Polyregular functions: some recent developments

Nguyễn Lê Thành Dũng (a.k.a. Tito) — nltd@nguyentito.eu — ÉNS Lyon
joint work with Sandra Kiefer (Oxford) and Cécilia Pradic (Swansea)

Links seminar, Inria Lille, January 20th, 2023

Usual transduction classes

Several generalizations of regular languages to *functions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{sequential functions}}_{\substack{\textit{deterministic} \text{ finite transducers}}} \subsetneq \underbrace{\text{rational functions}}_{\substack{\textit{nondeterministic} \text{ transducers}}} \subsetneq \text{regular functions}$$

Usual transduction classes

Several generalizations of regular languages to *functions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{sequential functions}}_{\textit{deterministic}\text{ finite transducers}} \subsetneq \underbrace{\text{rational functions}}_{\textit{nondeterministic}\text{ transducers}} \subsetneq \text{regular functions}$$

Nice properties, e.g. L regular $\implies f^{-1}(L)$ regular

Linear growth: $|f(w)| = O(|w|)$.

Usual transduction classes

Several generalizations of regular languages to *functions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{sequential functions}}_{\textit{deterministic}\text{ finite transducers}} \subsetneq \underbrace{\text{rational functions}}_{\textit{nondeterministic}\text{ transducers}} \subsetneq \text{regular functions}$$

Nice properties, e.g. L regular $\implies f^{-1}(L)$ regular

Linear growth: $|f(w)| = O(|w|)$. Beyond that:

- some old (hyper)exp classes: L-systems, iterated pushdown transducers, ...
- *polyregular functions*, with $|f(w)| = |w|^{O(1)}$

Usual transduction classes

Several generalizations of regular languages to *functions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{sequential functions}}_{\textit{deterministic}\text{ finite transducers}} \subsetneq \underbrace{\text{rational functions}}_{\textit{nondeterministic}\text{ transducers}} \subsetneq \text{regular functions}$$

Nice properties, e.g. L regular $\implies f^{-1}(L)$ regular

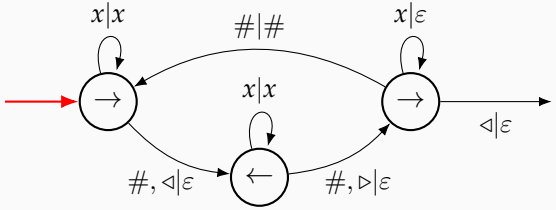
Linear growth: $|f(w)| = O(|w|)$. Beyond that:

- some old (hyper)exp classes: L-systems, iterated pushdown transducers, ...
- *polyregular functions*, with $|f(w)| = |w|^{O(1)}$

Let's start by defining regular and polyregular functions

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



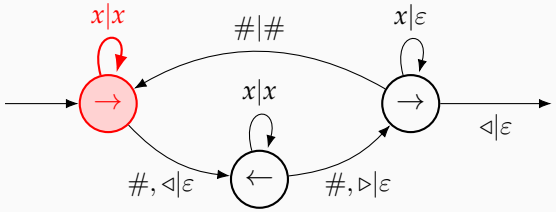
$(x \in \{a, b, c\})$

$\triangleright$	a	b	c	$\#$	b	a	c	$\#$	c	b	$\triangleleft$

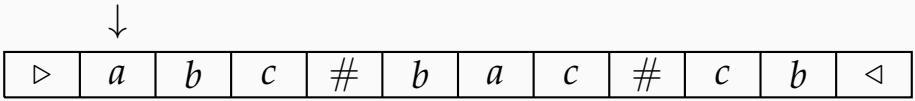
Output:

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



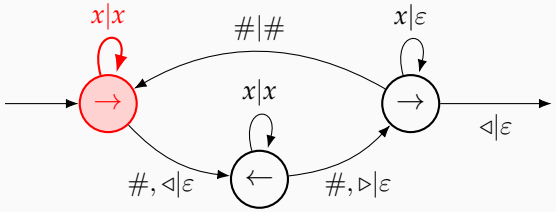
$(x \in \{a, b, c\})$



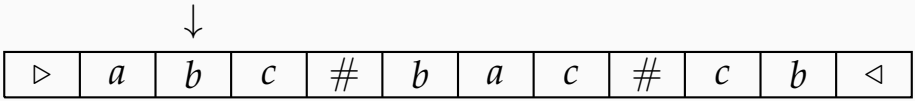
Output:

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



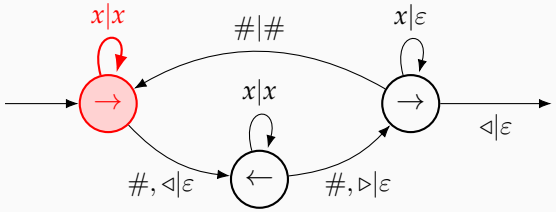
$(x \in \{a, b, c\})$



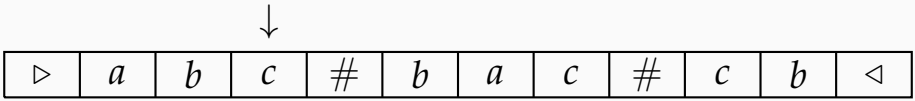
Output: a

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



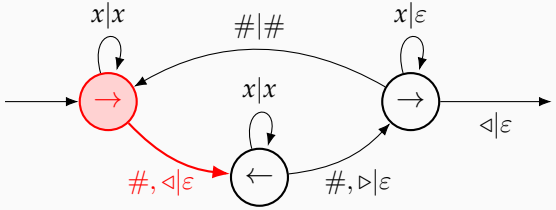
$(x \in \{a, b, c\})$



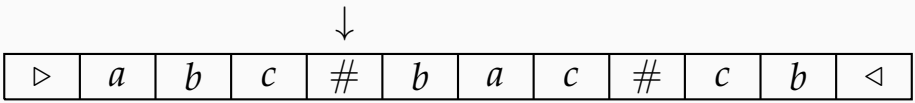
Output: ab

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



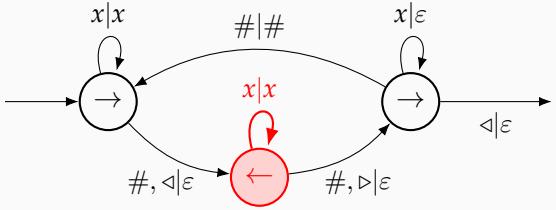
$(x \in \{a, b, c\})$



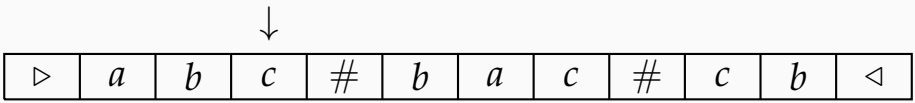
Output: abc

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



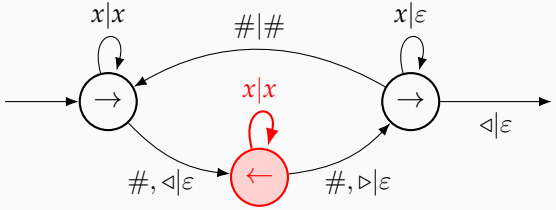
$(x \in \{a, b, c\})$



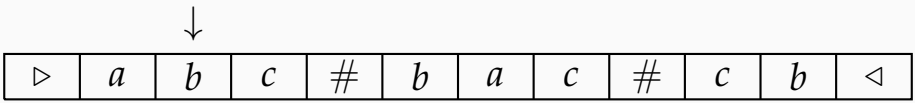
Output: abc

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



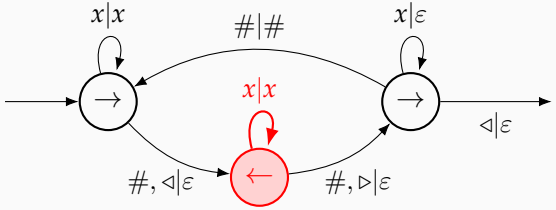
$(x \in \{a, b, c\})$



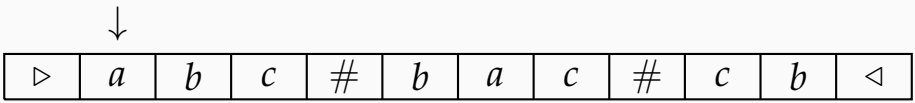
Output: *abcc*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



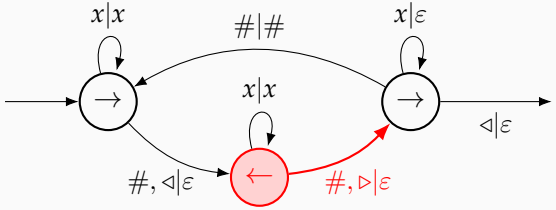
$(x \in \{a, b, c\})$



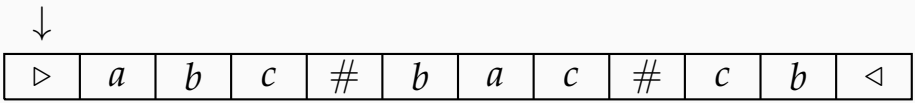
Output: *abccb*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



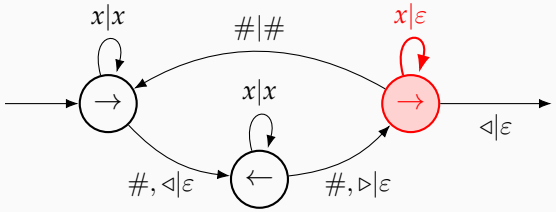
$(x \in \{a, b, c\})$



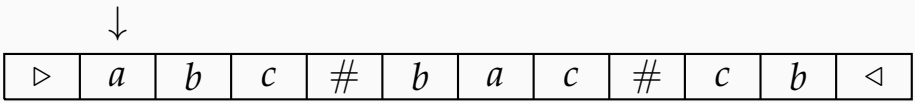
Output: *abccba*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



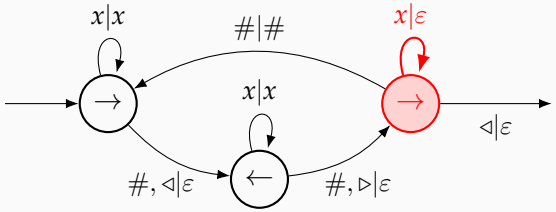
$(x \in \{a, b, c\})$



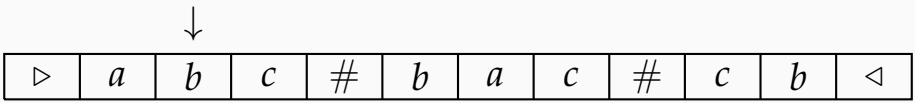
Output: *abccba*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



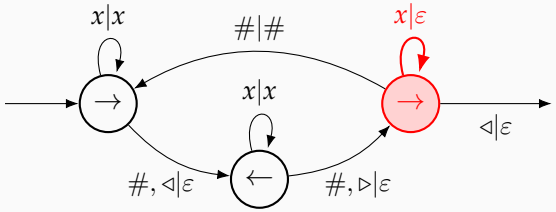
$(x \in \{a, b, c\})$



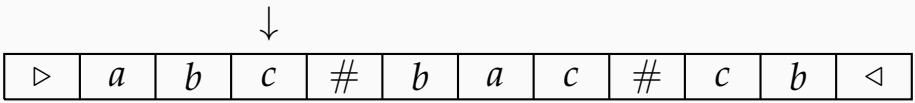
Output: *abccba*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



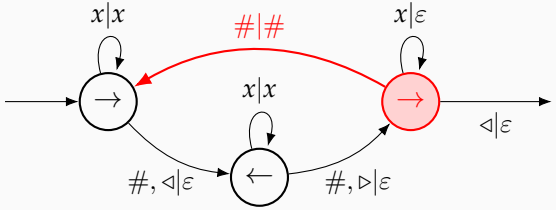
$(x \in \{a, b, c\})$



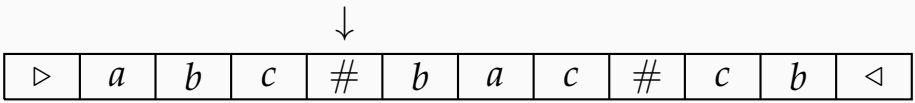
Output: *abccba*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



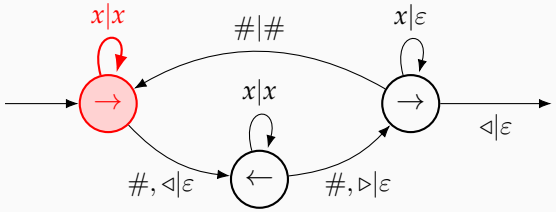
$(x \in \{a, b, c\})$



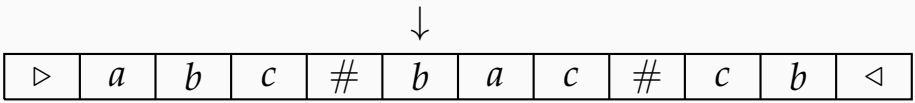
Output: *abccba*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



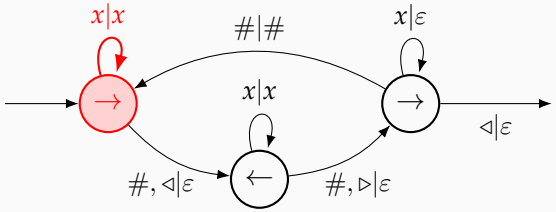
$(x \in \{a, b, c\})$



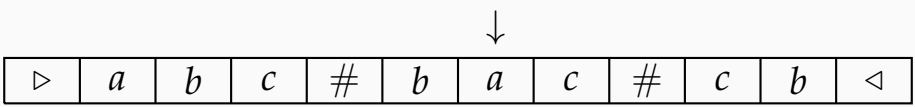
Output: $abccba\#$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



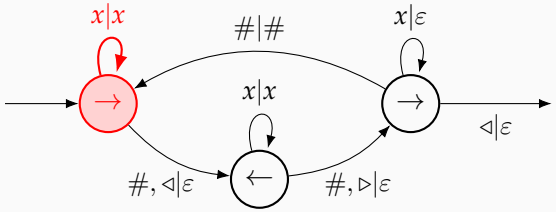
$(x \in \{a, b, c\})$



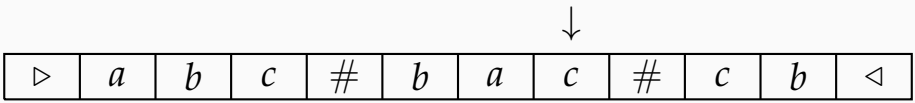
Output: $abccba\#b$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



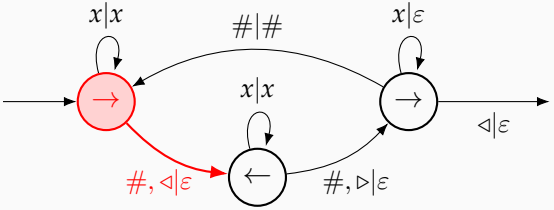
$(x \in \{a, b, c\})$



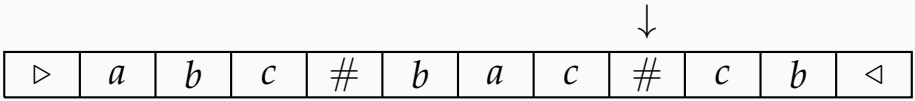
Output: $abccba\#ba$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



