Fonctions polyrégulières : quelques développements récents

Lê Thành Dũng (Tito) Nguyễn — nltd@nguyentito.eu
travaux en collaboration avec :
Mikołaj Bojańczyk, Gaëtan Douéneau-Tabot, Sandra Kiefer et Pierre Pradic

Séminaire MoVe, LIS, Université Aix-Marseille, jeudi 24 février 2022

Classes habituelles de transductions

Plusieurs généralisations des langages rationnels à des *fonctions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{fonctions séquentielles}}_{\text{transducteurs finis } \textit{déterministes}} \subsetneq \underbrace{\text{fonctions rationnelles}}_{\text{transducteurs } \textit{non-déterministes}} \subsetneq \text{fonctions régulières}$$

Classes habituelles de transductions

Plusieurs généralisations des langages rationnels à des *fonctions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{fonctions séquentielles}}_{\text{transducteurs finis } \textit{déterministes}} \subsetneq \underbrace{\text{fonctions rationnelles}}_{\text{transducteurs } \textit{non-déterministes}} \subsetneq \text{fonctions régulières}$$

Bonnes propriétés, par ex. L rationnel $\implies f^{-1}(L)$ rationnel

Croissance *linéaire* : $|f(w)| = O(|w|)$.

Classes habituelles de transductions

Plusieurs généralisations des langages rationnels à des *fonctions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{fonctions séquentielles}}_{\text{transducteurs finis } \textit{déterministes}} \subsetneq \underbrace{\text{fonctions rationnelles}}_{\text{transducteurs } \textit{non-déterministes}} \subsetneq \text{fonctions régulières}$$

Bonnes propriétés, par ex. L rationnel $\implies f^{-1}(L)$ rationnel

Croissance *linéaire* : $|f(w)| = O(|w|)$. Au-delà :

- quelques vieilles classes : L-systèmes, transducteurs à pile de pile, ...
- *fonctions polyrégulières*, avec $|f(w)| = |w|^{O(1)}$

Classes habituelles de transductions

Plusieurs généralisations des langages rationnels à des *fonctions* $f : \Gamma^* \to \Sigma^*$:

$$\underbrace{\text{fonctions séquentielles}}_{\text{transducteurs finis } \textit{déterministes}} \subsetneq \underbrace{\text{fonctions rationnelles}}_{\text{transducteurs } \textit{non-déterministes}} \subsetneq \text{fonctions régulières}$$

Bonnes propriétés, par ex. L rationnel $\implies f^{-1}(L)$ rationnel

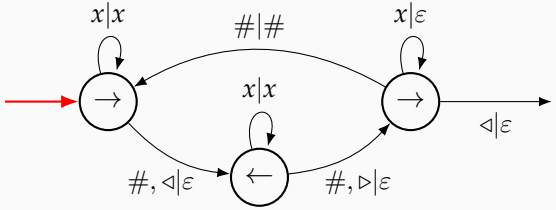
Croissance *linéaire* : $|f(w)| = O(|w|)$. Au-delà :

- quelques vieilles classes : L-systèmes, transducteurs à pile de pile, ...
- *fonctions polyrégulières*, avec $|f(w)| = |w|^{O(1)}$

Commençons par définir les régulières et polyrégulières

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



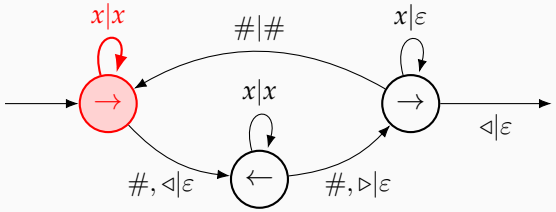
$(x \in \{a, b, c\})$

▷	a	b	c	#	b	a	c	#	c	b	◁

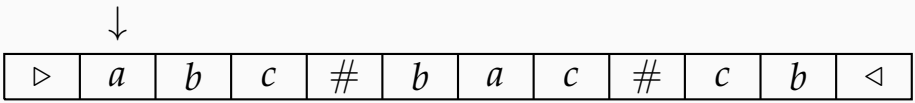
Sortie :

