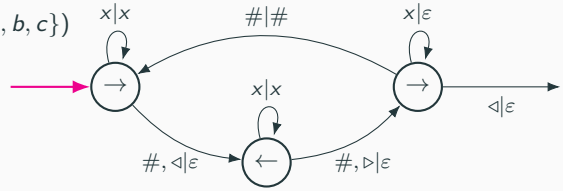
Two-way transducers: executive summary

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

Example:

$$\begin{array}{lccl} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$

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



Output:

▷	a	b	c	#	b	a	c	#	c	a	◁

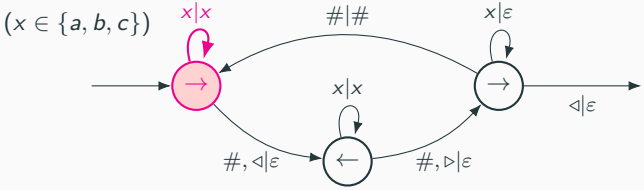
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

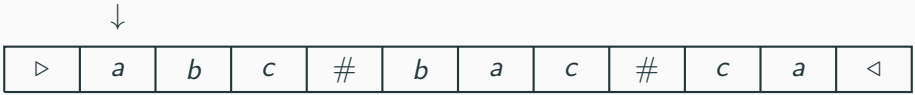
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{array}{rccc} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$



Output:



Deterministic two-way transducers (2DFT)

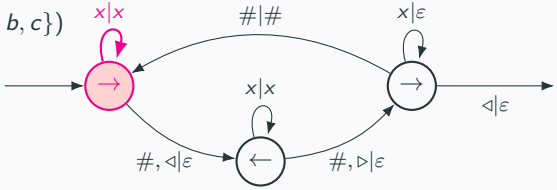
Two-way transducers: executive summary

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

Example:

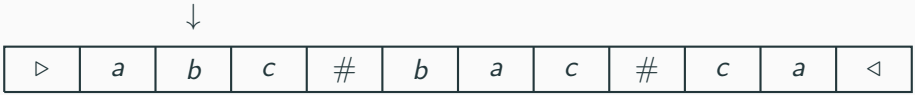
$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$

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



Output:

a



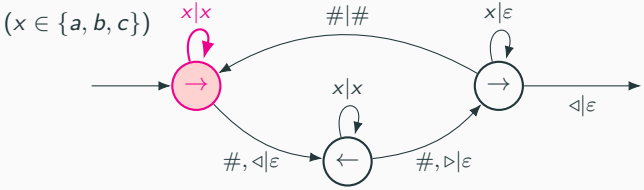
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



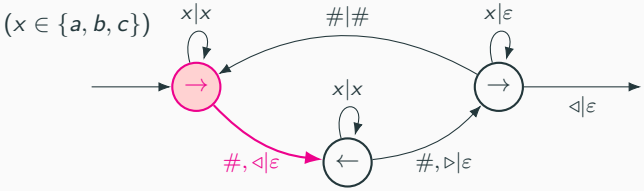
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

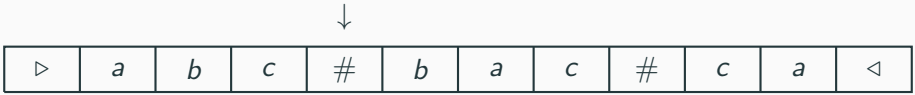
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abc



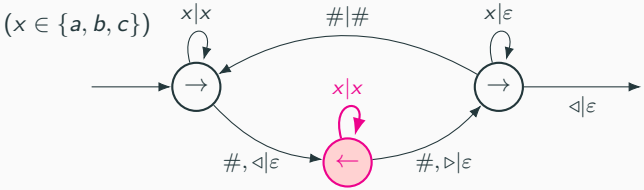
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

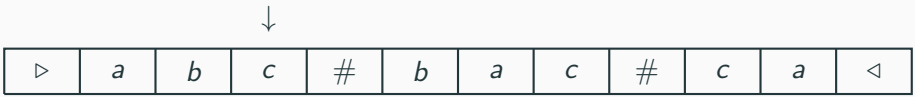
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{array}{lccl} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$



Output:
abc



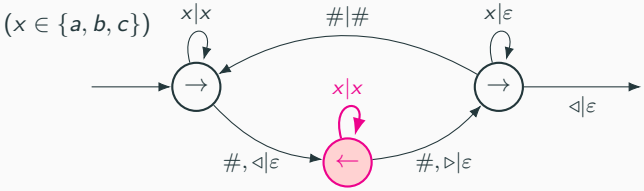
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

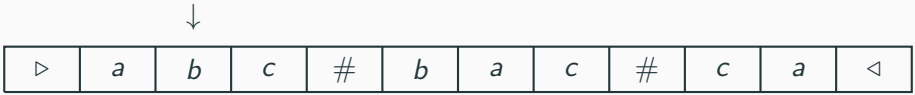
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abcc



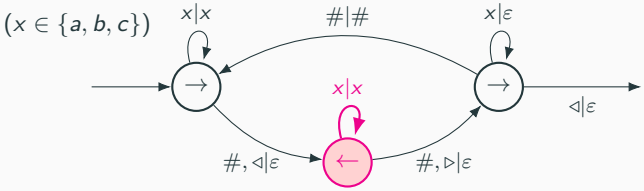
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

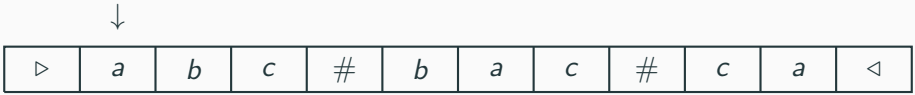
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccb



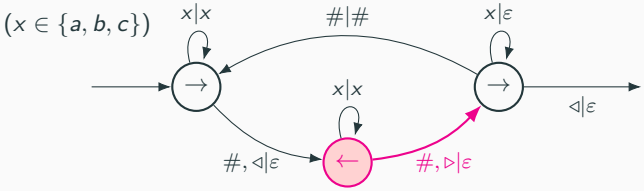
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

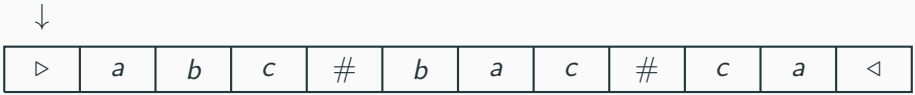
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{array}{lccl} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$



Output:
abccba



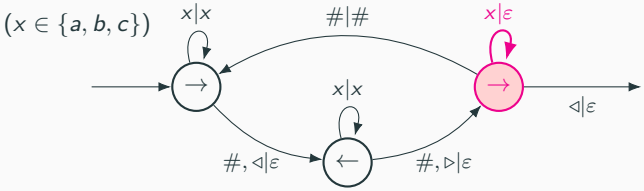
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

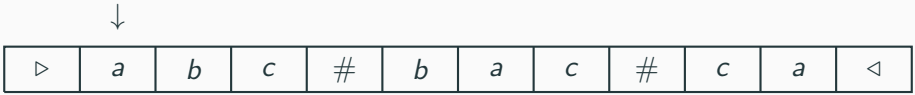
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{array}{lccl} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$