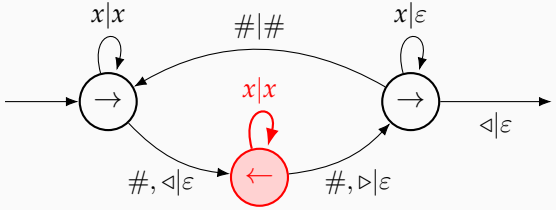
$(x \in \{a, b, c\})$



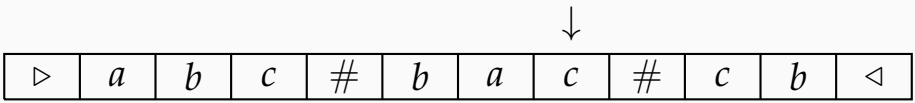
Output: $abccba\#bac$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



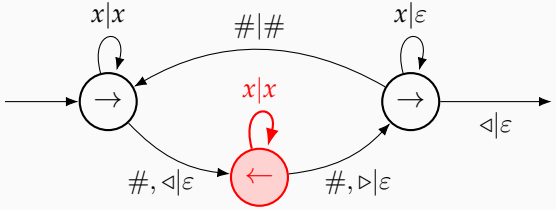
$(x \in \{a, b, c\})$



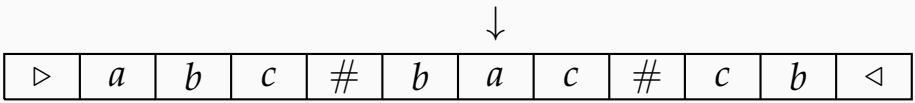
Output: $abccba\#bac$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



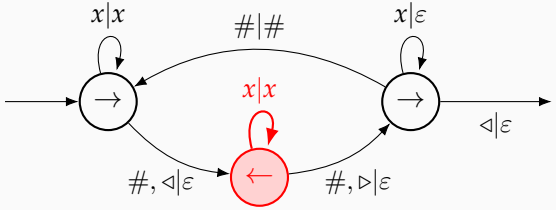
$(x \in \{a, b, c\})$



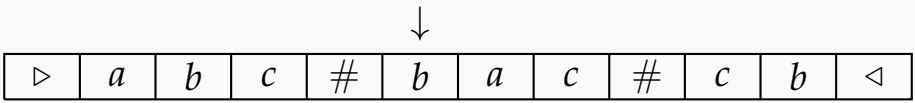
Output: *abccba#bacc*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



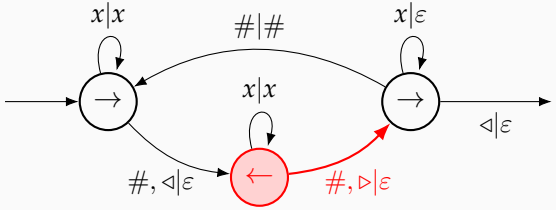
$(x \in \{a, b, c\})$



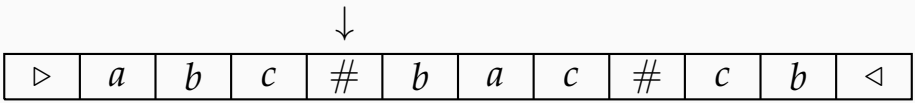
Output: *abccba#bacca*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



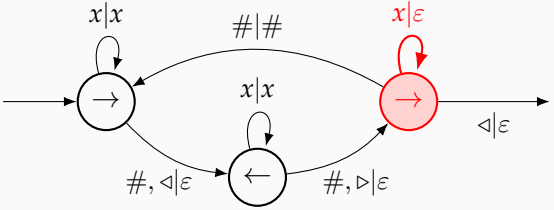
$(x \in \{a, b, c\})$



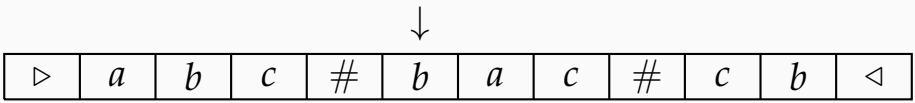
Output: *abccba#baccab*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



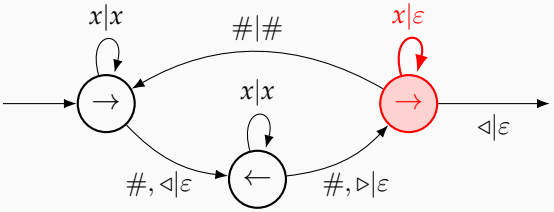
$(x \in \{a, b, c\})$



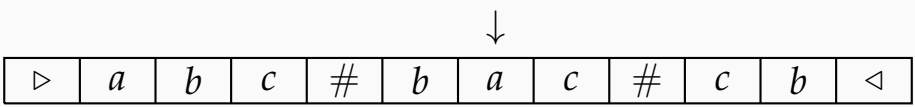
Output: *abccba#baccab*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



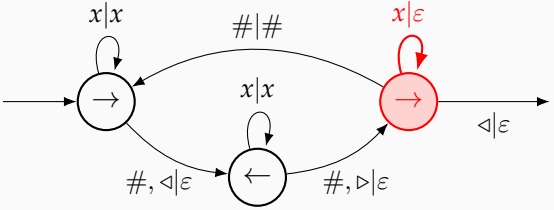
$(x \in \{a, b, c\})$



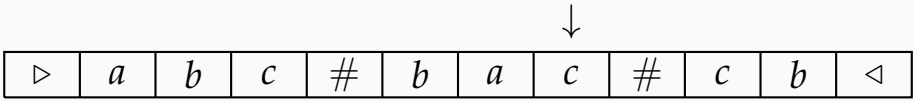
Output: *abccba#baccab*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



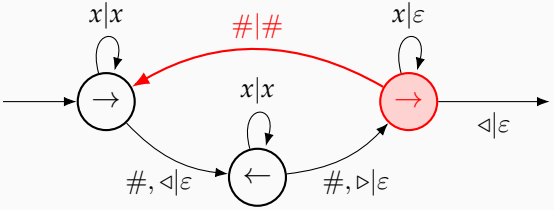
$(x \in \{a, b, c\})$



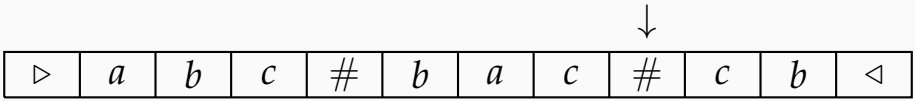
Output: *abccba#baccab*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



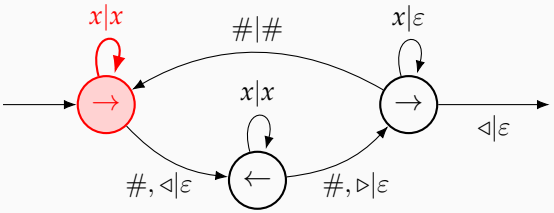
$(x \in \{a, b, c\})$



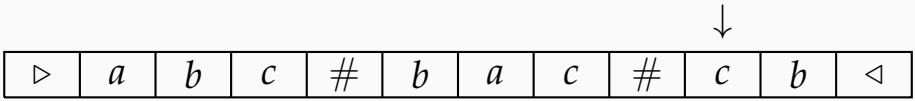
Output: *abccba#baccab*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



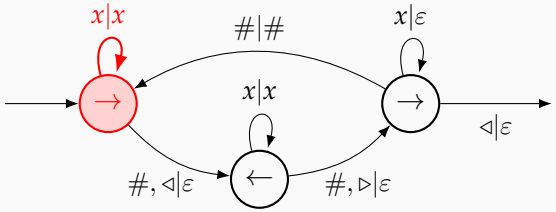
$(x \in \{a, b, c\})$



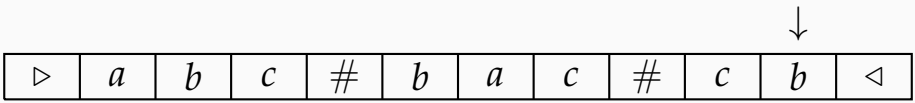
Output: $abccba\#baccab\#$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



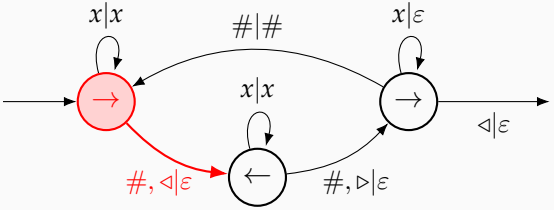
$(x \in \{a, b, c\})$



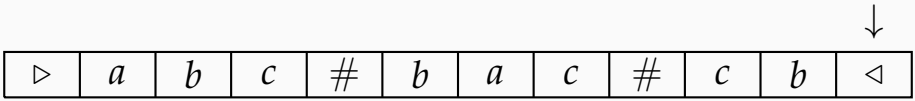
Output: $abccba\#baccab\#c$

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



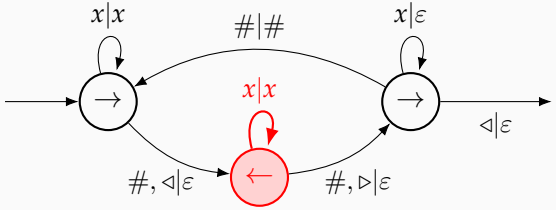
$(x \in \{a, b, c\})$



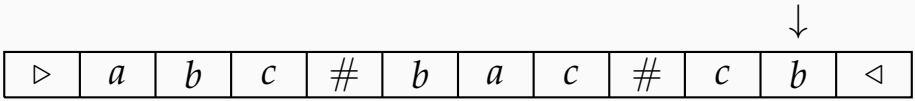
Output: *abccba#baccab#cb*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



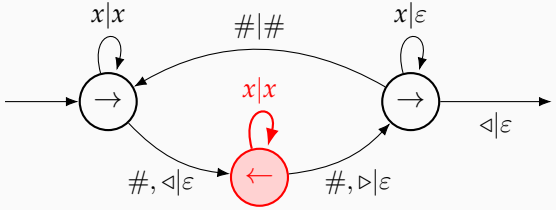
$(x \in \{a, b, c\})$



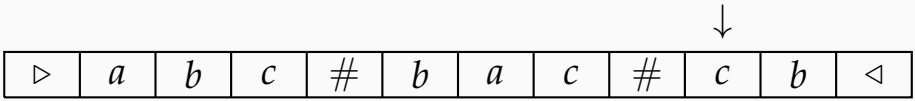
Output: *abccba#baccab#cb*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



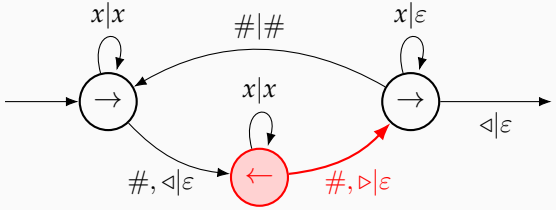
$(x \in \{a, b, c\})$



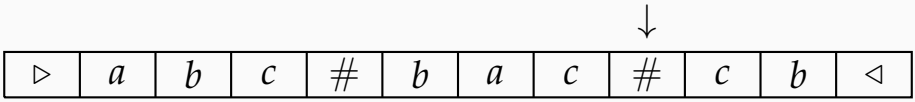
Output: *abccba#baccab#cbb*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



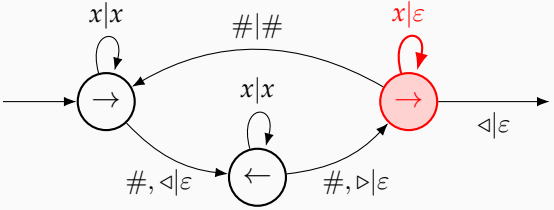
$(x \in \{a, b, c\})$



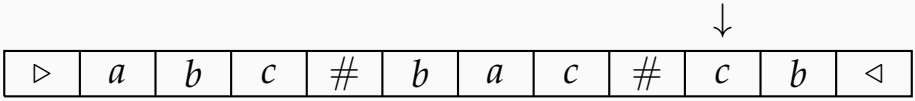
Output: *abccba#baccab#cbbc*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



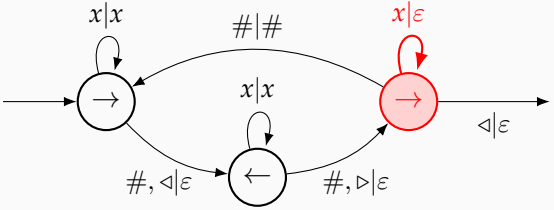
$(x \in \{a, b, c\})$



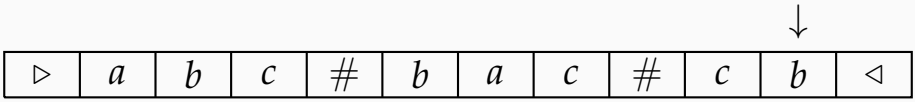
Output: *abccba#baccab#cbbc*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



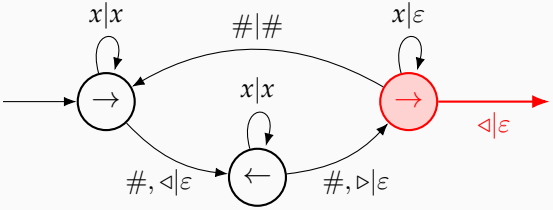
$(x \in \{a, b, c\})$



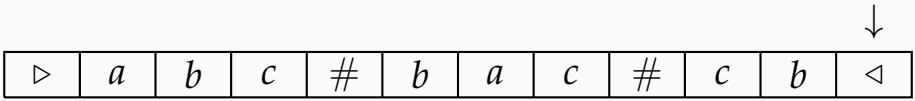
Output: *abccba#baccab#cbbc*

Two-way transducers

Example: $w_1\# \ldots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \ldots \#w_n \cdot \texttt{reverse}(w_n)$



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



Output: *abccba#baccab#cbbc*

(Poly)regular functions

Definition

Regular functions = computed by two-way transducers
(the reading head can move left or right at each transition)

(Poly)regular functions

Definition

Regular functions = computed by two-way transducers
(the reading head can move left or right at each transition)

To get polynomial growth: multiple heads!

- with no restriction, you get Logspace [Hartmanis 1972]
- idea: impose a "stack condition" on the heads
  $\longrightarrow$ *pebble transducers*, the historical definition of polyregular functions

(Poly)regular functions

Definition

Regular functions = computed by two-way transducers
(the reading head can move left or right at each transition)

To get polynomial growth: multiple heads!