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



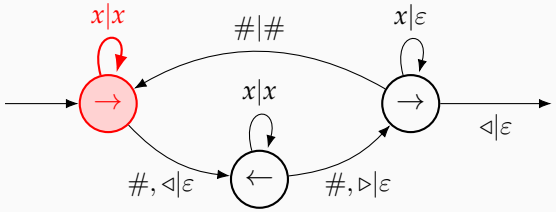
$(x \in \{a, b, c\})$



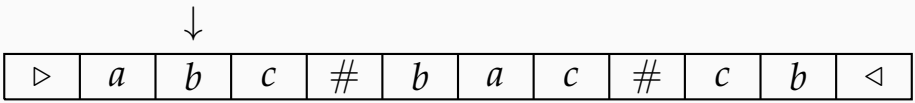
Sortie :

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



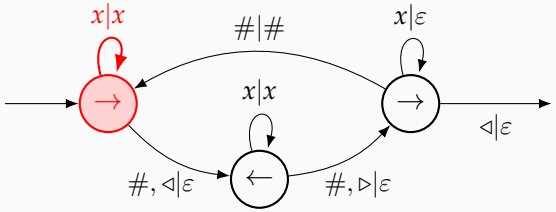
$(x \in \{a, b, c\})$



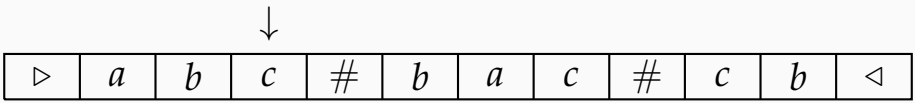
Sortie : a

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



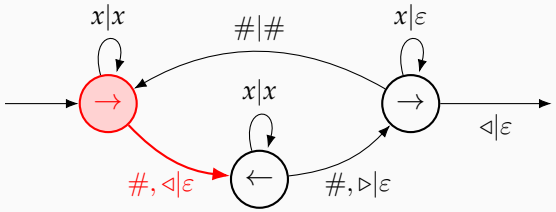
$(x \in \{a, b, c\})$



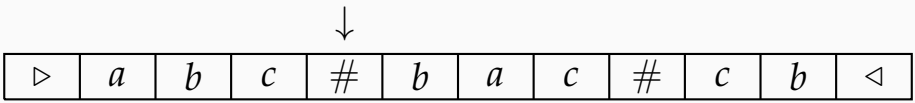
Sortie : ab

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



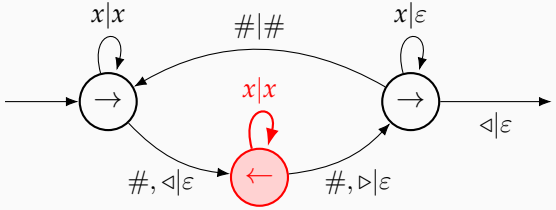
$(x \in \{a, b, c\})$



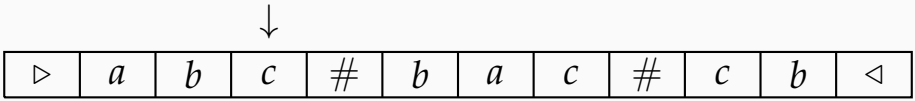
Sortie : abc

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



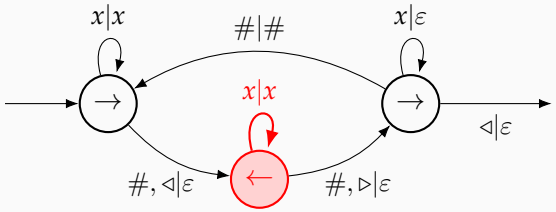
$(x \in \{a, b, c\})$



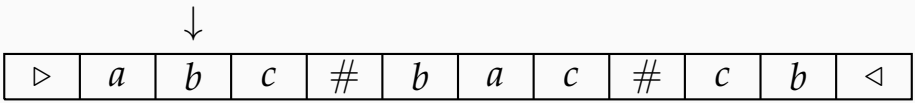
Sortie : abc

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