Output:
abccba



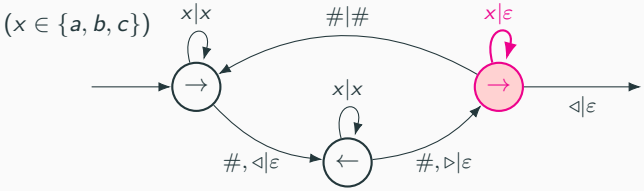
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

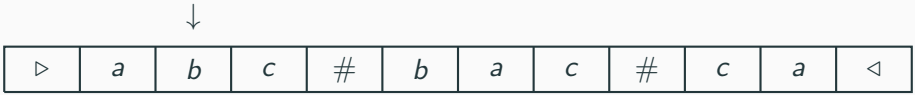
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\begin{array}{rccc} \texttt{mapPalin}: & \{a,b,c,\#\}^* & \longrightarrow & \{a,b,c,\#\}^* \\ & w_1\#\ldots\#w_n & \longmapsto & w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n) \end{array}$$



Output:
abccba



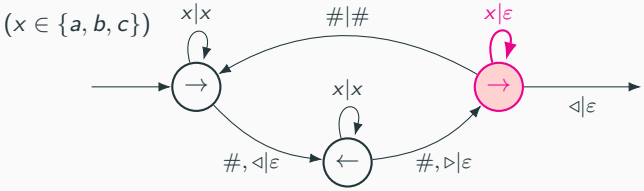
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba

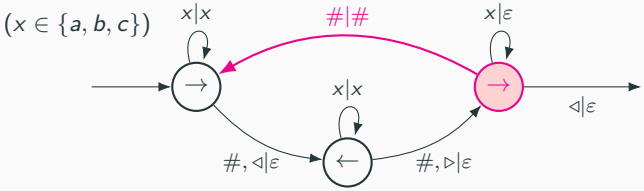
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

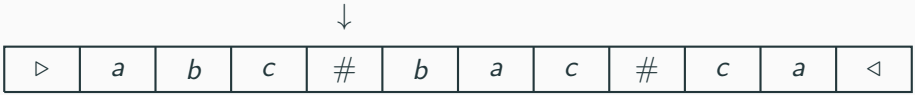
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba



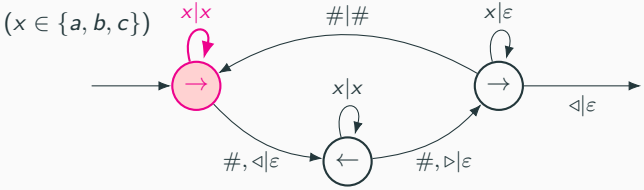
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

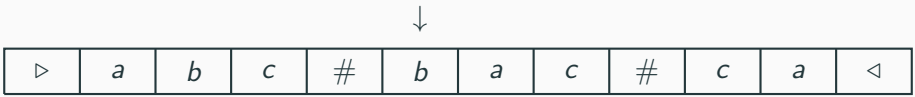
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba#



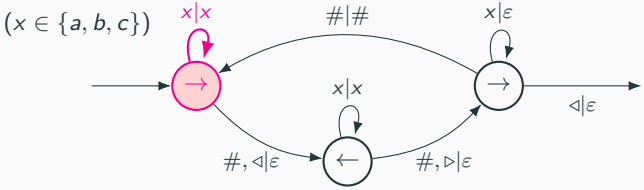
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

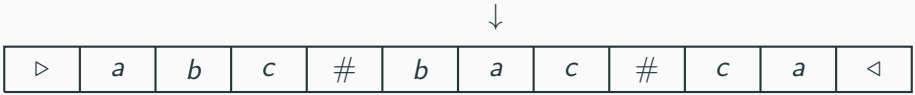
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba#b



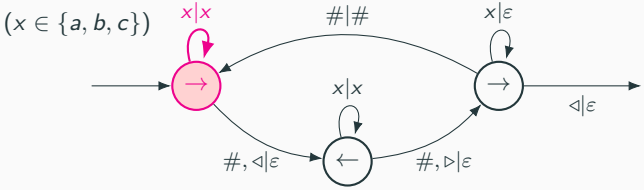
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

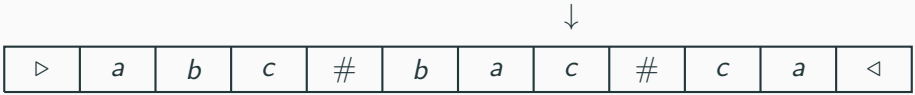
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba#ba



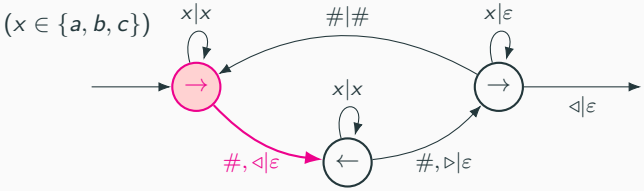
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

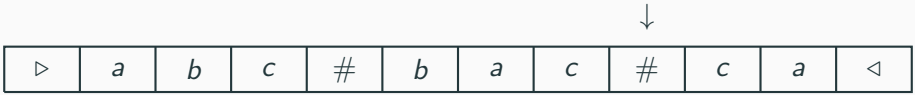
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#bac$



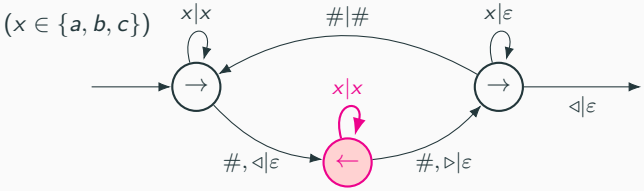
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

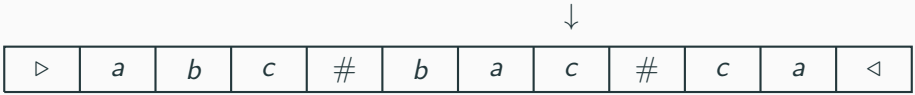
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
$abccba\#bac$



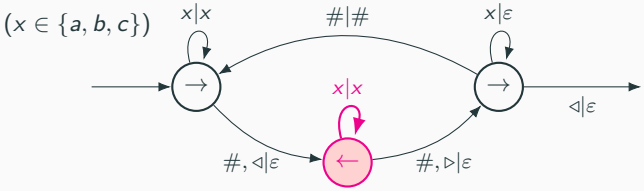
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

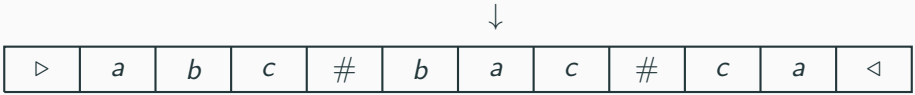
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
$abccba\#bacc$



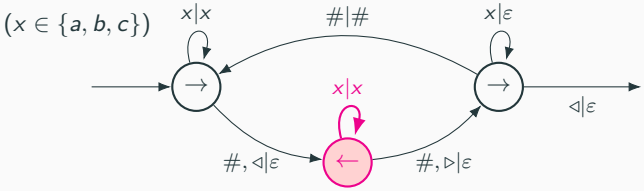
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

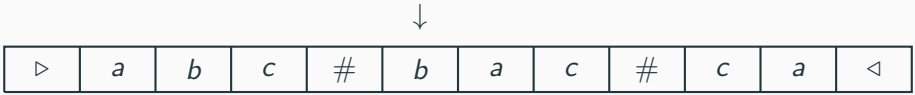
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
$abccba\#bacca$