- with no restriction, you get Logspace [Hartmanis 1972]
- idea: impose a "stack condition" on the heads
  $\longrightarrow$ *pebble transducers*, the historical definition of polyregular functions

Many equivalent definitions; we will see a few of them
for polyregular functions, all those alternatives are recent
[Bojańczyk 2018, 2023; Bojańczyk, Kiefer & Lhote 2019]

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

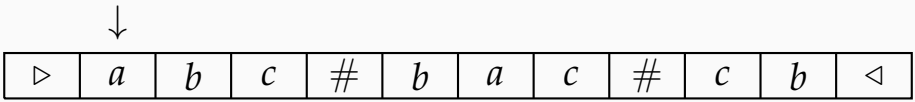
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



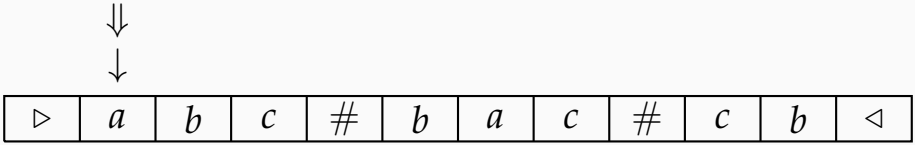
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



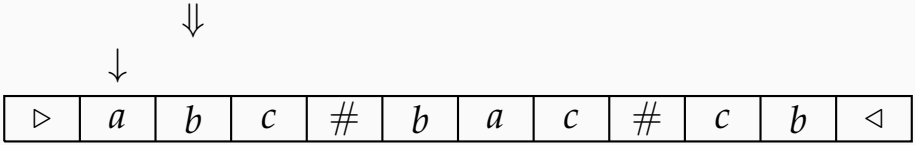
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



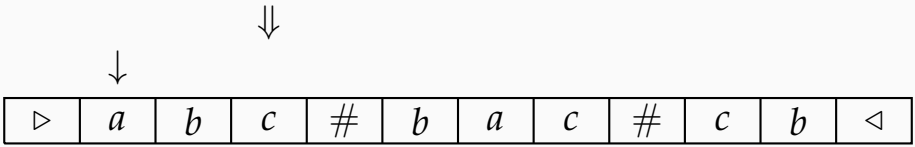
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



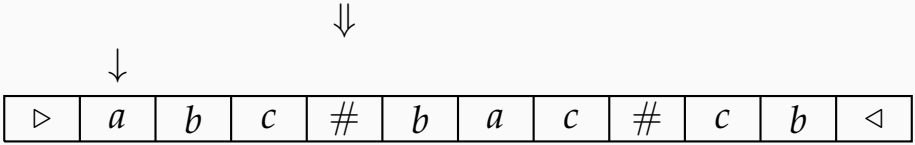
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



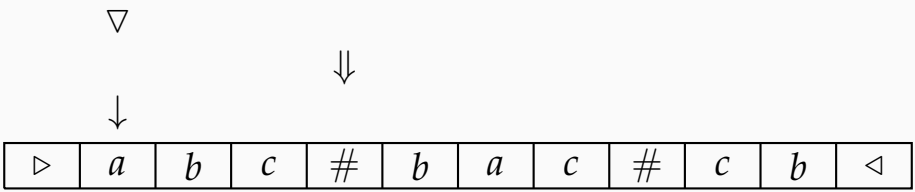
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



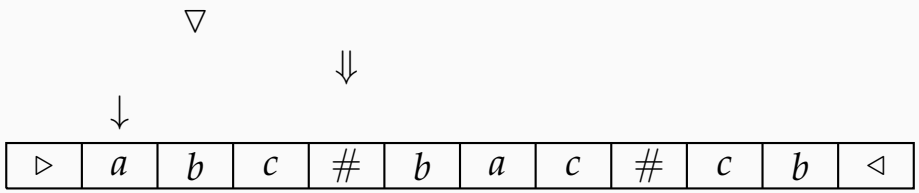
Output:

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



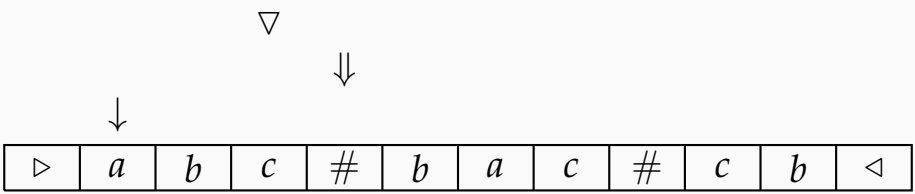
Output: a

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



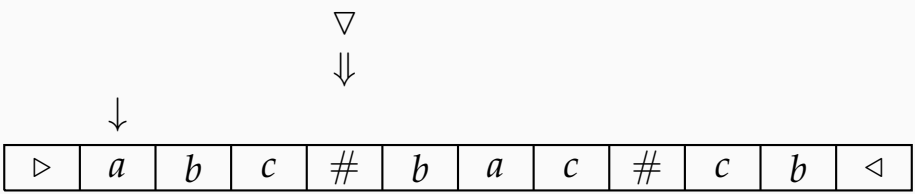
Output: ab

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



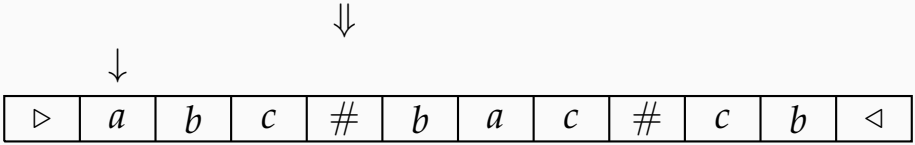
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



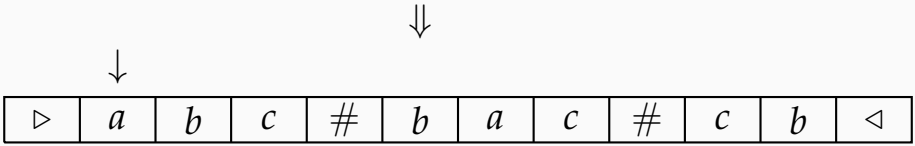
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



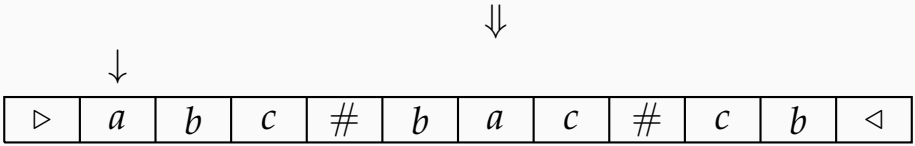
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



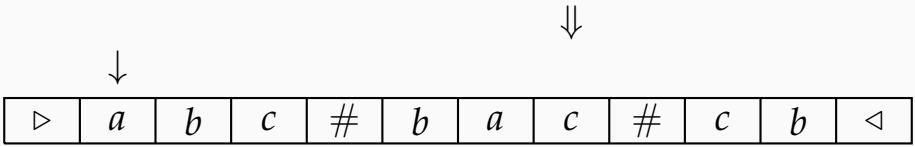
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



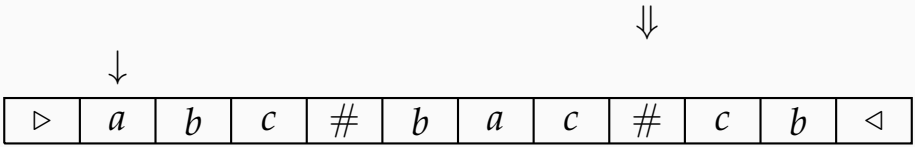
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



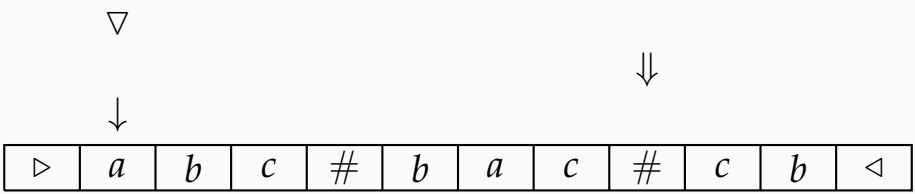
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



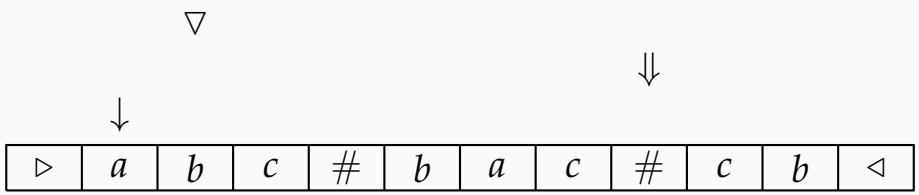
Output: abc

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



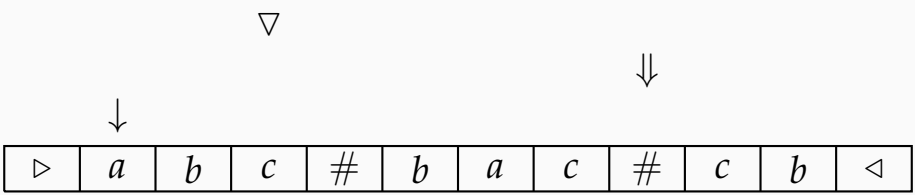
Output: *abca*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



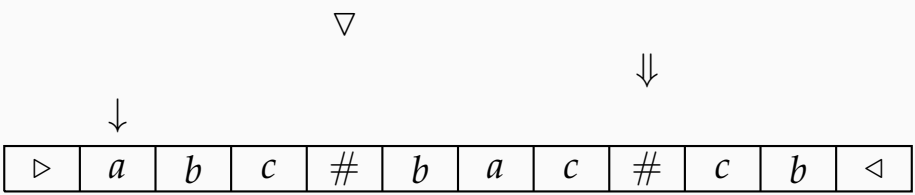
Output: *abcab*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



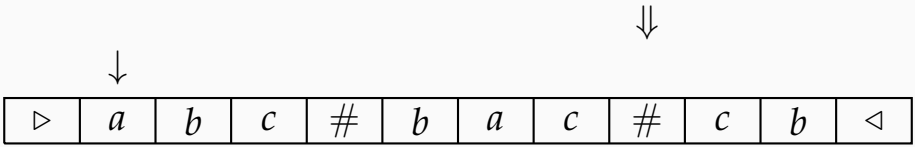
Output: *abcabc*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



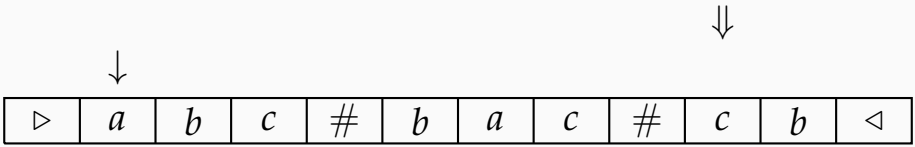
Output: *abcabc*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



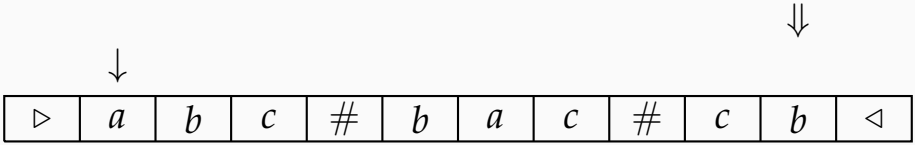
Output: *abcabc*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



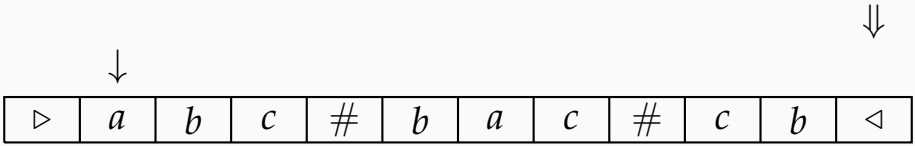
Output: *abcabc*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



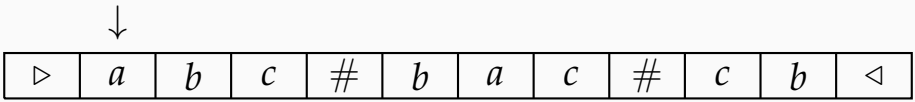
Output: *abcabc*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



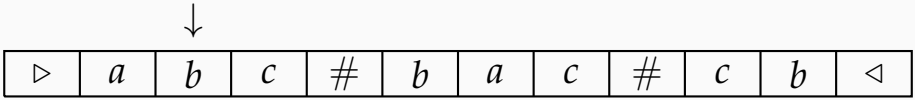
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\# \dots \#w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



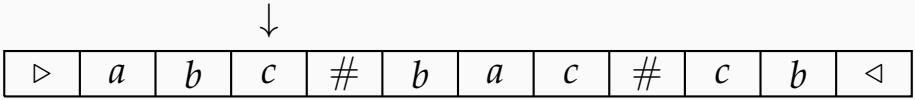
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



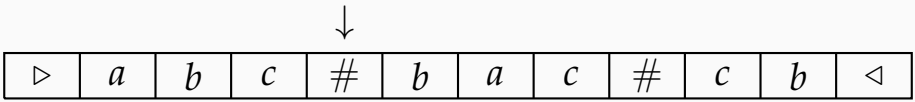
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n \# \dots \# (w_n)^n$



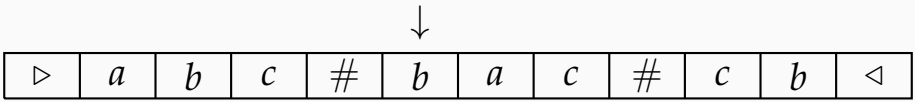
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



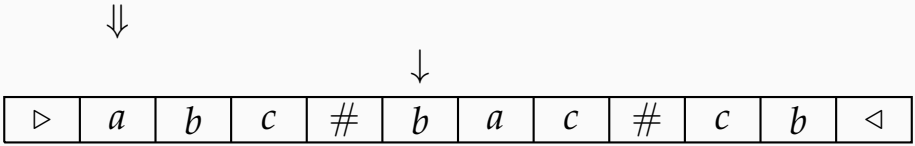
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



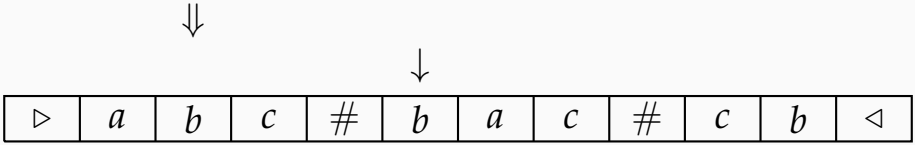
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



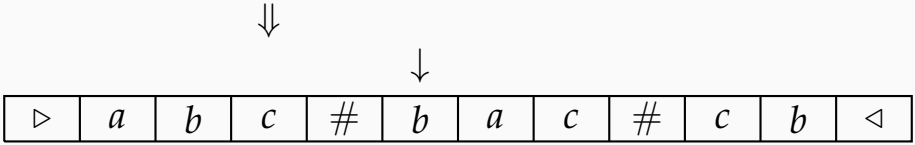
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



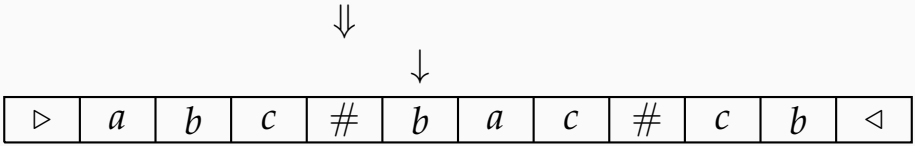
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



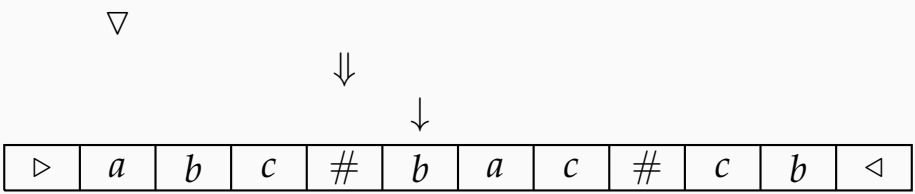
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