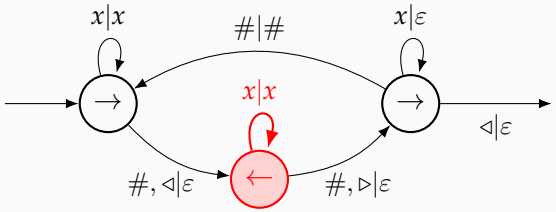
$(x \in \{a, b, c\})$



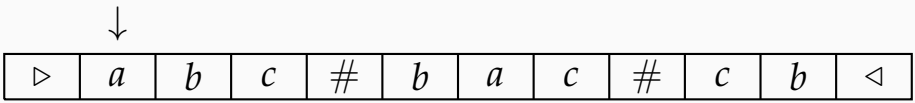
Sortie : *abcc*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



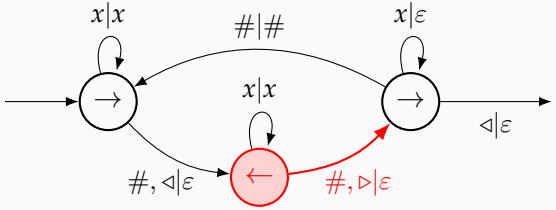
$(x \in \{a, b, c\})$



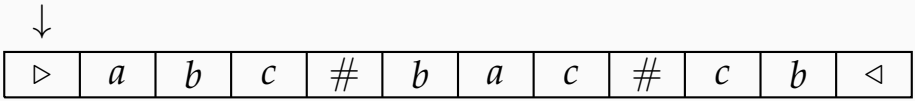
Sortie : *abccb*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



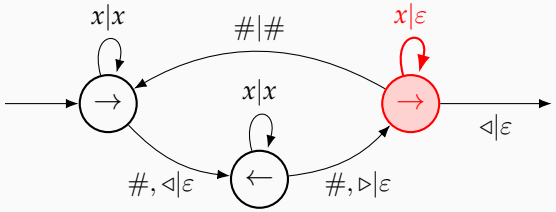
$(x \in \{a, b, c\})$



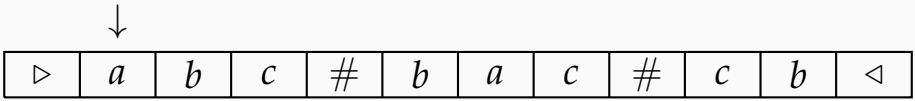
Sortie : *abccba*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



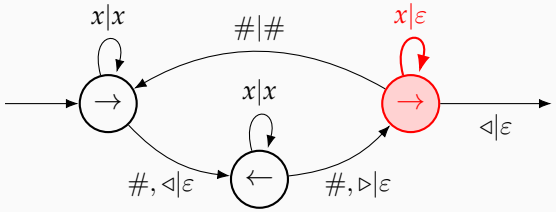
$(x \in \{a, b, c\})$



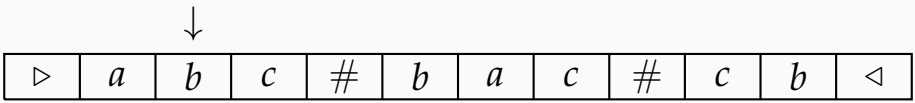
Sortie : *abccba*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



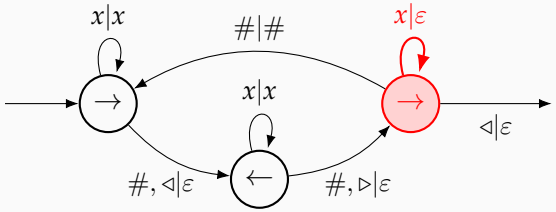
$(x \in \{a, b, c\})$



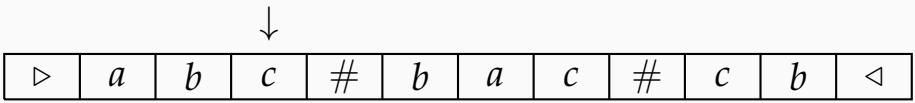
Sortie : *abccba*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