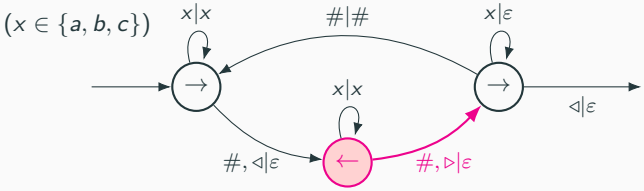
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

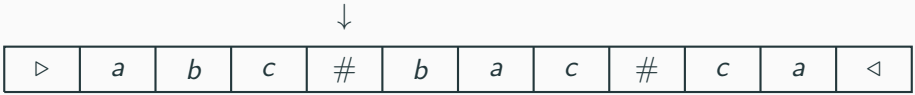
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba#baccab



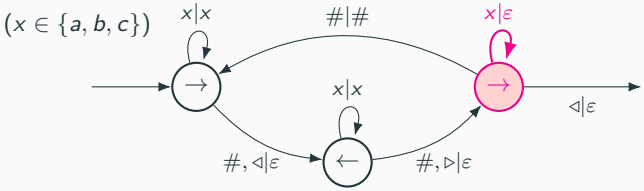
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

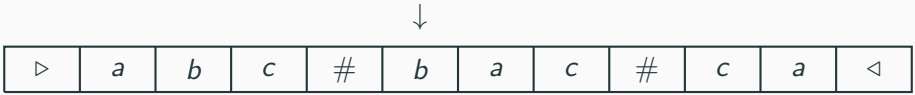
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
abccba#baccab



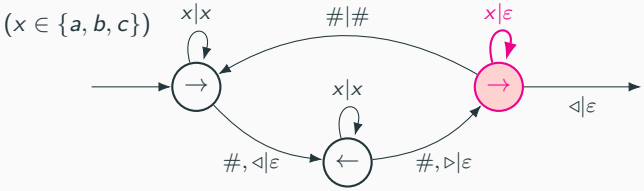
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

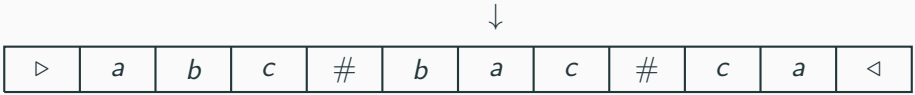
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
$abccba\#baccab$



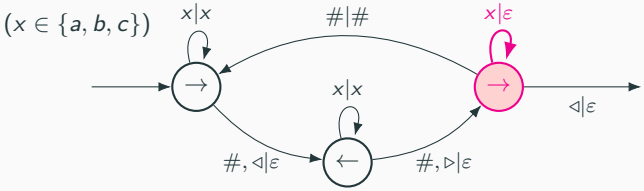
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

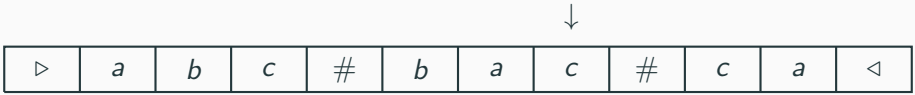
Finite set of states + bidirectional reading head + output produced from left to right

Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:
$abccba\#baccab$



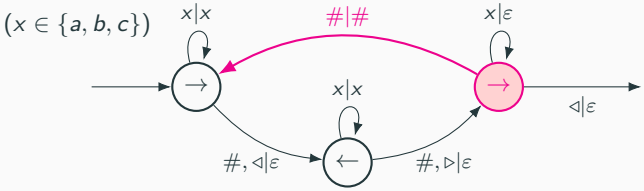
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

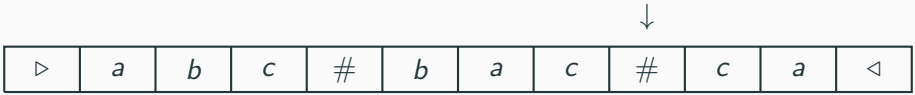
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab$



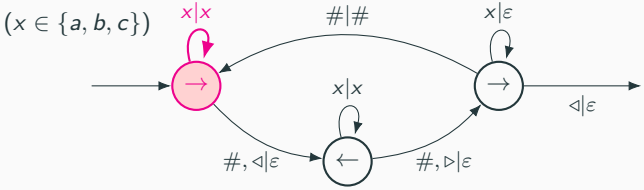
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

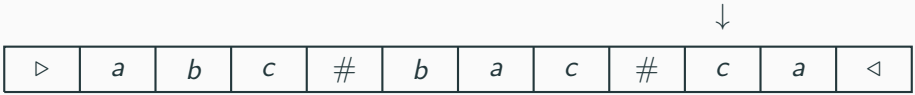
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#$



Deterministic two-way transducers (2DFT)

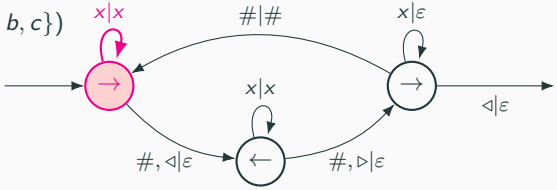
Two-way transducers: executive summary

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

Example:

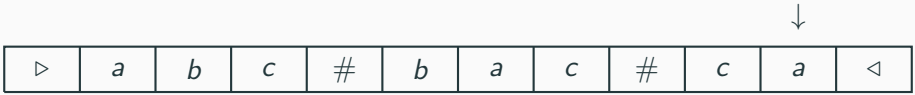
$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$

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



Output:

$abccba\#baccab\#c$



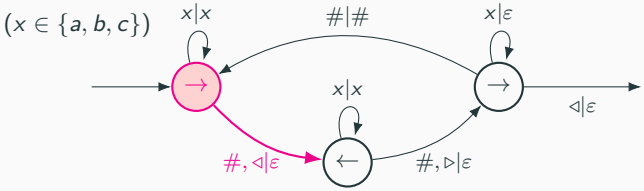
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

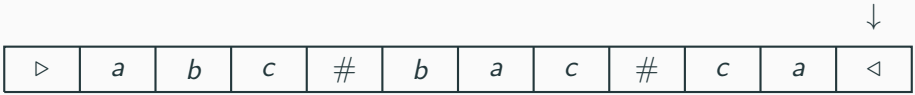
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

abccba#baccab#ca



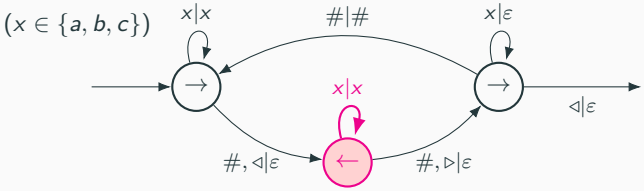
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

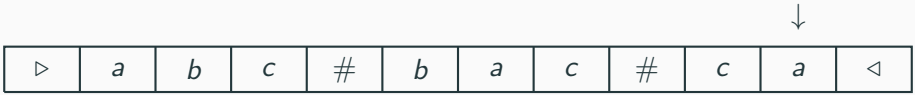
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#ca$