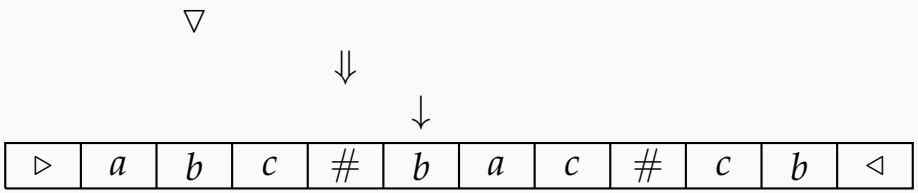
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



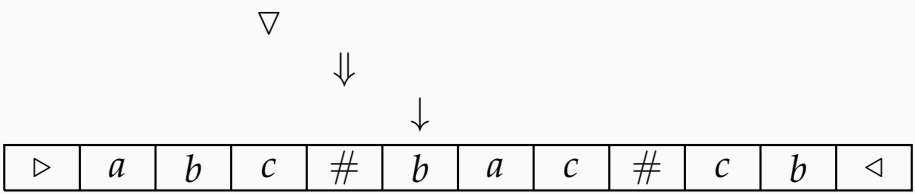
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



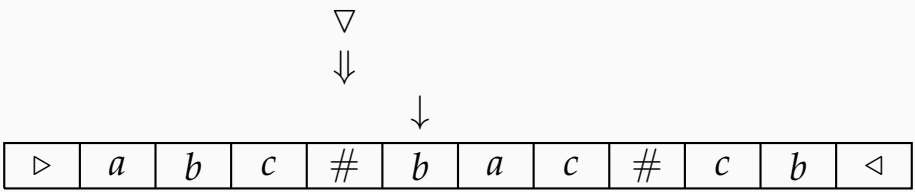
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



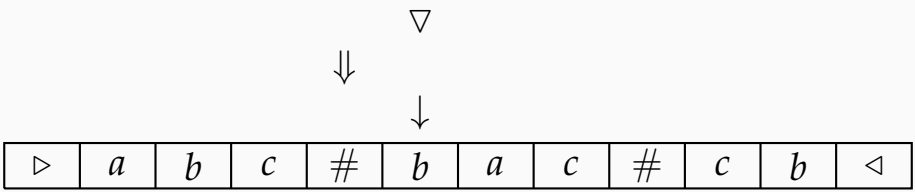
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



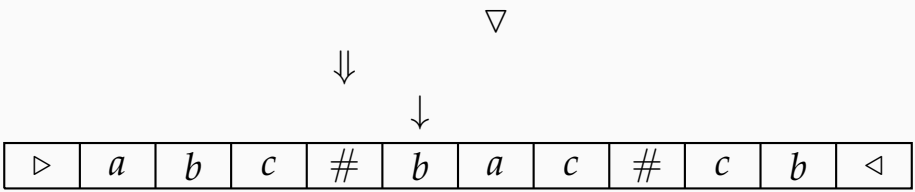
Output: $abcabc\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



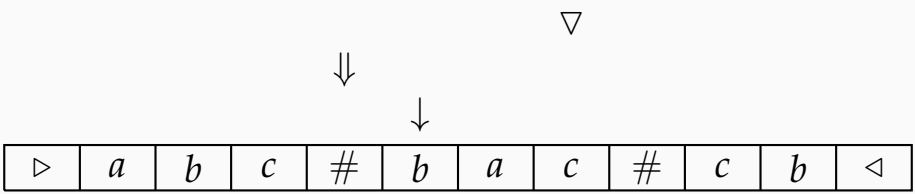
Output: $abcabc\#b$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



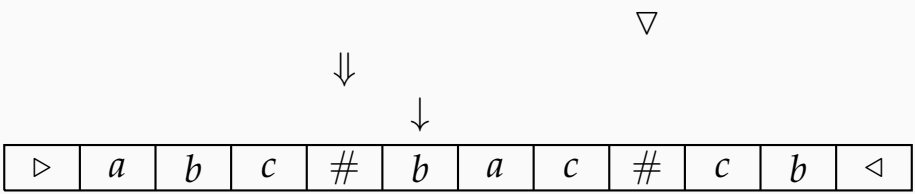
Output: *abcabc#ba*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



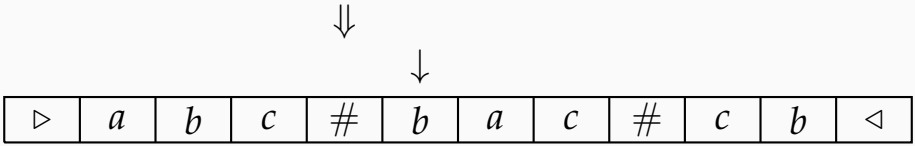
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



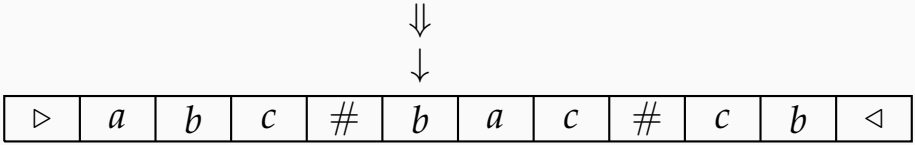
Output: *abcabc#bac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



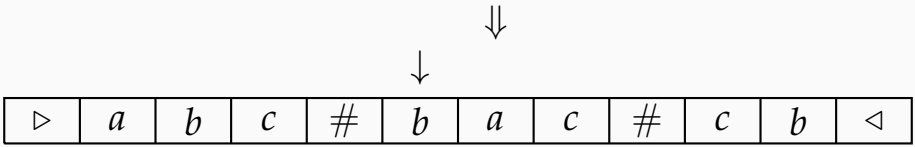
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



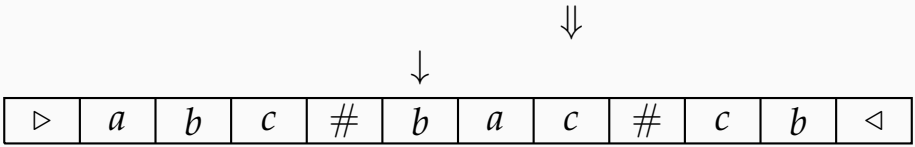
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



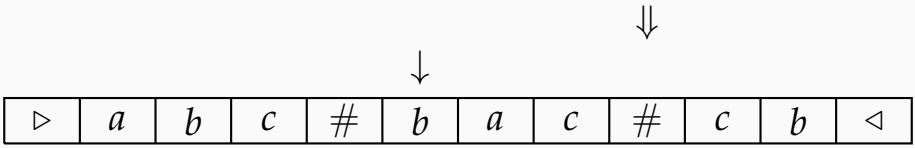
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



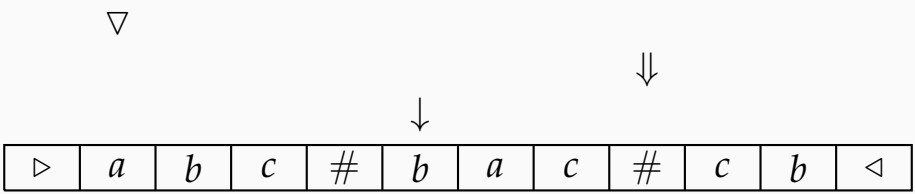
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



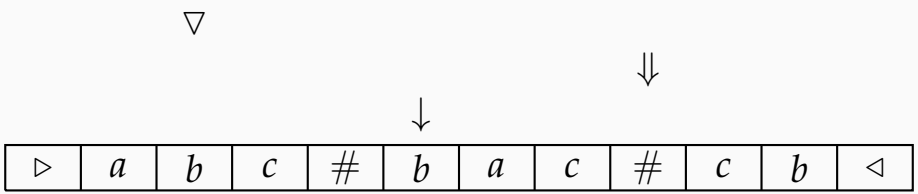
Output: $abcabc\#bac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



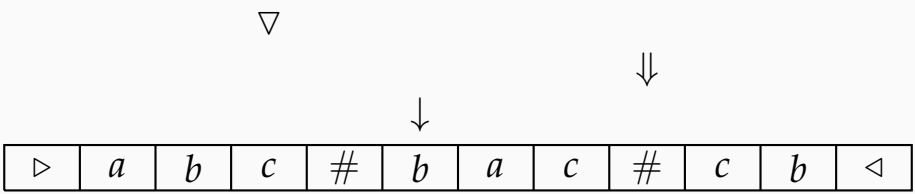
Output: *abcabc#bac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



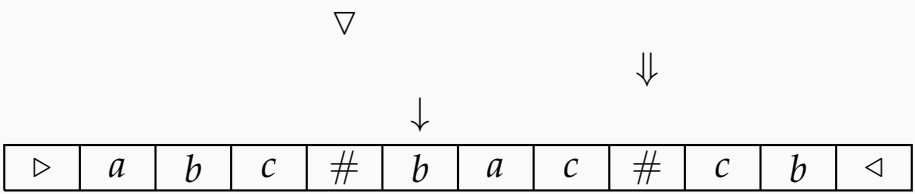
Output: *abcabc#bac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



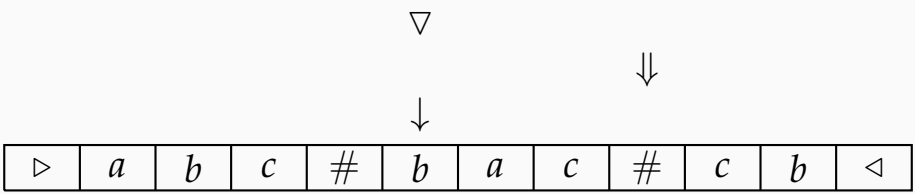
Output: *abcabc#bac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



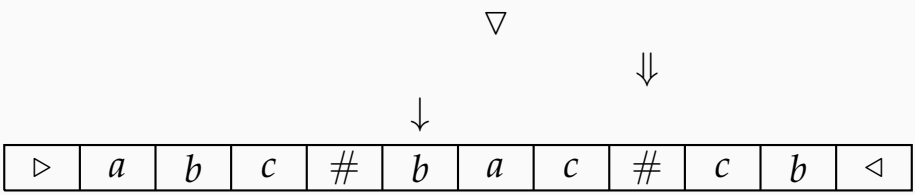
Output: *abcabc#bac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



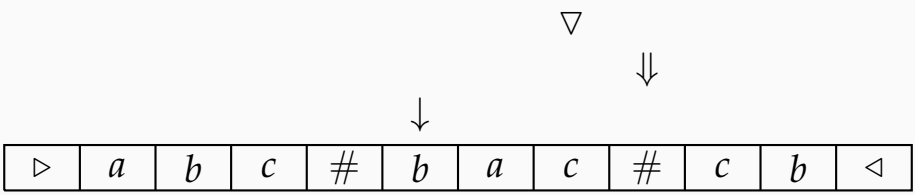
Output: *abcabc#bacb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



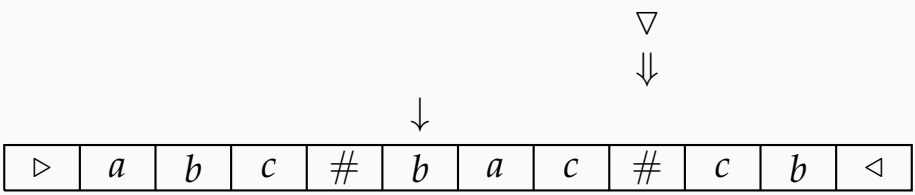
Output: *abcabc#bacba*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



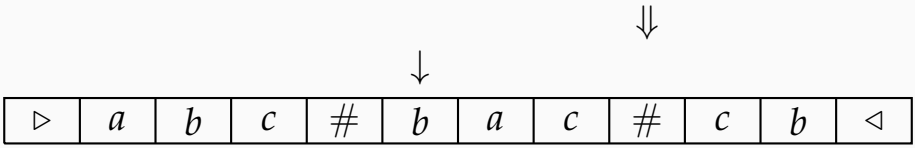
Output: $abcabc\#bacbac$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



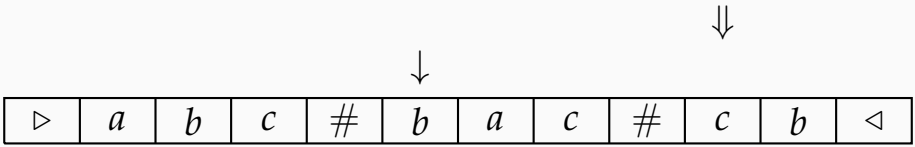
Output: *abcabc#bacbac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



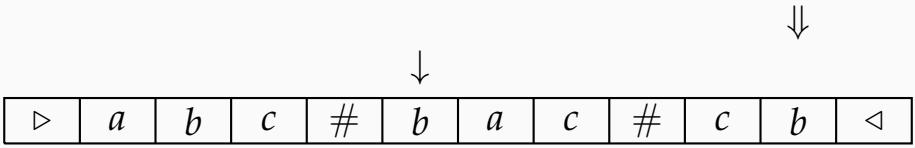
Output: *abcabc#bacbac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



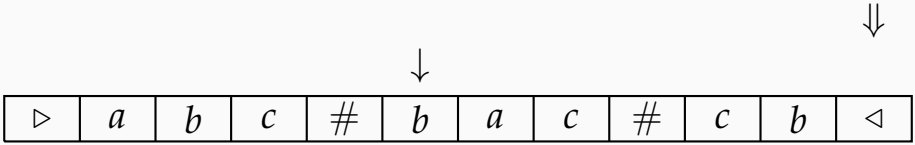
Output: *abcabc#bacbac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



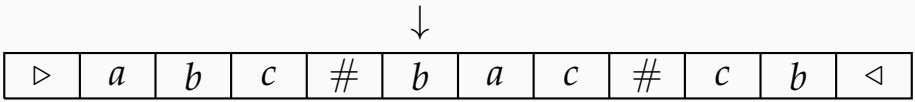
Output: *abcabc#bacbac*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



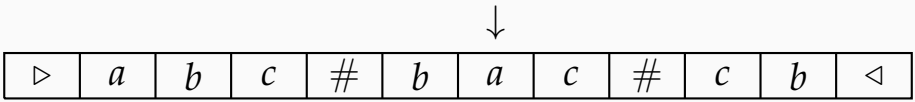
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\# \dots \#w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



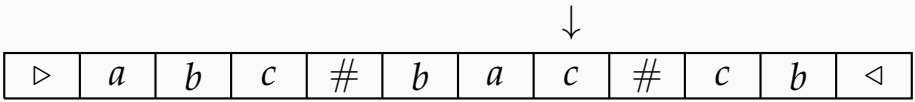
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



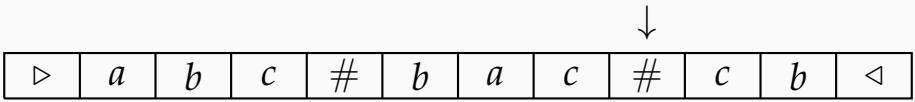
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