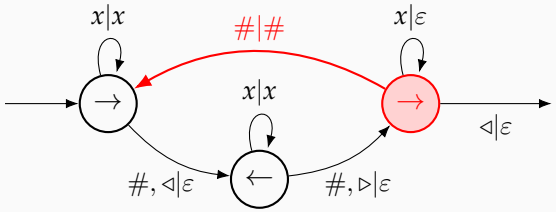
$(x \in \{a, b, c\})$



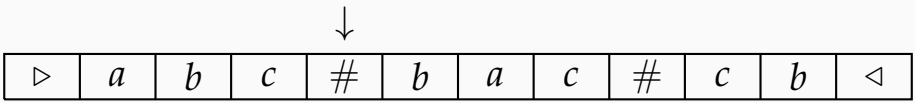
Sortie : *abccba*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



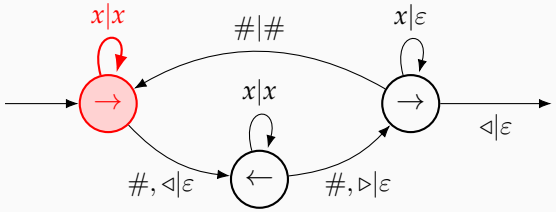
$(x \in \{a, b, c\})$



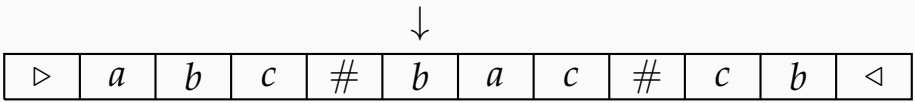
Sortie : *abccba*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