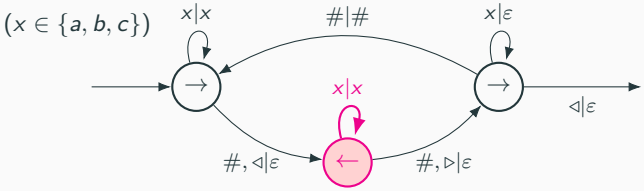
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

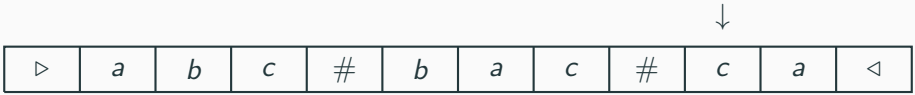
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#caa$



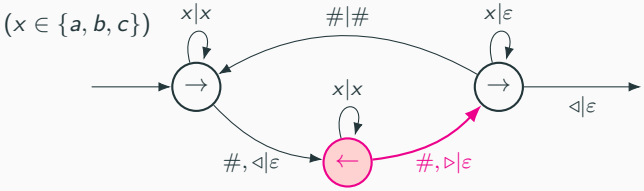
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

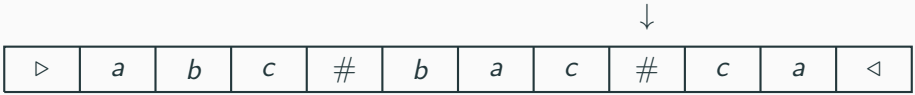
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#caac$



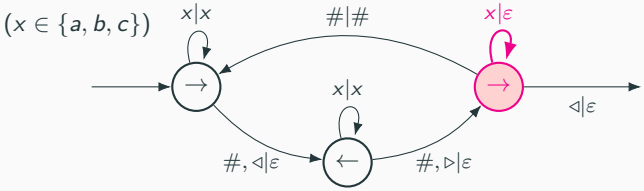
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

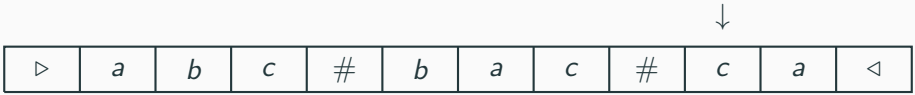
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#caac$



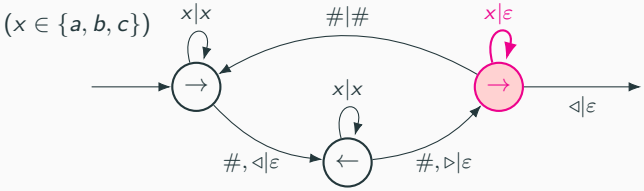
Deterministic two-way transducers (2DFT)

Two-way transducers: executive summary

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

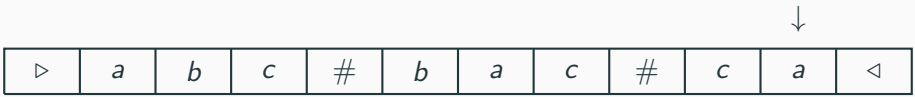
Example:

$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$



Output:

$abccba\#baccab\#caac$



Deterministic two-way transducers (2DFT)

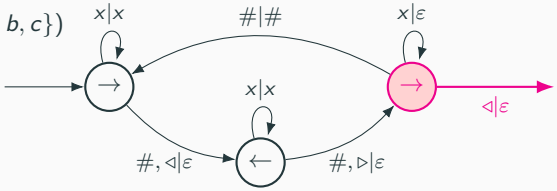
Two-way transducers: executive summary

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

Example:

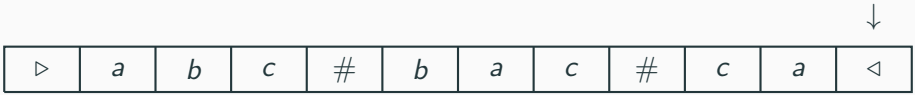
$$\texttt{mapPalin}: \quad \{a,b,c,\#\}^* \longrightarrow \{a,b,c,\#\}^*$$
$$w_1\#\ldots\#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\#\ldots\#w_n \cdot \texttt{reverse}(w_n)$$

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



Output:

$abccba\#baccab\#caac$



Regular functions

Functions $\Sigma^* \to \Gamma^*$ definable by 2DFTs are called **regular functions**

Properties of regular functions

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

Alternative characterizations

- Via Monadic Second-Order logic (MSO transductions)
- Copyless streaming string transducers
- Various functional programming or regexp-like (declarative) formalisms
- Minimal linear λ-calculus and Church encodings [Nguyễn,Noûs,P. 2020]

Polyregular functions

Polyregular functions:

- A larger class of string-to-string transductions
- Garnered significant attention recently, starting with [Bojańczyk 2018]

Properties

- *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
- Simply typed λ-calculus augmented with a list type and some list manipulation primitives
- Composition closure of [regular functions $\cup$ "squaring with underlining"]

Polyregular functions

Polyregular functions:

- A larger class of string-to-string transductions
- Garnered significant attention recently, starting with [Bojańczyk 2018]

Properties

- *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
- Simply typed λ-calculus augmented with a list type and some list manipulation primitives
- Composition closure of [regular functions $\cup$ "squaring with underlining"]
- **k-pebble string-to-string transducers**

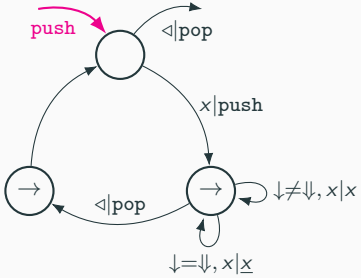
Pebble transducers

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" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$

▷	a	b	c	◁

output $=$

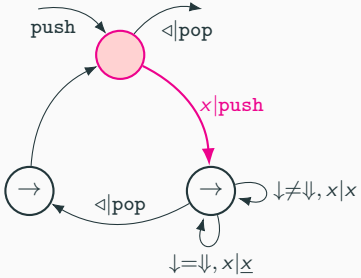
Pebble transducers

k-pebble transducers: executive summary

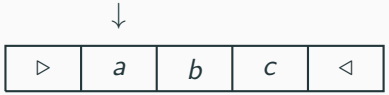
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



output $=$

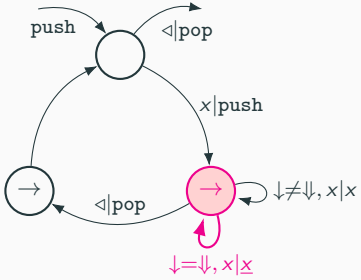
Pebble transducers

k-pebble transducers: executive summary

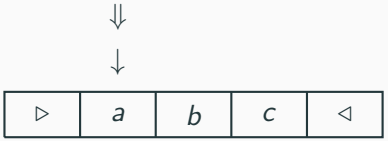
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



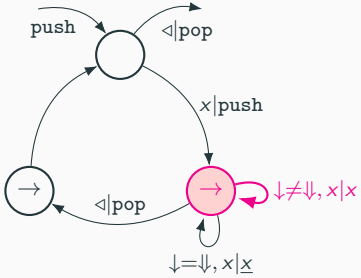
output $=$

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" ($k = 2$)



$$\begin{array}{rccc} \texttt{squaring}: & \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ & aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$