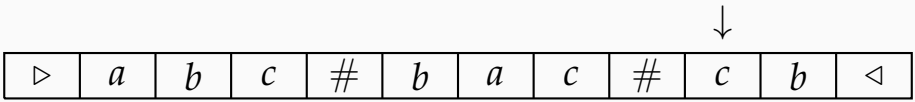
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



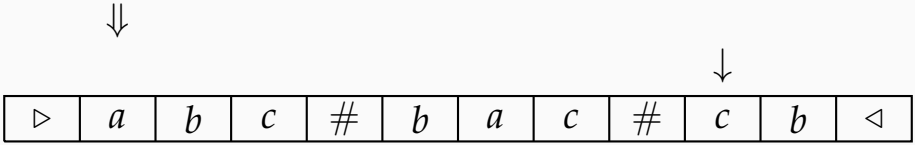
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



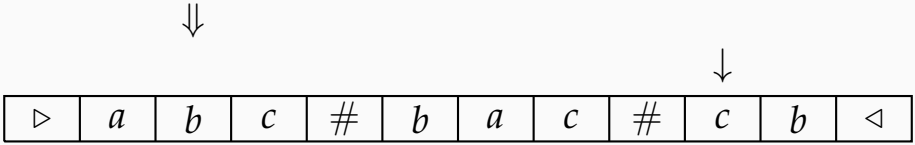
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



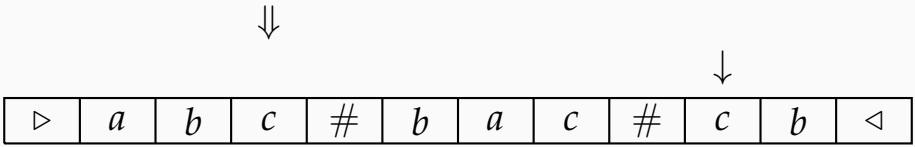
Output: *abcabc#bacbac#*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\# \ldots \#w_n \longmapsto (w_0)^n\# \ldots \#(w_n)^n$



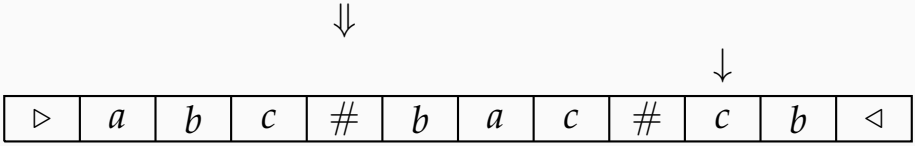
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



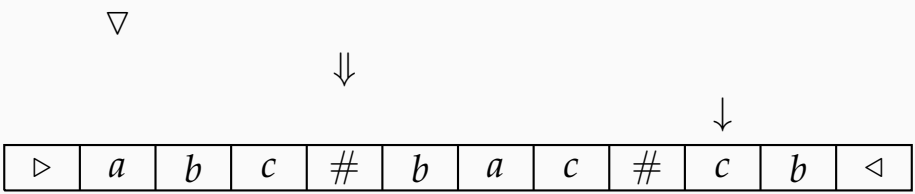
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



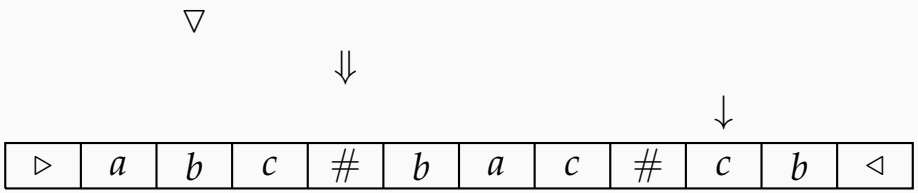
Output: *abcabc#bacbac#*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



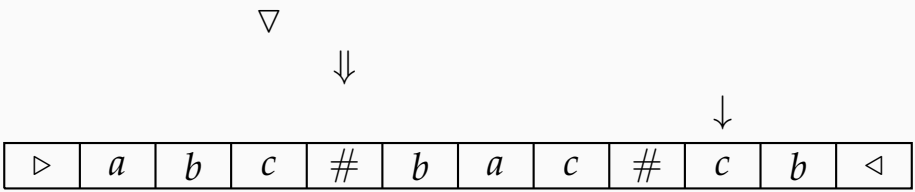
Output: *abcabc#bacbac#*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



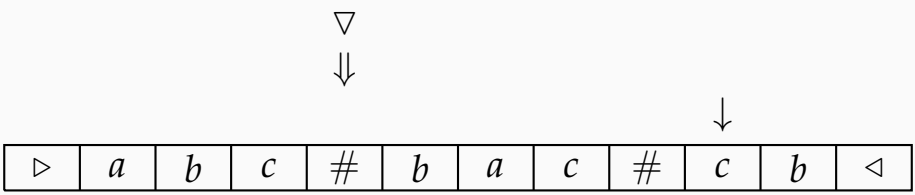
Output: *abcabc#bacbac#*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n\# \dots \#(w_n)^n$



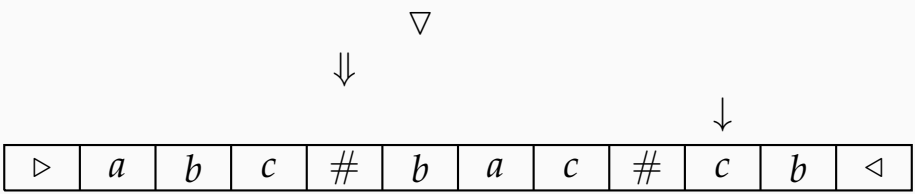
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\# \dots \# w_n \longmapsto (w_0)^n \# \dots \# (w_n)^n$



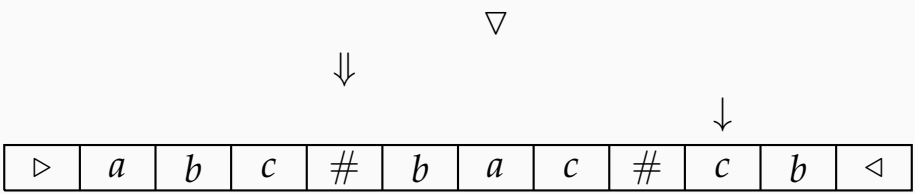
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



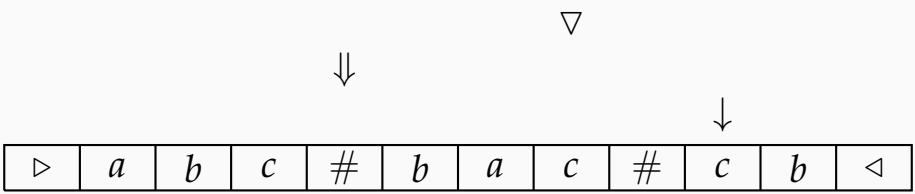
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



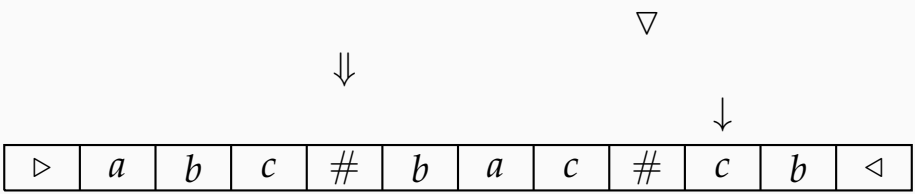
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



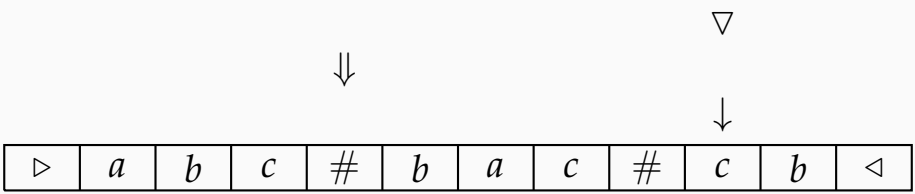
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



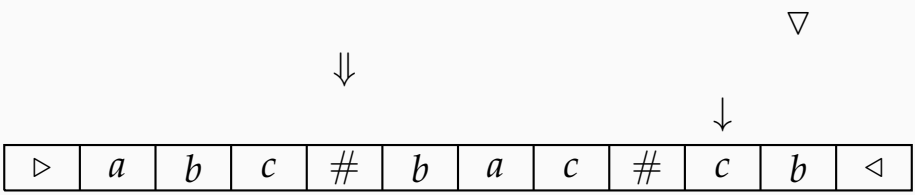
Output: $abcabc\#bacbac\#$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



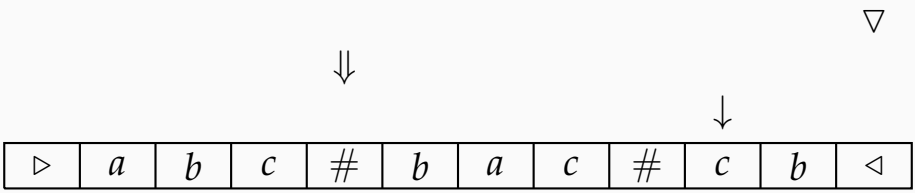
Output: $abcabc\#bacbac\#c$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



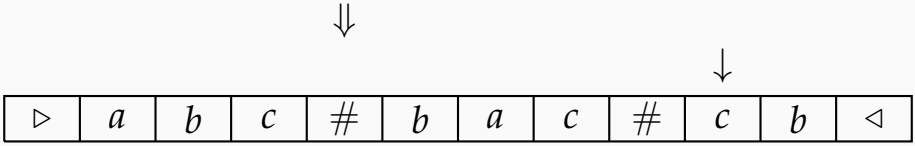
Output: *abcabc#bacbac#cb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



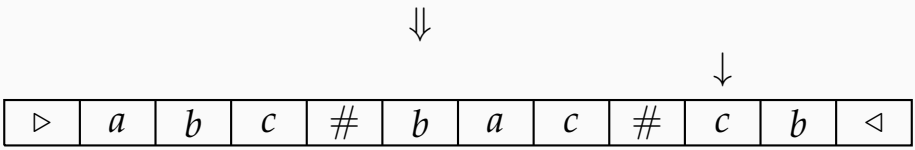
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



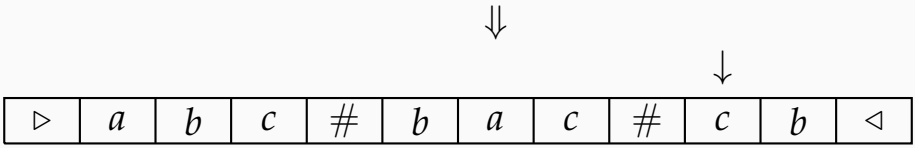
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



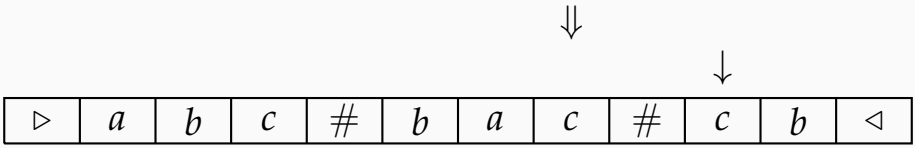
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



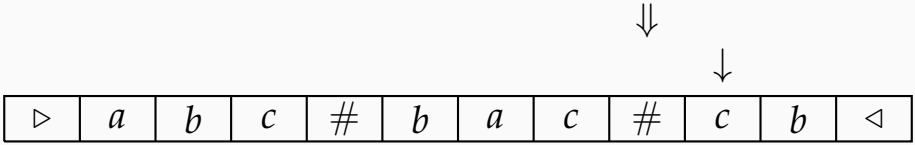
Output: *abcabc#bacbac#cb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



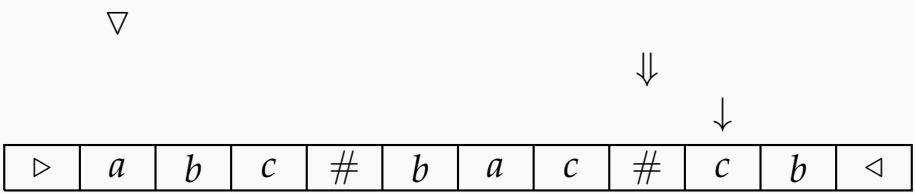
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



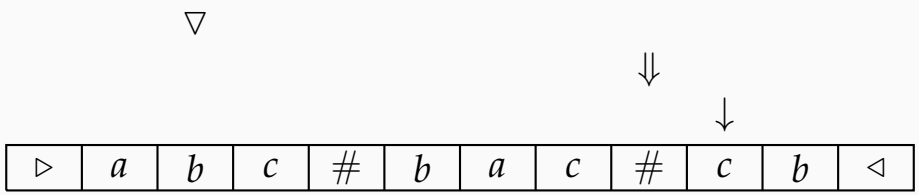
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



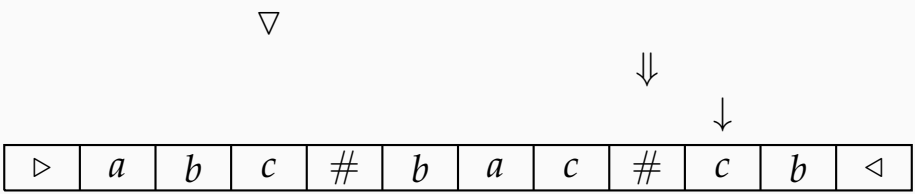
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



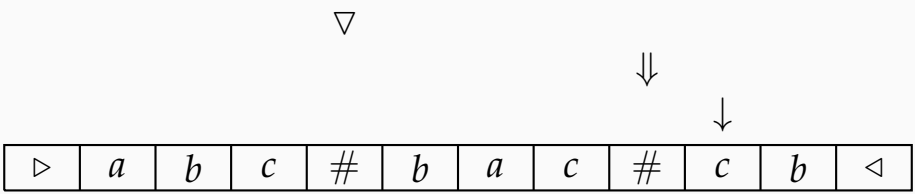
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



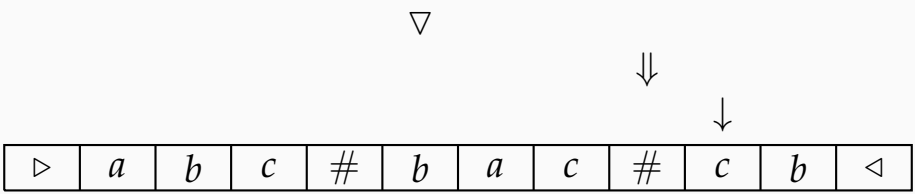
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



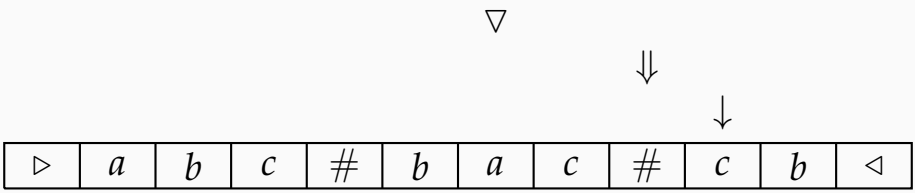
Output: *abcabc#bacbac#cb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