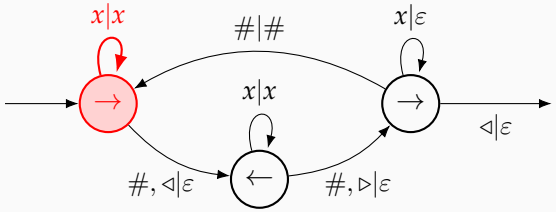
$(x \in \{a, b, c\})$



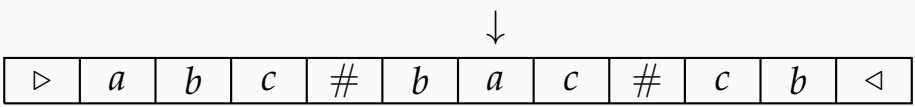
Sortie : $abccba\#$

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



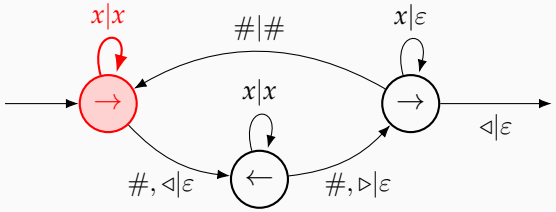
$(x \in \{a, b, c\})$



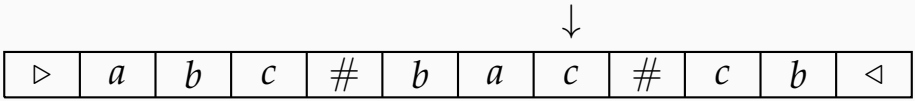
Sortie : $abccba\#b$

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



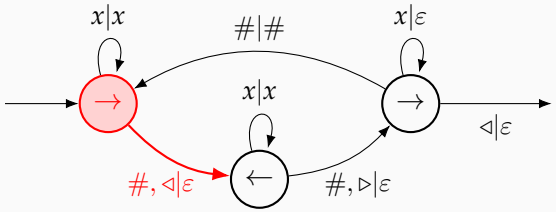
$(x \in \{a, b, c\})$



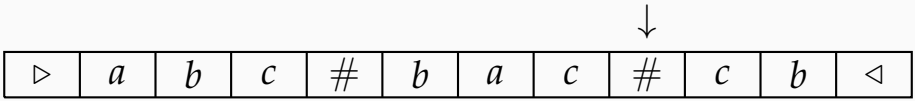
Sortie : $abccba\#ba$

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



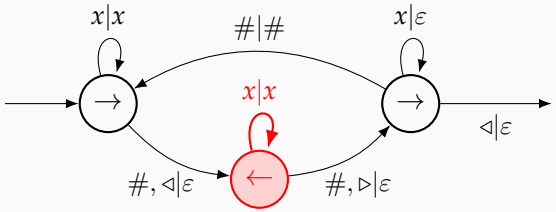
$(x \in \{a, b, c\})$



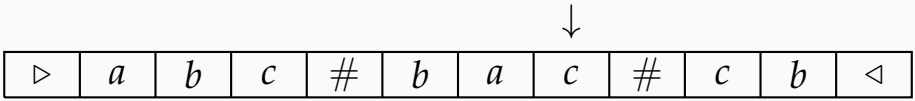
Sortie : *abccba#bac*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



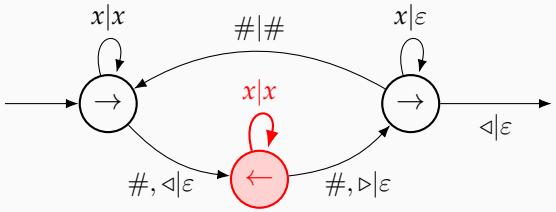
$(x \in \{a, b, c\})$



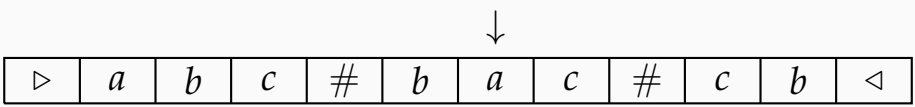
Sortie : *abccba#bac*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



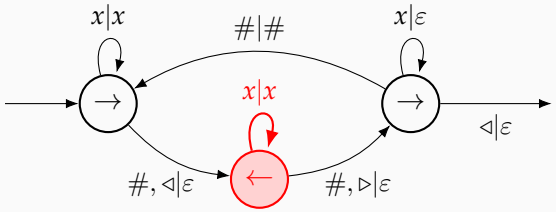
$(x \in \{a, b, c\})$



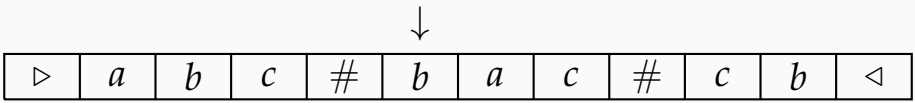
Sortie : *abccba#bacc*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



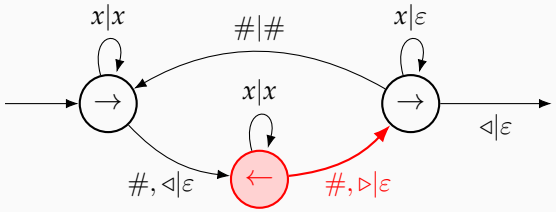
$(x \in \{a, b, c\})$



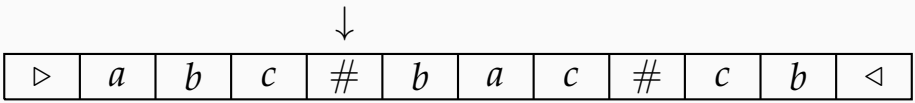
Sortie : *abccba#bacca*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