$\Downarrow$

$\downarrow$

▷	a	b	c	◁

output $= \underline{a}$

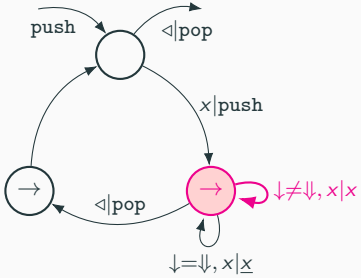
Pebble transducers

k-pebble transducers: executive summary

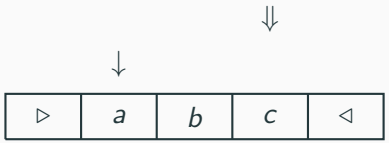
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



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

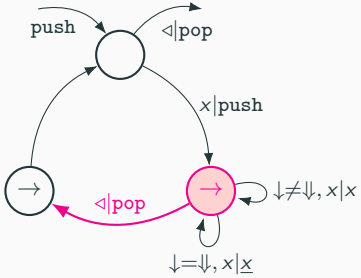
Pebble transducers

k-pebble transducers: executive summary

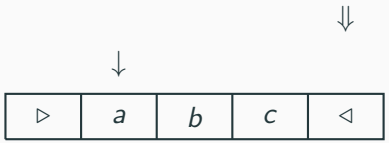
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



output $= \underline{a}bc$

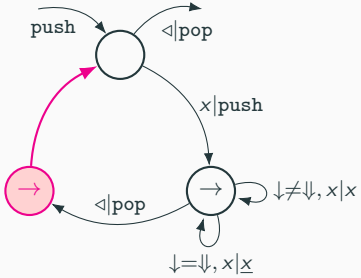
Pebble transducers

k-pebble transducers: executive summary

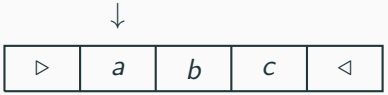
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{lcl} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



output $= \underline{a}bc$

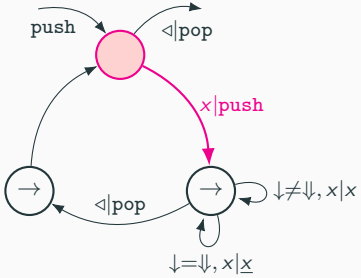
Pebble transducers

k-pebble transducers: executive summary

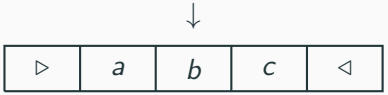
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\begin{array}{lccc} \texttt{squaring}: & \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ & aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



output $= \underline{a}bc$

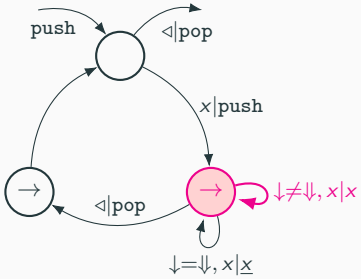
Pebble transducers

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" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$

$\Downarrow$

$\downarrow$

▷	a	b	c	◁

output $= \underline{a}bc$

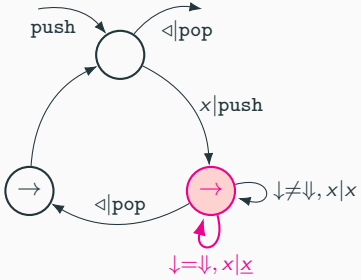
Pebble transducers

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" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$

$\Downarrow$

$\downarrow$

▷	a	b	c	◁

output $= \underline{a}bca$

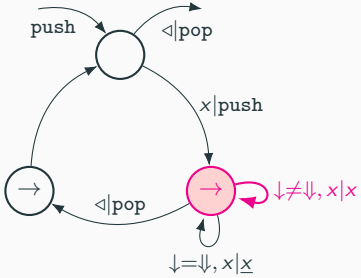
Pebble transducers

k-pebble transducers: executive summary

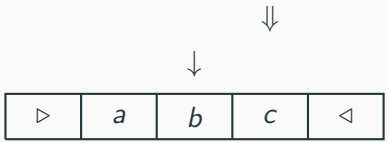
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



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

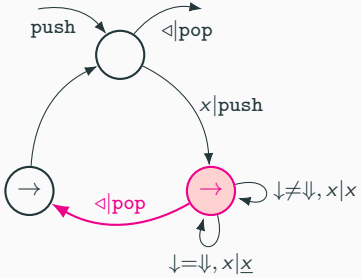
Pebble transducers

k-pebble transducers: executive summary

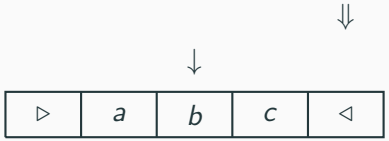
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



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

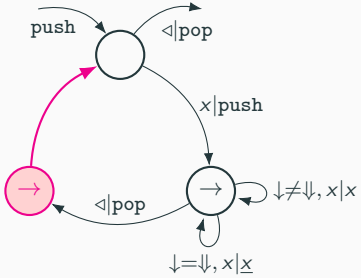
Pebble transducers

k-pebble transducers: executive summary

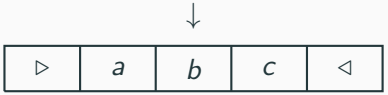
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



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

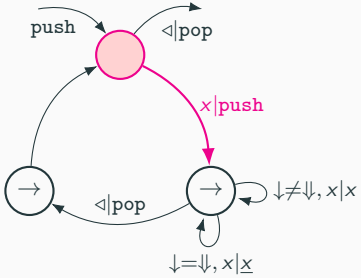
Pebble transducers

k-pebble transducers: executive summary

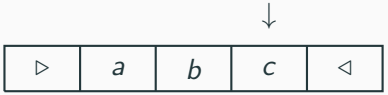
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{lcl} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



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

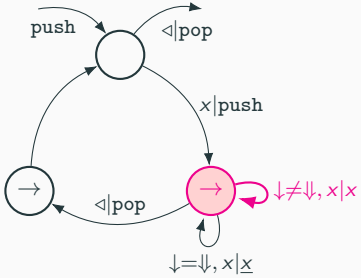
Pebble transducers

k-pebble transducers: executive summary

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

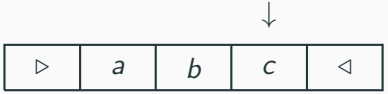
- Heads can be moved, pushed, popped
- Arbitrary comparisons between heads in the stack
- 1-pebble transducers $\cong$ 2DFTs

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \begin{array}{ccc} \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$