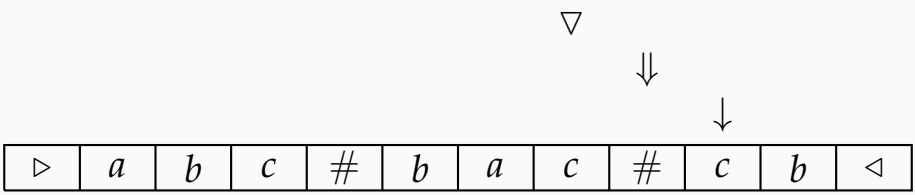
Output: *abcabc#bacbac#cb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



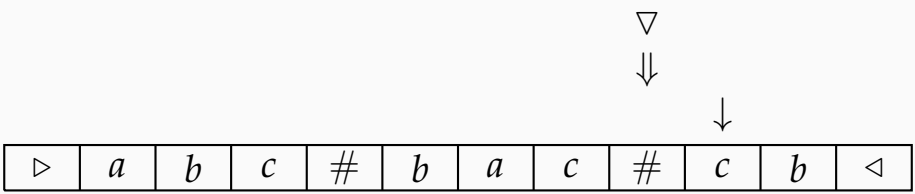
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



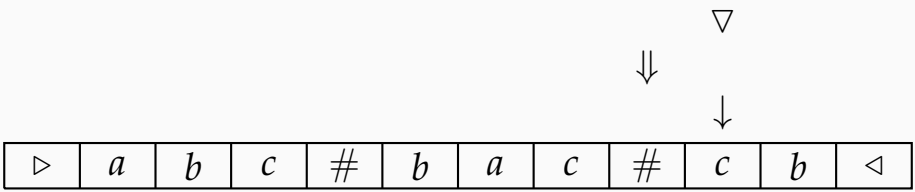
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



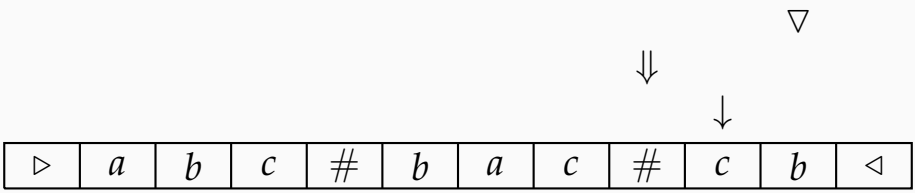
Output: $abcabc\#bacbac\#cb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



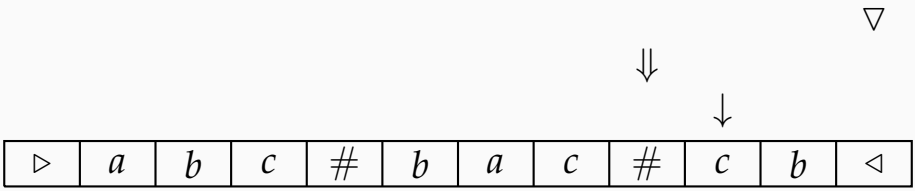
Output: $abcabc\#bacbac\#cbc$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



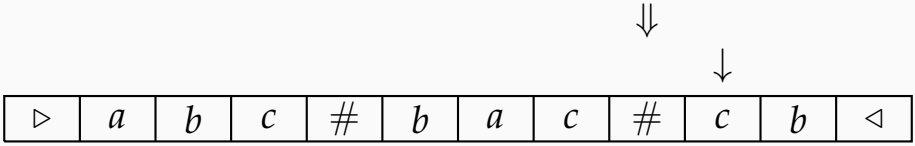
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



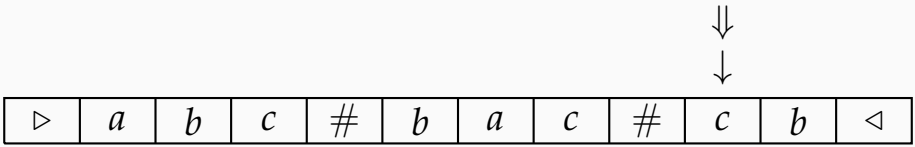
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



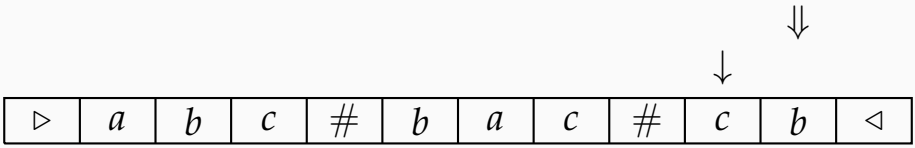
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



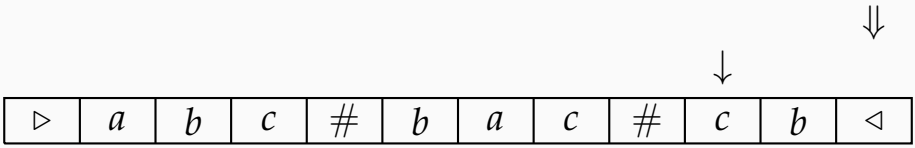
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



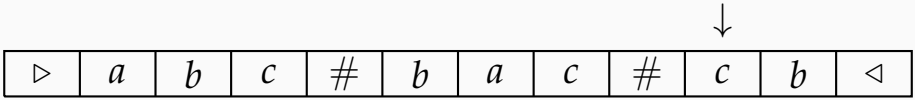
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



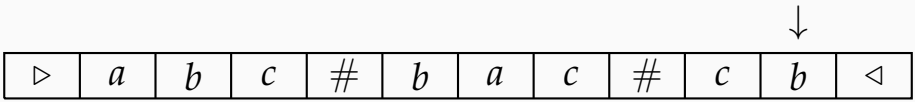
Output: *abcabc#bacbac#cbcb*

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



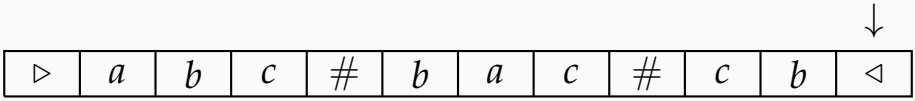
Output: $abcabc\#bacbac\#cbcb$

Pebble transducers [Milo, Suciu & Vianu 2000; Engelfriet & Maneth 2002]

Polyregular functions = computed by k-pebble transducers ($k \geq 1$)

Finite states + *stack* of height $\leqslant k$ of two-way heads ("pebbles")

"Inner squaring" $\quad \texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



Output: $abcabc\#bacbac\#cbcb$

Remarks on pebble transducers

"Inner squaring" `innsq` : $w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Our transducer checks whether the 1st and 3rd pebble have the same position
  $\longrightarrow$ are these comparisons necessary to compute `innsq` ?

Remarks on pebble transducers

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Our transducer checks whether the 1st and 3rd pebble have the same position
  $\longrightarrow$ are these comparisons necessary to compute `innsq` ?
- $|\texttt{innsq}(w)| = O(|w|^2)$, yet with 3 pebbles one can grow as fast as $|w|^3$
  $\longrightarrow$ could `innsq` be computed with only 2 pebbles?

Remarks on pebble transducers

"Inner squaring" $\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Our transducer checks whether the 1st and 3rd pebble have the same position
  $\longrightarrow$ are these comparisons necessary to compute $\texttt{innsq}$?
- $|\texttt{innsq}(w)| = O(|w|^2)$, yet with 3 pebbles one can grow as fast as $|w|^3$
  $\longrightarrow$ could $\texttt{innsq}$ be computed with only 2 pebbles?

Contributions presented in this talk

- *Polyblind* functions [N., Camille Noûs (fictional coauthor) & Cécilia Pradic, ICALP'21]
  = computed by "blind" transducers that don't compare their pebbles
  original name: *comparison-free polyregular* functions
- *Growth* of polyregular functions vs "resource usage"
  preprints: [Bojańczyk 2022] / [Kiefer, N. & Pradic 2023]

Polyblind $\subsetneq$ polyregular: comparisons are necessary

Theorem: separation results

The following functions are polyregular, but not polyblind:

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [N., Noûs & Pradic, ICALP'21]
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\#\ldots\#a^{n_k\times n_k}$ ibid.
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\ldots a^{n_k\times n_k}$ [Douéneau-Tabot MFCS'21]
- `innsq` $\to$ new; related to questions around growth rate

Polyblind $\subsetneq$ polyregular: comparisons are necessary

Theorem: separation results

The following functions are polyregular, but not polyblind:

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [N., Noûs & Pradic, ICALP'21]
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\#\ldots\#a^{n_k\times n_k}$ ibid.
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\ldots a^{n_k\times n_k}$ [Douéneau-Tabot MFCS'21]
- `innsq` $\to$ new; related to questions around growth rate

$\longrightarrow$ A new function class, defined rather naturally. Example:

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

Polyblind $\subsetneq$ polyregular: comparisons are necessary

Theorem: separation results

The following functions are polyregular, but not polyblind:

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [N., Noûs & Pradic, ICALP'21]
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\#\ldots\#a^{n_k\times n_k}$ ibid.
- $a^{n_1}\#\ldots\#a^{n_k} \mapsto a^{n_1\times n_1}\ldots a^{n_k\times n_k}$ [Douéneau-Tabot MFCS'21]
- `innsq` $\to$ new; related to questions around growth rate

$\longrightarrow$ A new function class, defined rather naturally. Example:

$$\texttt{squaring}(abb) = \underline{a}abb\underline{b}abbb\underline{a}bb$$

But is it robust/canonical? Or well-behaved?

Polyblind functions and composition

Theorem: closure under composition

If f and g are polyblind, then so is $g \circ f$.

Polyblind functions and composition

Theorem: closure under composition

If f and g are polyblind, then so is $g \circ f$.

But wait, there's more:

Alternative definition of polyblind functions

They form the smallest class closed under composition, containing the regular functions and `squaring` (reminder: $\texttt{squaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$).

Polyblind functions and composition

Theorem: closure under composition

If f and g are polyblind, then so is $g \circ f$.

But wait, there's more:

Alternative definition of polyblind functions

They form the smallest class closed under composition, containing the regular functions and `squaring` (reminder: $\texttt{squaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$).

Analogous to a characterization of polyregular functions where `squaring` is replaced by $abb \mapsto \underline{a}bba\underline{b}bab\underline{b}$ [Bojańczyk 2018]

Digression: polyblind functions in a linear λ-calculus

The actual initial motivation to introduce polyblind functions:

- Joint work with Cécilia Pradic during my PhD:
  characterizing regular functions in a λ-calculus with a linear type system
  (linear logic [Girard 1987] $\simeq$ “copyless” / “single use”)

Digression: polyblind functions in a linear λ-calculus

The actual initial motivation to introduce polyblind functions:

- Joint work with Cécilia Pradic during my PhD:
  characterizing regular functions in a λ-calculus with a linear type system
  (linear logic [Girard 1987] $\simeq$ "copyless" / "single use")
- By relaxing one of the linearity conditions, we stumbled upon a class of functions that did not already exist...

Digression: polyblind functions in a linear λ-calculus

The actual initial motivation to introduce polyblind functions:

- Joint work with Cécilia Pradic during my PhD:
  characterizing regular functions in a λ-calculus with a linear type system
  (linear logic [Girard 1987] $\simeq$ "copyless" / "single use")
- By relaxing one of the linearity conditions, we stumbled upon a class of functions that did not already exist...
- Discovery (also in my PhD): this corresponds to blind pebble transducers!
  $\longrightarrow$ additional argument for the canonicity of these functions

Digression: polyblind functions in a linear λ-calculus

The actual initial motivation to introduce polyblind functions:

- Joint work with Cécilia Pradic during my PhD:
  characterizing regular functions in a λ-calculus with a linear type system
  (linear logic [Girard 1987] $\simeq$ "copyless" / "single use")
- By relaxing one of the linearity conditions, we stumbled upon a class of functions that did not already exist...
- Discovery (also in my PhD): this corresponds to blind pebble transducers!
  $\longrightarrow$ additional argument for the canonicity of these functions

Remark: closure under composition is *automatic* using the λ-calculus definition
(hidden structure of monoidal closed category in the direct composition proof)

Growth rate vs number of pebbles

Theorem: blind pebble minimisation **(most technical result in the ICALP'21 paper)**

$f : \Gamma^* \to \Sigma^*$ is polyblind and $|f(w)| = O(|w|^k)$

$\iff$

f is computed by a blind k-pebble transducer

- Original proof: Ramsey-based pumping (anecdote time: LICS = bad)

Growth rate vs number of pebbles

Theorem: blind pebble minimisation **(most technical result in the ICALP'21 paper)**

$f : \Gamma^* \to \Sigma^*$ is polyblind and $|f(w)| = O(|w|^k)$

$\iff$

f is computed by a blind k-pebble transducer

- Original proof: Ramsey-based pumping (anecdote time: LICS = bad)
- Effective (re)proof using factorization forests [Douéneau-Tabot FoSSaCS'23]

Growth rate vs number of pebbles

Theorem: blind pebble minimisation **(most technical result in the ICALP'21 paper)**

$f : \Gamma^* \to \Sigma^*$ is polyblind and $|f(w)| = O(|w|^k)$

$\iff$

f is computed by a blind k-pebble transducer

- Original proof: Ramsey-based pumping (anecdote time: LICS = bad)
- Effective (re)proof using factorization forests [Douéneau-Tabot FoSSaCS'23]

Theorem (Bojańczyk 2022 / Kiefer, N. & Pradic 2023)

Even though "inner squaring" has growth $O(n^2)$, it requires 3 pebbles!

That said, polyregular + $O(n)$ $\implies$ regular i.e. 1-pebble

(corollary of [Engelfriet, Inaba & Maneth 2021] on tree transducers)

Lower bounds on the number of pebbles

Theorem

$\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$ requires 3 pebbles

- elementary proof (Kiefer, N. & Pradic): use a pumping lemma on the *output languages* of regular functions ($\simeq$ k-iterativity) [Rozoy 1986 – in French!]
  sidesteps a detailed analysis of the execution of a transducer
- conceptual proof (Bojańczyk): detour through *atoms* (infinite alphabets)
  + technically challening "deatomization" theorem

Lower bounds on the number of pebbles

Theorem

$\texttt{innsq} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$ requires 3 pebbles

- elementary proof (Kiefer, N. & Pradic): use a pumping lemma on the *output languages* of regular functions ($\simeq$ k-iterativity) [Rozoy 1986 – in French!]
  sidesteps a detailed analysis of the execution of a transducer
- conceptual proof (Bojańczyk): detour through *atoms* (infinite alphabets)
  + technically challening "deatomization" theorem

The second approach – but not the first – extends to:

Theorem

For any $k \geq 2$ there is a polyregular f with $|f(w)| = O(|w|^2)$ that needs k pebbles.

Pebble non-minimization for output languages

Theorem (proved by Mikołaj Bojańczyk via "deatomization")

For any $k \geq 2$ there is a polyregular f with $|f(w)| = O(|w|^2)$ that needs k pebbles.

We strengthen this to:

Theorem (Sandra Kiefer, N. & Cécilia Pradic)

For any $k \geq 1$ there is a polyregular f with $|f(w)| = O(|w|^2)$ whose *output language* differs from that of any $\underbrace{k\text{-fold composition of macro tree transducers}}_{\text{includes all } k\text{-pebble transducers [Engelfriet \& Maneth 2003]}}$.

Completely different argument; very similar counterexample functions...