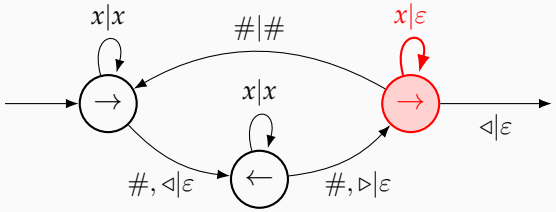
$(x \in \{a, b, c\})$



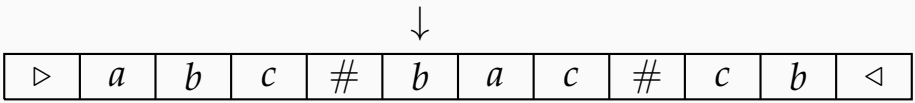
Sortie : *abccba#baccab*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



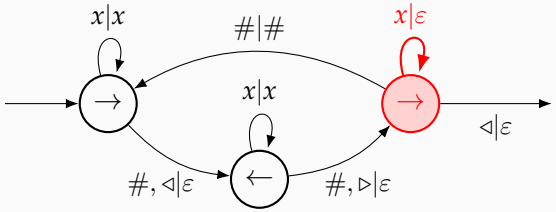
$(x \in \{a, b, c\})$



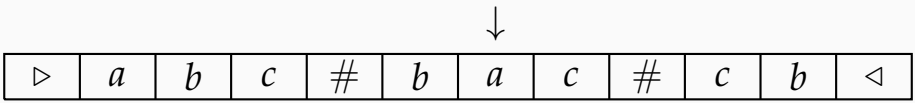
Sortie : *abccba#baccab*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



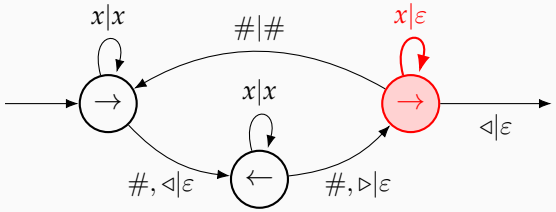
$(x \in \{a, b, c\})$



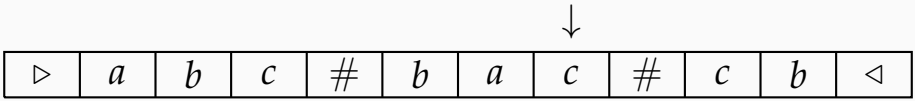
Sortie : *abccba#baccab*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



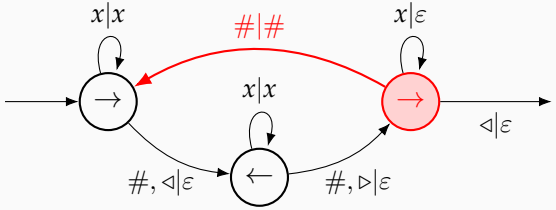
$(x \in \{a, b, c\})$



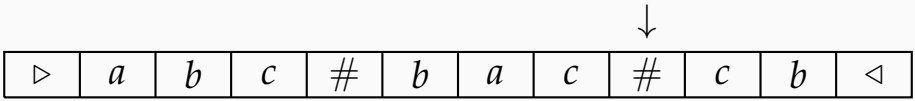
Sortie : *abccba#baccab*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



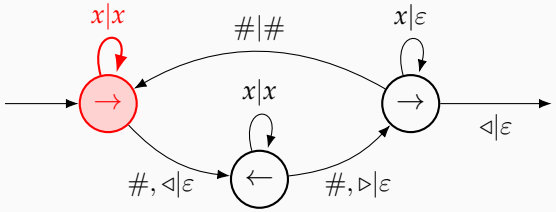
$(x \in \{a, b, c\})$



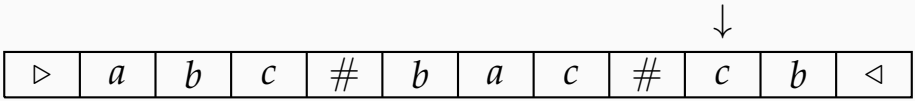
Sortie : *abccba#baccab*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