$\Downarrow$



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

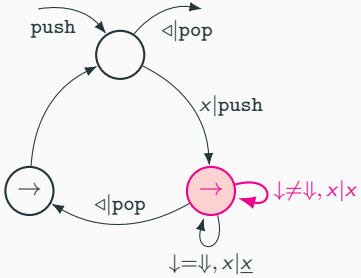
Pebble transducers

k-pebble transducers: executive summary

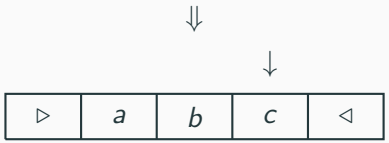
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



output $= \underline{a}bca\underline{b}ca$

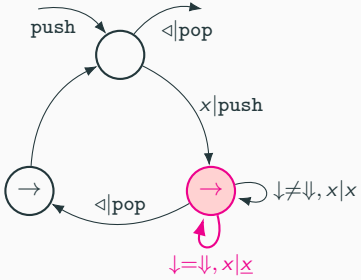
Pebble transducers

k-pebble transducers: executive summary

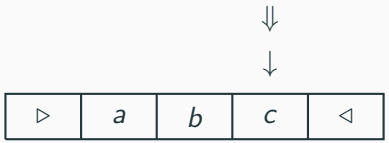
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



output $= \underline{a}bca\underline{b}cab$

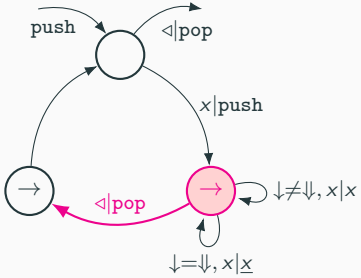
Pebble transducers

k-pebble transducers: executive summary

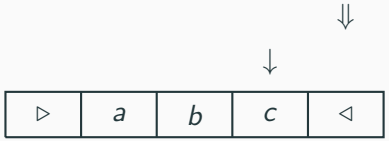
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



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

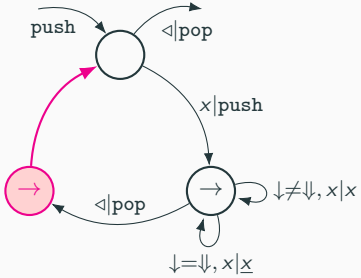
Pebble transducers

k-pebble transducers: executive summary

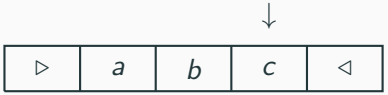
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\texttt{squaring}: \quad \Sigma^* \to (\Sigma \cup \underline{\Sigma})^*$$
$$aab \mapsto \underline{a}aba\underline{a}baa\underline{b}$$



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

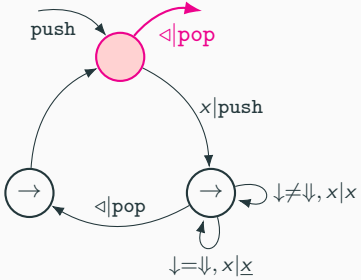
Pebble transducers

k-pebble transducers: executive summary

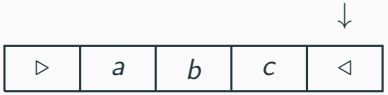
Finite set of states + a stack of two-way reading heads of height $\leq k$

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

Example: "squaring with underlining" ($k = 2$)



$$\begin{array}{lccc} \texttt{squaring}: & \Sigma^* & \to & (\Sigma \cup \underline{\Sigma})^* \\ & aab & \mapsto & \underline{a}aba\underline{a}baa\underline{b} \end{array}$$



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

Polyblind/comparison-free pebble transducers

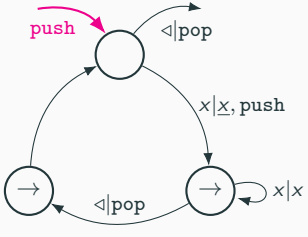
Main question

What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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



▷	a	b	c	◁

output =

Polyblind/comparison-free pebble transducers

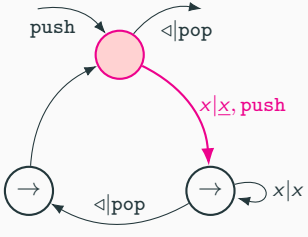
Main question

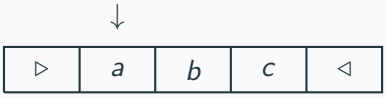
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output =

Polyblind/comparison-free pebble transducers

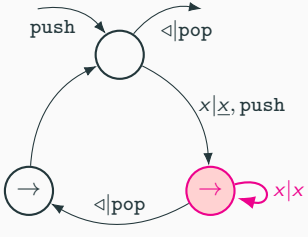
Main question

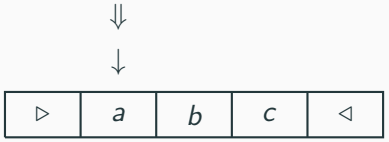
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}$

Polyblind/comparison-free pebble transducers

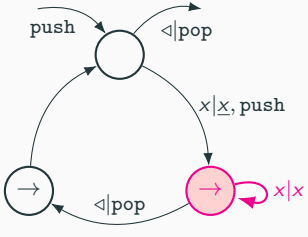
Main question

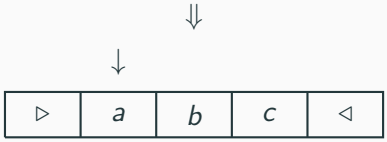
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}a$

Polyblind/comparison-free pebble transducers

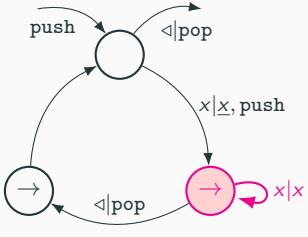
Main question

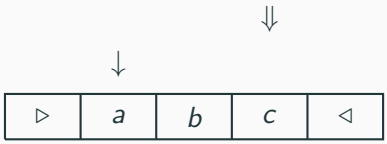
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}ab$

Polyblind/comparison-free pebble transducers

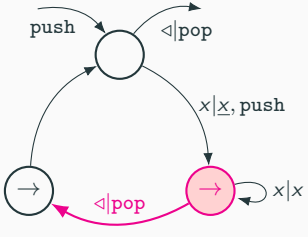
Main question

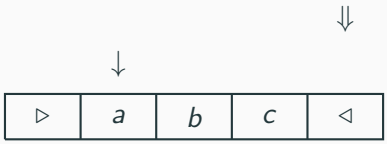
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}abc$

Polyblind/comparison-free pebble transducers

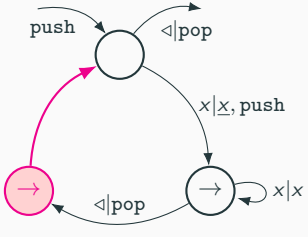
Main question

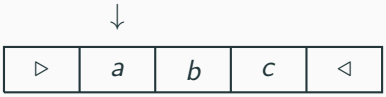
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}abc$

Polyblind/comparison-free pebble transducers

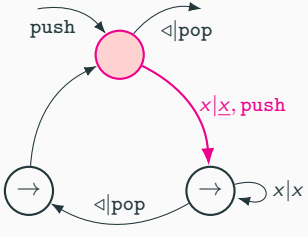
Main question

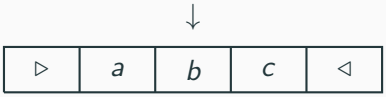
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





output $= \underline{a}abc$

Polyblind/comparison-free pebble transducers

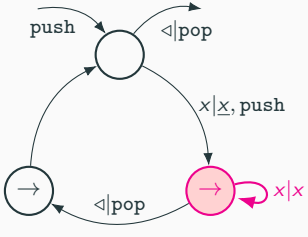
Main question

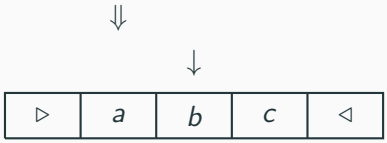
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