(inspired by [Engelfriet & Maneth MFCS'02] – earliest paper on pebble *string* transducers)

Pebble non-minimization for output languages

Theorem (proved by Mikołaj Bojańczyk via "deatomization")

For any $k \geq 2$ there is a polyregular f with $|f(w)| = O(|w|^2)$ that needs k pebbles.

We strengthen this to:

Theorem (Sandra Kiefer, N. & Cécilia Pradic)

For any $k \geq 1$ there is a polyregular f with $|f(w)| = O(|w|^2)$ whose *output language* differs from that of any $\underbrace{k\text{-fold composition of macro tree transducers}}_{\text{includes all } k\text{-pebble transducers [Engelfriet \& Maneth 2003]}}$.

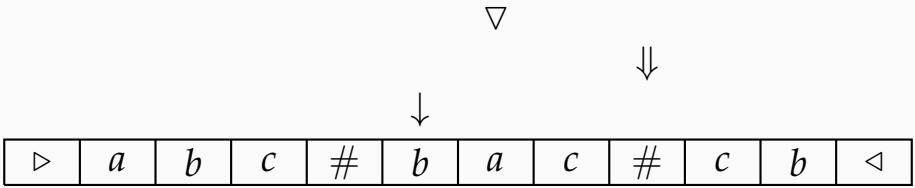
Completely different argument; very similar counterexample functions...

(inspired by [Engelfriet & Maneth MFCS'02] – earliest paper on pebble *string* transducers)

Remark: any *polyblind* function can be realized as a composition of *two* MTTs

(special case of [Engelfriet, Hoogeboom & Samwel 2021])

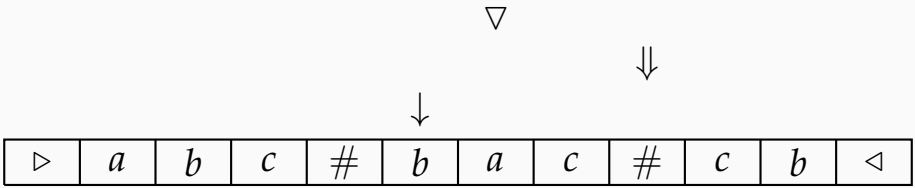
Pebble minimization: obstruction in the non-blind case



Output: $abcabc\#bacb\color{red}{a}\ldots$

- when our transducer for `innsq` outputs, *the position of* $\triangledown$ *determines that of* $\downarrow$
- redundant but *necessary*: to move $\Downarrow$, you have to *pop* $\triangledown$

Pebble minimization: obstruction in the non-blind case



Output: $abcabc\#bacba\ldots$

- when our transducer for `innsq` outputs, *the position of* $\triangledown$ *determines that of* $\downarrow$
- redundant but *necessary*: to move $\Downarrow$, you have to *pop* $\triangledown$

We'd like more flexibility to only need 2 "input pointers"

$\longrightarrow$ another formalism: MSO interpretations

Logical transductions

Example: $w \mapsto a^{|w|} \cdot \texttt{reverse}(w)$

$$\begin{array}{cc} a\ b\ a\ c & \\ 1\ 2\ 3\ 4 & \rightsquigarrow \end{array} \quad \begin{array}{cccccccc} a & a & a & a & c & a & b & a \\ \lambda 1 & \lambda 2 & \lambda 3 & \lambda 4 & \rho 4 & \rho 3 & \rho 2 & \rho 1 \end{array}$$

$I = \{\lambda, \rho\}$ $\qquad a(\lambda i) = \texttt{true}, a(\rho i) = (w[i] = a)$ $\qquad \lambda 1 \prec \lambda 2 \prec \cdots \prec \rho 2 \prec \rho 1$

Logical transductions

Example: $w \mapsto a^{|w|} \cdot \texttt{reverse}(w)$

$$\begin{array}{cc} a\ b\ a\ c & \\ 1\ 2\ 3\ 4 & \rightsquigarrow \end{array} \begin{array}{cccccccc} a & a & a & a & c & a & b & a \\ \lambda 1 & \lambda 2 & \lambda 3 & \lambda 4 & \rho 4 & \rho 3 & \rho 2 & \rho 1 \end{array}$$

$I = \{\lambda, \rho\}$ $\qquad a(\lambda i) = \texttt{true}, a(\rho i) = (w[i] = a)$ $\qquad \lambda 1 \prec \lambda 2 \prec \cdots \prec \rho 2 \prec \rho 1$

Idea: for an input word $w \in \Gamma^*$, define over $z, z' \in I \times \{1, \ldots, |w|\}$

unary relations $a(z)$ for $a \in \Sigma$ $\quad + \quad$ a binary relation $z \prec z'$

if we're lucky, the result is isomorphic to an output word $f(w) \in \Sigma^*$

MSO transductions

Not really a reminder on Monadic Second-Order logic

MSO formula: $\varphi(x_1, \ldots, x_n)$, the x_k refer to *positions* of a word w ($1 \leq x_k \leq |w|$)

- $n = 0$: $L \subseteq \Gamma^*$ is regular $\iff \exists \varphi.\ L = \{w \in \Sigma^* \mid w \vDash \varphi\}$
- $n \geq 1$: generalization to words with n marked positions

MSO transduction = finite set I + $\varphi_a^i(x)$ + $\varphi_\prec^{i,j}(x, y)$ for $a \in \Sigma$ and $i, j \in I$

Theorem [Engelfriet & Hoogeboom 2001]

String-to-string MSO transductions = regular functions (i.e. 1-pebble)

(Not too hard to prove once you know regular functions are closed under composition)

MSO interpretations in higher dimension

MSO interpretation $\Gamma^* \to \Sigma^*$ = choose $k \in \mathbb{N}$, a finite set I & MSO formulas

$$\varphi_a^i(x_1, \ldots, x_k) \text{ for } a \in \Sigma \qquad \varphi_\prec^{i,j}(x_1, \ldots, x_k, y_1, \ldots, y_k) \qquad i, j \in I$$

$w \in \Gamma^* \longmapsto$ relations $a(-)$ and $\prec$ over $I \times \{1, \ldots, |w|\}^k$

again, if we're lucky, this structure is isomorphic to some $f(w) \in \Sigma^*$

MSO interpretations in higher dimension

MSO interpretation $\Gamma^* \to \Sigma^*$ = choose $k \in \mathbb{N}$, a finite set I & MSO formulas

$$\varphi_a^i(x_1, \dots, x_k) \text{ for } a \in \Sigma \qquad \varphi_\prec^{i,j}(x_1, \dots, x_k, y_1, \dots, y_k) \qquad i, j \in I$$

$w \in \Gamma^* \longmapsto$ relations $a(-)$ and $\prec$ over $I \times \{1, \dots, |w|\}^k$

again, if we're lucky, this structure is isomorphic to some $f(w) \in \Sigma^*$

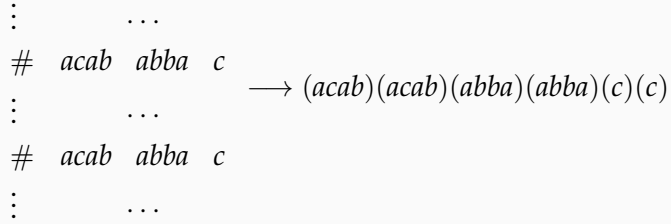
Theorem [Bojańczyk, Kiefer & Lhote 2019]

String-to-string MSO interpretations = polyregular functions

- Highly technical proof using finite model theory
- Somewhat "unnatural": no reason *a priori* for MSO interpretations to preserve regular languages by inverse image

MSO interpretation example

$\texttt{innsq}' : w_0\#\ldots\#w_n \longmapsto (w_0)^n\ldots(w_n)^n$ has a dim. 2 (optimal) interpretation:



MSO interpretation example

$\texttt{innsq}' : w_0\#\ldots\#w_n \longmapsto (w_0)^n\ldots(w_n)^n$ has a dim. 2 (optimal) interpretation:

$$\begin{array}{cccc} & acab\#abba\#c & & \\ \vdots & \cdots & & \\ \# & acab & abba & c \\ \vdots & \cdots & & \\ \# & acab & abba & c \\ \vdots & \cdots & & \end{array} \longrightarrow (acab)(acab)(abba)(abba)(c)(c)$$

- $\varphi_a(x_1, x_2) = a(x_1) \wedge \#(x_2)$ (omitting copy indices: $|I| = 1$)
- $\varphi_{\prec}(x_1, x_2, y_1, y_2) = \exists x_3, y_3.$ which begin blocks containing resp. x_1, y_1 and $(x_3, x_2, x_1) < (y_3, y_2, y_1)$ lex. $\longrightarrow$ pebbles $\downarrow, \Downarrow, \triangledown$

Dimension minimisation

Theorem

MSO interpretations *of dim. k* on strings = polyregular fn *with growth* $O(n^k)$

(Corollary: new proof of polyregular + $O(n) \implies$ regular)

Dimension minimisation

Theorem

MSO interpretations *of dim. k* on strings = polyregular fn *with growth* $O(n^k)$

(Corollary: new proof of polyregular + $O(n) \implies$ regular)

Fundamentally, it's not about interpretations, it's about *queries*:

Main lemma (Bojańczyk): reflects the dependencies between positions

Every MSO formula $\varphi(x_1, \dots, x_n)$ is equivalent to $\bigvee_{i \in I} \varphi_i$ (I finite) where $\forall i \in I$,

- $\exists Y_i \subseteq \{1, \dots, n\}$ such that for every marked word, when φ_i holds, the values of x_m for $m \in Y_i$ determine all others
- moreover $\max\{\text{nb of tuples of pos. in } w \text{ satisfying } \varphi_i \mid |w| = n\} = \Theta(n^{|Y_i|})$

Tools in the proof: compositionality of MSO + factorisation forests

An easier proof of dimension minimization

Nathan Lhote's idea: Bojańczyk's lemma looks like some decomposition theorems on *polynomially ambiguous automata* e.g. [Kreutzer & Riveros LICS'13; Paul DLT'16]
$(*)$

Sandra, Cécilia and me — no factorization forests / Green's relations / etc. needed

$\rightsquigarrow$ we invoke a similar simple property of $\mathbb{N}$-*weighted automata*:

An easier proof of dimension minimization

Nathan Lhote's idea: Bojańczyk's lemma looks like some decomposition theorems on *polynomially ambiguous automata* e.g. [Kreutzer & Riveros LICS'13; Paul DLT'16] $(*)$

Sandra, Cécilia and me — no factorization forests / Green's relations / etc. needed

$\rightsquigarrow$ we invoke a similar simple property of $\mathbb{N}$-*weighted automata*:

Lemma (folklore?) **this version is from [Douéneau-Tabot, Filiot & Gastin MFCS'20]**

If $\mathcal{A} = \Big[\mu \in \mathrm{Hom}(\Gamma^*, \underbrace{\mathbb{N}^{Q\times Q}}_{\text{multiplicative monoid of matrices}}) + \text{initial/final vectors } \mathbb{N}^Q\Big]$ is a *trim* $\mathbb{N}$-automaton,

automata POV: transitions go from Q_i to Q_j with $i \leq j$

and $k = \inf\{\ell \mid \mathcal{A}(w) = O(|w|^\ell)\} < +\infty$, then $\exists$ a block triangular structure $Q = Q_0 \sqcup \cdots \sqcup Q_k$ on $\mu(\Gamma^*)$ such that the diagonal blocks are *bounded*.

$\Longrightarrow$ quickly deduce a slight weakening of the lemma on MSO queries!

(advantage: should work on *trees* by adapting $(*)$)

Conclusion

(Poly)regular functions: transduction classes with many characterizations

(automata, logic, composition of basic functions)

Developments presented in this talk

- Polyblind (a.k.a. comparison-free polyregular) functions
- *Growth* of polyregular functions vs "resource usage"

Conclusion

(Poly)regular functions: transduction classes with many characterizations

(automata, logic, composition of basic functions)

Developments presented in this talk

- Polyblind (a.k.a. comparison-free polyregular) functions
- *Growth* of polyregular functions vs "resource usage"

- Pebble min. works in special cases: unary outputs [Douéneau-Tabot MFCS'21], unary inputs [Kiefer, N. & Pradic 2023], restricted comparisons [D-T. FoSSaCS'23]

"generalizing" the blind case that we proved

Conclusion (Thanks for your attention!)

(Poly)regular functions: transduction classes with many characterizations

(automata, logic, composition of basic functions)

Developments presented in this talk

- Polyblind (a.k.a. comparison-free polyregular) functions
- *Growth* of polyregular functions vs "resource usage"

- Pebble min. works in special cases: unary outputs [Douéneau-Tabot MFCS'21], unary inputs [Kiefer, N. & Pradic 2023], restricted comparisons [D-T. FoSSaCS'23]

  "generalizing" the blind case that we proved
- Logic for polyblind functions? `innsq` refutes a naive attempt
- Aperiodicity (note: $n(n+1)/2$ polyblind and FO-polyregular but not FO-polyblind!)
- Decision problems: equivalence, membership… e.g. [D-T. ICALP'22]

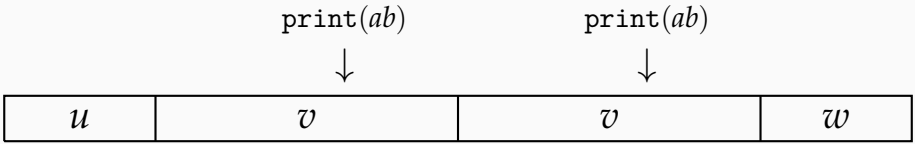
Backup slides

Pebble minimization : vague idea of the nullary case [Lhote 2020]

Consider a two-way transducer computing $f : \Gamma^* \to \Sigma^*$.

We are looking to "pump" triplets $(u, v, w) \in (\Gamma^*)^3$ such that:
"$\forall n, m \in \mathbb{N}$ the transducer has the same behavior in the factor v of $(uv^m)v(v^nw)$"

to find them: finite monoid of behaviors + Ramsey's theorem

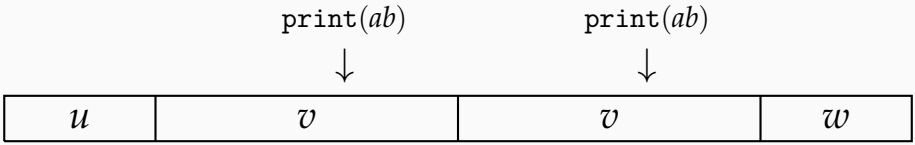


Pebble minimization : vague idea of the nullary case [Lhote 2020]

Consider a two-way transducer computing $f : \Gamma^* \to \Sigma^*$.