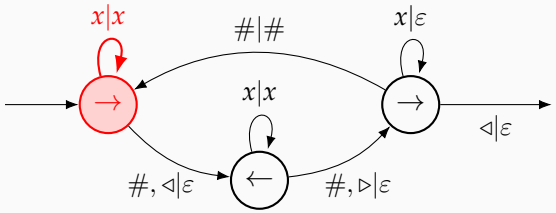
$(x \in \{a, b, c\})$



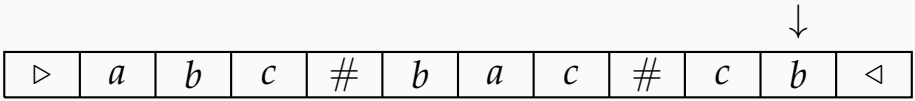
Sortie : *abccba#baccab#*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



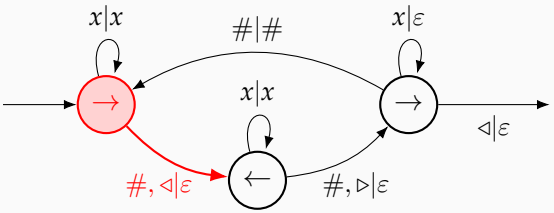
$(x \in \{a, b, c\})$



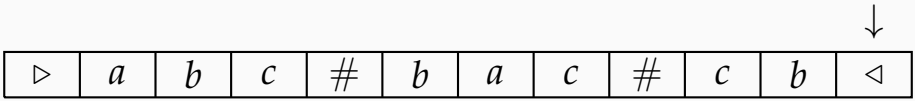
Sortie : *abccba#baccab#c*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



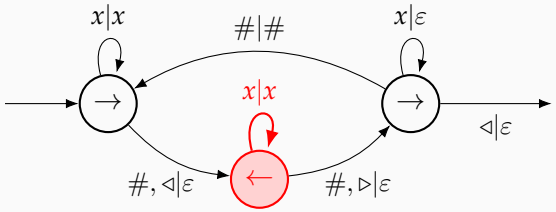
$(x \in \{a, b, c\})$



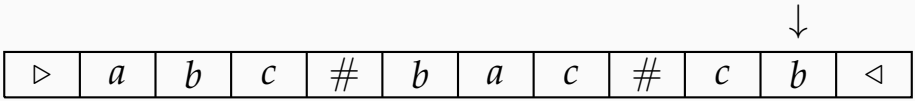
Sortie : *abccba#baccab#cb*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



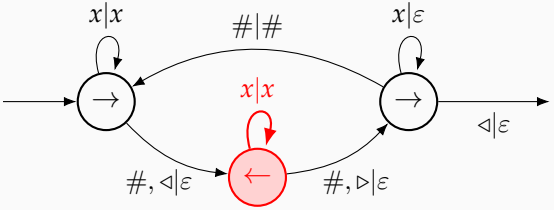
$(x \in \{a, b, c\})$



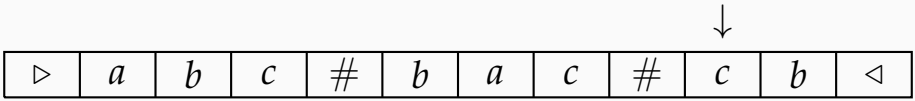
Sortie : *abccba#baccab#cb*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



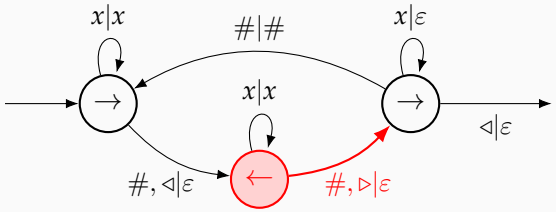
$(x \in \{a, b, c\})$



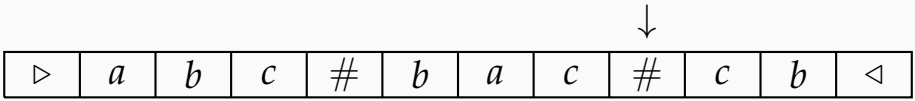
Sortie : *abccba#baccab#cbb*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