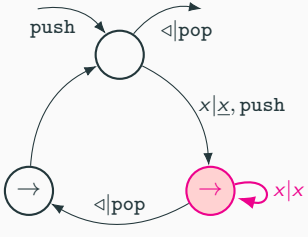
Polyblind/comparison-free pebble transducers

Main question

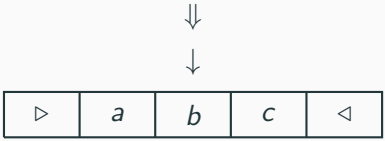
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"



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



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

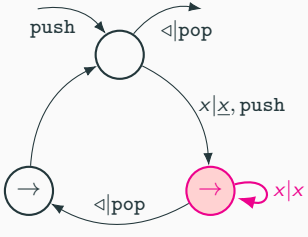
Polyblind/comparison-free pebble transducers

Main question

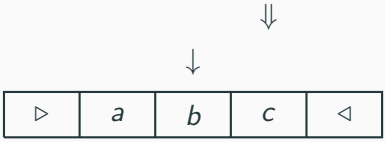
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"



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



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

Polyblind/comparison-free pebble transducers

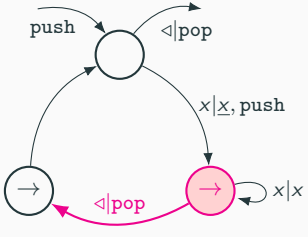
Main question

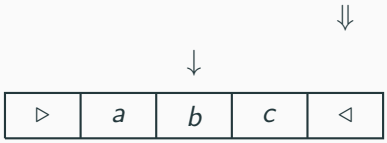
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

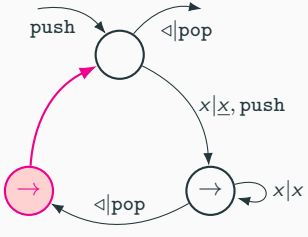
Main question

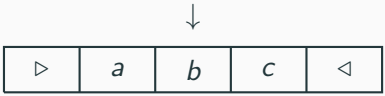
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

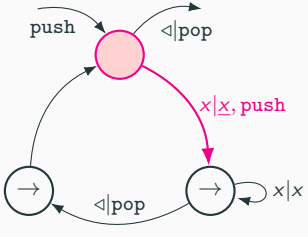
Main question

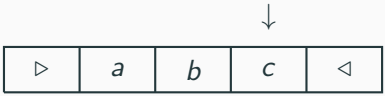
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

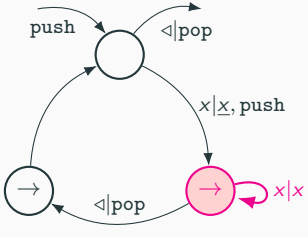
Main question

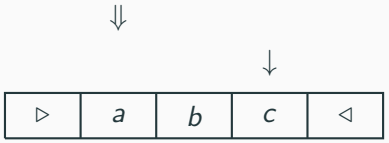
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

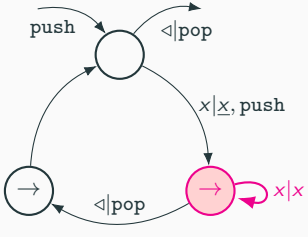
Main question

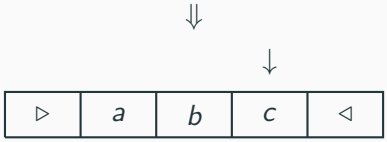
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

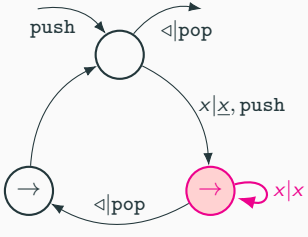
Main question

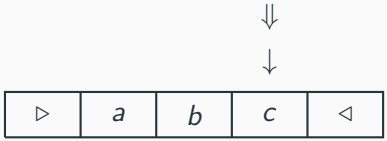
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

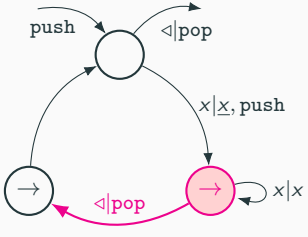
Main question

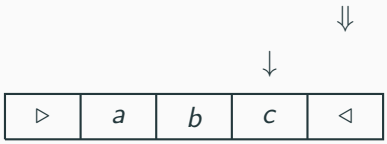
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

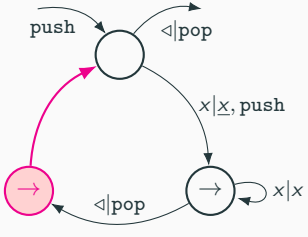
Main question

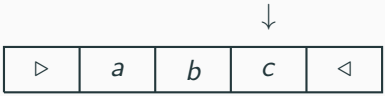
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

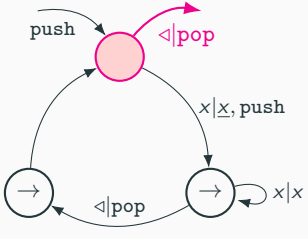
Main question

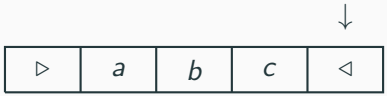
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Polyblind/comparison-free pebble transducers

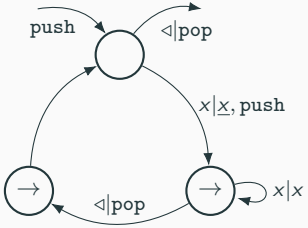
Main question

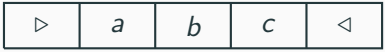
What happens if we disallow comparisons between reading heads?

Non-example: "squaring with underlining"

Example: "polyblind squaring"

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





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

Contributions

- Alternative characterizations
- Separation results
- Along the way: closure by composition, pebble minimization

Some alternative characterizations

Alternative characterization (1/2): composition by substitutions

Definition (Composition by substitutions)

Let Γ, Σ and I be finite alphabets and $f : \Gamma^* \to I^*$, $g_i : \Gamma^* \to \Sigma^*$ and $w \in \Gamma^*$.
Define $\mathrm{CbS}(f, (g_i)_{i\in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

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

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

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

- 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 $f : \Gamma^* \to I^*$, $g_i : \Gamma^* \to \Sigma^*$ and $w \in \Gamma^*$.
Define $\mathrm{CbS}(f, (g_i)_{i \in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

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

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

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

- 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 $\mathrm{CbS}(f, (g_i)_{i \in I})$ is polyblind

Alternative characterization (1/2): composition by substitutions

Definition (Composition by substitutions)

Let Γ, Σ and I be finite alphabets and $f : \Gamma^* \to I^*$, $g_i : \Gamma^* \to \Sigma^*$ and $w \in \Gamma^*$.
Define $\mathrm{CbS}(f, (g_i)_{i\in I})(w)$ so that, if $f(w) = i_1 \dots i_k$, then

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

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

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

- 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 $\mathrm{CbS}(f, (g_i)_{i\in I})$ is polyblind

- More convenient to manipulate formally
- Tight link between the number of pebbles and the nesting of the CbS operator

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

We have an alternative characterization based on linear the λ-calculus

- Not presented in the paper, mostly based on [Nguyễn,Noûs,P. 2020]
- Hints at the following non-trivial theorem

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

Closure under composition

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

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

We have an alternative characterization based on linear the λ-calculus

- Not presented in the paper, mostly based on [Nguyễn,Noûs,P. 2020]
- Hints at the following non-trivial theorem

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

Closure under composition

If $f : \Sigma^* \to \Gamma^*$ and $g : \Gamma^* \to \Delta^*$ 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 $\to$ all polyregular functions

- Regular functions can also themselves be decomposed

Not all polyregular functions are polyblind

Separation results

Theorem

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

Corollary: "squaring with underlining" is not CF.