We are looking to "pump" triplets $(u, v, w) \in (\Gamma^*)^3$ such that:
"$\forall n, m \in \mathbb{N}$ the transducer has the same behavior in the factor v of $(uv^m)v(v^n w)$"

to find them: finite monoid of behaviors + Ramsey's theorem

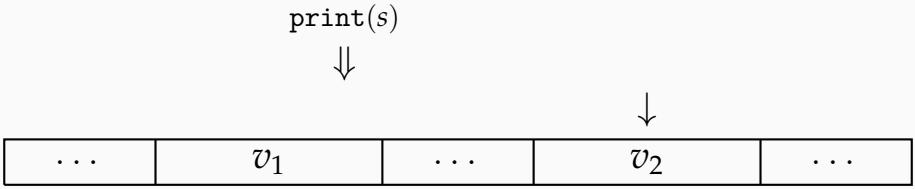


- above: $|f(uv^n w)| \geq n|ab| = 2n$
- if no triplet is "producing": $|f(w)| = O(1)$

Pebble minimisation : vague idea (continued)

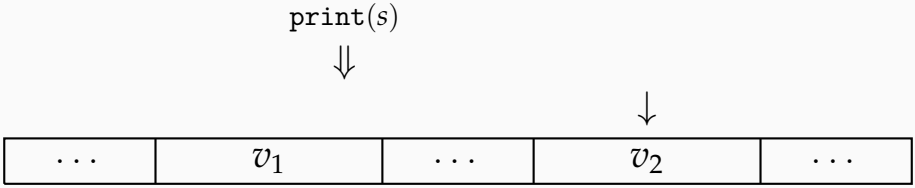
Consider a 2-pebble transducer that computes $f : \Gamma^* \to \Sigma^*$.

We would like to pump patterns (with contextual behavior condition?):



Pebble minimisation : vague idea (continued)

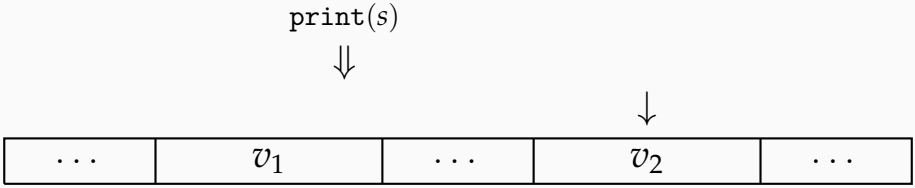
Consider a 2-pebble transducer that computes $f : \Gamma^* \to \Sigma^*$.
We would like to pump patterns (with contextual behavior condition?):



- above: $s \neq \varepsilon \implies |f(\dots(v_1)^n \dots (v_2)^m \dots)| \geq n \times m$
- if no such producing 5-tuple: simplifiable to 1-pebble??

Pebble minimisation : vague idea (continued)

Consider a 2-pebble transducer that computes $f : \Gamma^* \to \Sigma^*$.
We would like to pump patterns (with contextual behavior condition?):



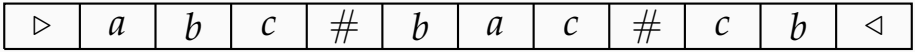
- above: $s \neq \varepsilon \implies |f(\dots(v_1)^n \dots (v_2)^m \dots)| \geq n \times m$
- if no such producing 5-tuple: simplifiable to 1-pebble??

Works (non-trivially) for blind pebble transducers

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

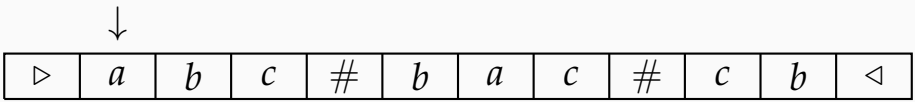


Output:

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

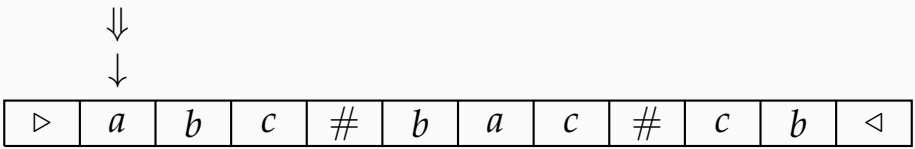


Output:

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

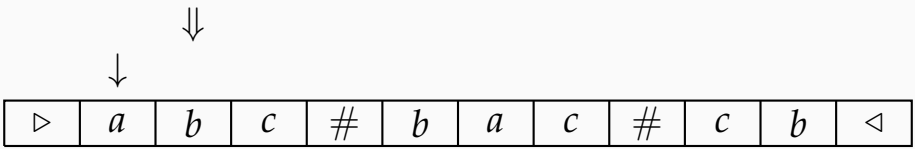


Output:

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

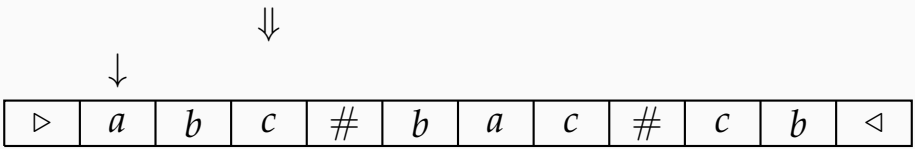


Output: a

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

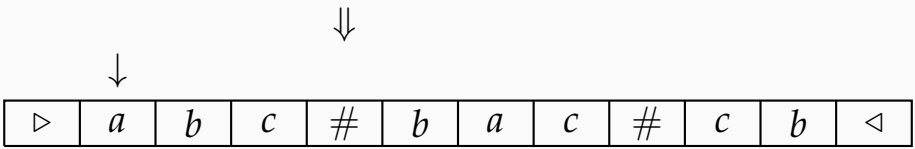


Output: ab

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

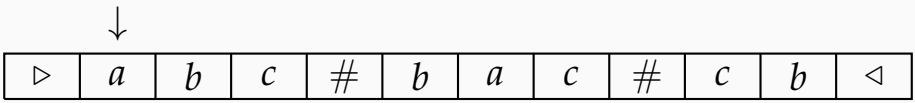


Output: *abc*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

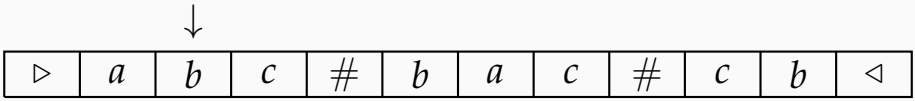


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

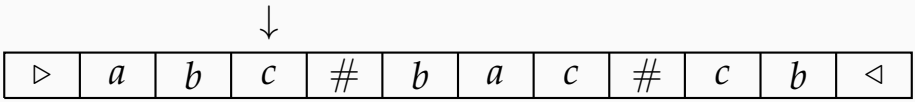


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

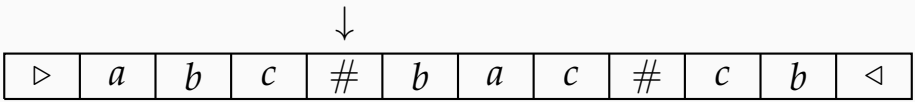


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

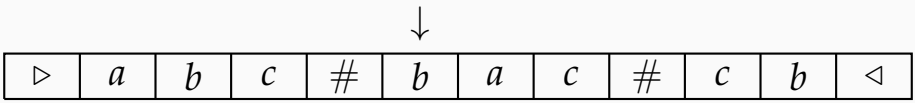


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

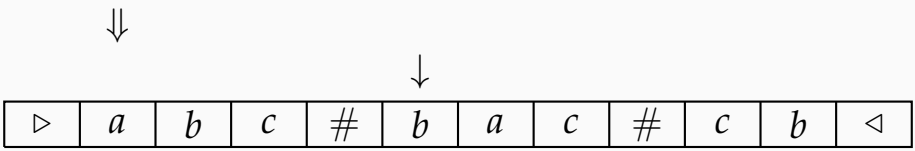


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

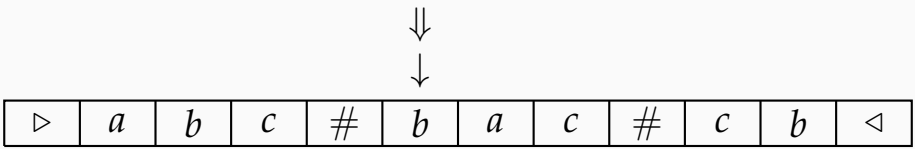


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

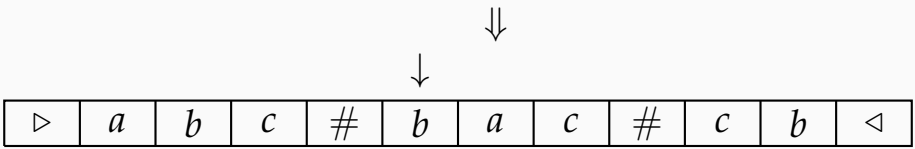


Output: $abc\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

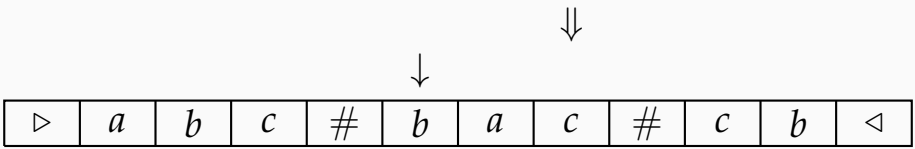


Output: $abc\#b$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

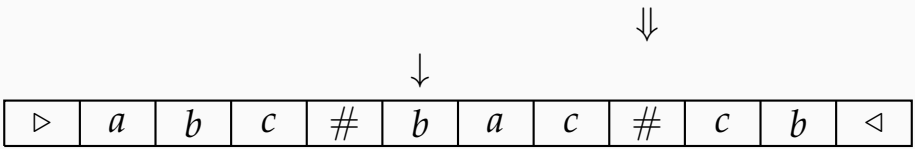


Output: $abc\#ba$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

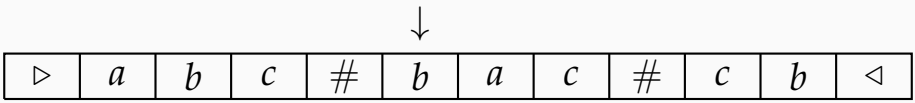


Output: *abc#bac*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

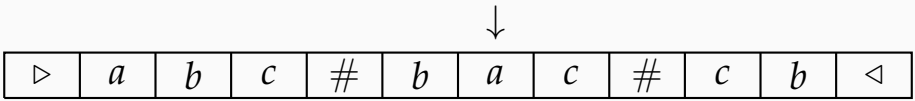


Output: $abc\#bac\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

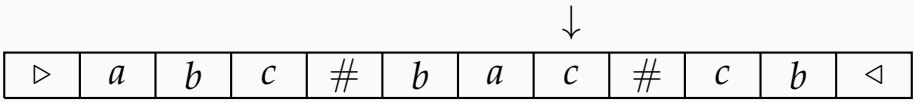


Output: $abc\#bac\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

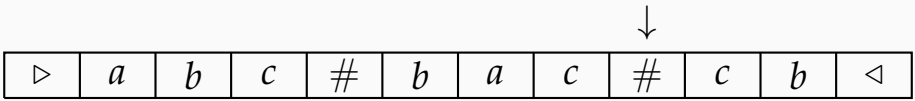


Output: *abc#bac#*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

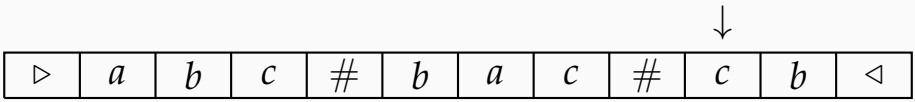


Output: $abc\#bac\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

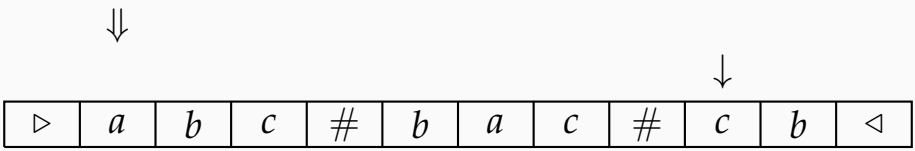


Output: *abc#bac#*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

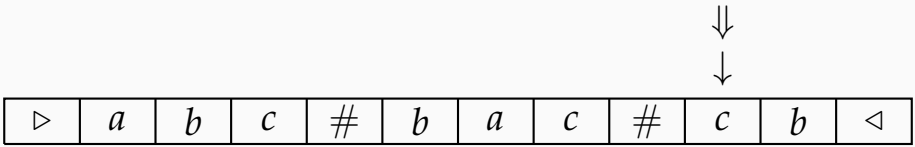


Output: $abc\#bac\#$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

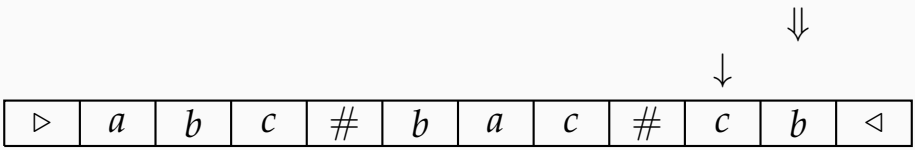


Output: *abc#bac#*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

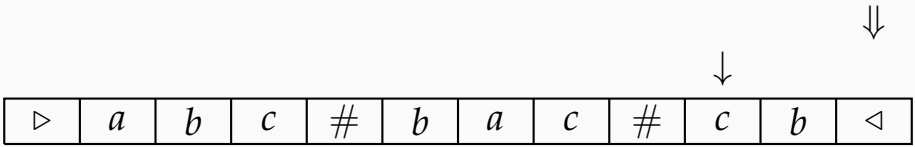


Output: $abc\#bac\#c$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

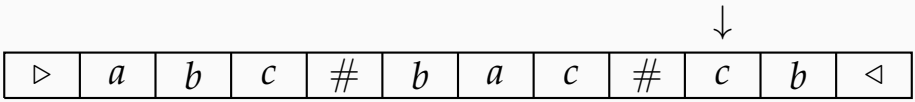


Output: *abc#bac#cb*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

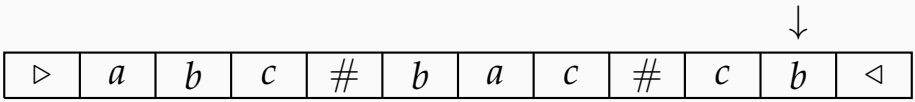


Output: *abc#bac#cb*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

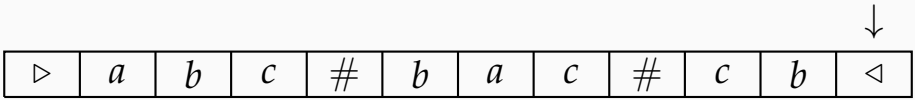


Output: *abc#bac#cb*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)

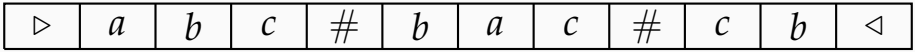


Output: *abc#bac#cb*

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)



Output: $abc\#bac\#cb$

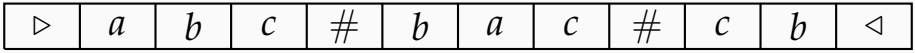
Pebble comparisons $\implies$ potential *dependencies* between their positions

$$\ldots(\#aa)(\#aa)(\#aa)(\#aa)(bb)(bb)(bb)\ldots$$

Pebble minimization: obstruction in the non-blind case (1)

Identity function computed block-wise with 2 pebbles

(basically `innsq` without the 2nd pebble that counts the $(-)^n$, think $\texttt{map}(\texttt{id}) = \texttt{id}$)



Output: $abc\#bac\#cb$

Pebble comparisons $\implies$ potential *dependencies* between their positions

$$\ldots(\#aa)(\#aa)(\#aa)(\#aa)(bb)(bb)(bb)\ldots$$

When we output something, *the position of* $\Downarrow$ *determines that of* $\downarrow$: redundant info