Separation results

Theorem

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

Corollary: "squaring with underlining" is not CF.

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

$$a \quad aa \quad \ldots \quad a^n$$

Separation results

Theorem

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

Corollary: "squaring with underlining" is not CF.

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

$$a \quad aa \quad \ldots \quad a^n$$

Lemma

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

Separation results

Theorem

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

Corollary: "squaring with underlining" is not CF.

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

$$a \quad aa \quad \ldots \quad a^n$$

Lemma

For any polyblind $g : \{a\}^* \to \Sigma^*$, 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 {lengths of maximal a-factors of $g(a^n)$}.

Why? $\to$ characterization on the next slide

Characterizing polyblind 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)

$f : \{a\}^* \to \Sigma^*$ is polyblind iff $\exists p \in \mathbb{N} \forall m \in \mathbb{N}.$ $(f(a^{(n+1)p+m}))_{n\in\mathbb{N}}$ is poly-pumping

(meaning of the $\exists\forall$: "ultimately periodic combinations")

Another polyregular non-polyblind function

Theorem

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

([Douéneau-Tabot 2021] proves a stronger result)

Another polyregular non-polyblind function

Theorem

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

([Douéneau-Tabot 2021] proves a stronger result)

Another polyregular non-polyblind function

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.

([Douéneau-Tabot 2021] proves a stronger result)

Definition

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

$g = \mathrm{map}(w \mapsto w^{|w|})$ therefore polyblind functions are *not* closed under map, unlike regular and polyreg functions

$\to$ obstruction to characterizing polyblind fn
by list-processing functional programs
(à la [Bojańczyk, Daviaud & Krishna 2018])

Separation proof idea for "map unary square" via pebble minimization

Pebble minimization

If f is polyblind and $|f(w)| = O(|w|^k)$ then some polyblind k-pebble transducer computes f.

Technical proof adapted from a false result for pebble transducers [Lhote 2020].

Theorem

$g(a^{n_1}\#\ldots\#a^{n_k}) = a^{n_1\times n_1}\#\ldots\#a^{n_k\times n_k}$ *is* not *polyblind.*

Proof by contradiction: assume g is polyblind.

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

$$g = \mathrm{CbS}(f, (h_i)_{i\in I}) \quad \text{i.e.} \quad f(w) = i_1 \ldots i_m \implies g(w) = h_{i_1}(w)\ldots h_{i_m}(w)$$

To conclude:

pumping argument + pigeonhole principle, exploiting the linear asymptotic growth

Might be doable without pebble minimization, but convenient and of independent interest

Further topics

Going first-order

First-order (FO)-regular functions

Robust subclass of regular functions; several characterizations:

- Logic: replace MSO by first-order logic
- 2DFT with *aperiodic* monoid of behaviors
- Functional programming or regexp-like e.g. [Dartois, Gastin & Krishna 2021]

$\rightsquigarrow$ Analogous class of FO-polyregular.

Going first-order

First-order (FO)-regular functions

Robust subclass of regular functions; several characterizations:

- Logic: replace MSO by first-order logic
- 2DFT with *aperiodic* monoid of behaviors
- Functional programming or regexp-like e.g. [Dartois, Gastin & Krishna 2021]

$\rightsquigarrow$ Analogous class of FO-polyregular. What about polyblind functions?

Going first-order

First-order (FO)-regular functions

Robust subclass of regular functions; several characterizations:

- Logic: replace MSO by first-order logic
- 2DFT with *aperiodic* monoid of behaviors
- Functional programming or regexp-like e.g. [Dartois, Gastin & Krishna 2021]

$\rightsquigarrow$ Analogous class of FO-polyregular. What about polyblind functions?

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 $\mathrm{CbS}(f, (g_i)_{i \in I})$ is FO-polyblind

Further topics

A few relevant directions:

- Extending this class to tree-to-tree functions and look for characterizations

  A linear λ-calculus characterization matches an analogue of the CbS definition

Further topics

A few relevant directions:

- Extending this class to tree-to-tree functions and look for characterizations

  A linear λ-calculus characterization matches an analogue of the CbS definition

- Equivalence, separation and membership problems, in the spirit of:

Theorem [Douéneau-Tabot 2021]

There is an algorithm with

- **Input:** a pebble transducer implementing a function f with quadratic growth
- **Output:** a polyblind transducer implementing f, or an error if there is none

Further topics

A few relevant directions:

- Extending this class to tree-to-tree functions and look for characterizations

  A linear λ-calculus characterization matches an analogue of the CbS definition

- Equivalence, separation and membership problems, in the spirit of:

Theorem [Douéneau-Tabot 2021]

There is an algorithm with

- **Input:** a pebble transducer implementing a function f with quadratic growth
- **Output:** a polyblind transducer implementing f, or an error if there is none

- Non-commutative linear λ-calculus characterization for the FO case

Conclusion

Summary

A class of string-to-string functions: *polyblind functions*.

Equivalent definitions

- By polyblind pebble transducers
- Inductively (composition by substitution)
- **As the composition closure of regular functions** + $\texttt{cfsquaring}(abc) = \underline{a}abc\underline{b}abc\underline{c}abc$

- L regular language $\implies$ $f^{-1}(L)$ also regular
- Polynomial growth: $|f(w)| = O(|w|^k)$
  - **pebble minimization theorem:** $k =$ **number of heads necessary to compute** f
- Strictly included in polyregular functions
  - $a^n \mapsto a\#aa\#\dots\#a^n$ and "map unary square" are polyregular but not polyblind
  - for $\{a\}^* \to \{a\}^*$ polyblind = polyreg
- Incomparable with polynomial HDT0L transductions
  - $a^n \mapsto a\#aa\#\dots\#a^n$ not polyblind but HDT0L
  - $w \mapsto w^{|w|}$ is polyblind but not HDT0L
- Well-behaved first-order counterpart

Summary

A class of string-to-string functions: *polyblind functions*.

Equivalent definitions

- By polyblind pebble transducers
- Inductively (composition by substitution)
- **As the composition closure of regular functions** + $\texttt{cfsquaring}(abc) = \underline{a}abc\underline{b}abc\underline{c}abc$

- L regular language $\implies$ $f^{-1}(L)$ also regular
- Polynomial growth: $|f(w)| = O(|w|^k)$
  - **pebble minimization theorem:** $k =$ **number of heads necessary to compute** f
- Strictly included in polyregular functions
  - $a^n \mapsto a\#aa\#\dots\#a^n$ and "map unary square" are polyregular but not polyblind
  - for $\{a\}^* \to \{a\}^*$ polyblind = polyreg
- Incomparable with polynomial HDT0L transductions
  - $a^n \mapsto a\#aa\#\dots\#a^n$ not polyblind but HDT0L
  - $w \mapsto w^{|w|}$ is polyblind but not HDT0L
- Well-behaved first-order counterpart

Thanks for your attention! Questions?