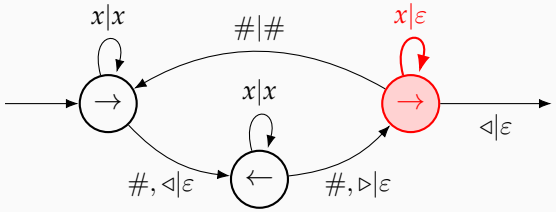
$(x \in \{a, b, c\})$



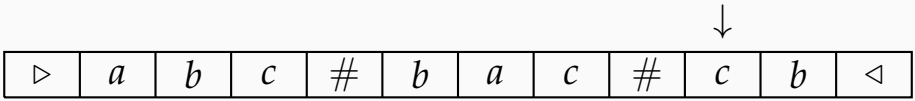
Sortie : *abccba#baccab#cbbc*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



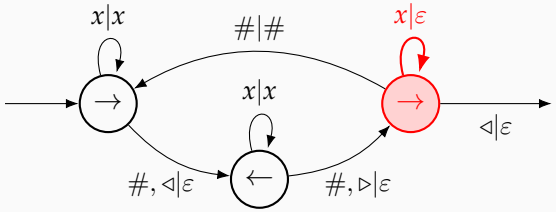
$(x \in \{a, b, c\})$



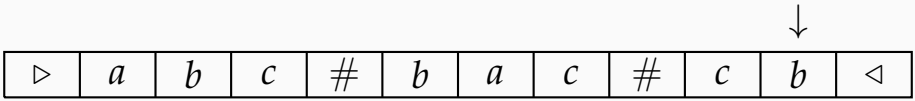
Sortie : *abccba#baccab#cbbc*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

Exemple : $w_1\# \dots \#w_n \longmapsto w_1 \cdot \texttt{reverse}(w_1)\# \dots \#w_n \cdot \texttt{reverse}(w_n)$



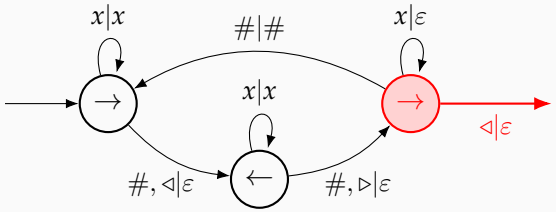
$(x \in \{a, b, c\})$



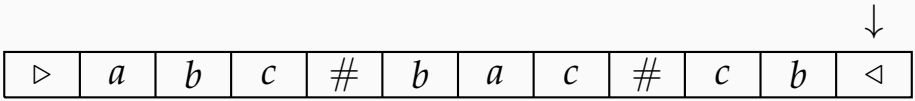
Sortie : *abccba#baccab#cbbc*

Transducteurs bidirectionnels (ou « boustrophédon » / EN : two-way)

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



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



Sortie : *abccba#baccab#cbbc*

Fonctions (poly)régulières

Definition

Fonctions régulières = calculées par des transducteurs bidirectionnels
(tête de lecture pouvant bouger à gauche ou à droite à chaque transition)

Fonctions (poly)régulières

Definition

Fonctions régulières = calculées par des transducteurs bidirectionnels
(tête de lecture pouvant bouger à gauche ou à droite à chaque transition)

Pour obtenir une croissance polynomiale : plusieurs têtes !

- sans restriction, ça donne Logspace [Hartmanis 1972]
- on met une condition de *pile* sur les têtes
  $\longrightarrow$ *transducteurs à jetons*, définition historique des polyrégulières

Fonctions (poly)régulières

Definition

Fonctions régulières = calculées par des transducteurs bidirectionnels
(tête de lecture pouvant bouger à gauche ou à droite à chaque transition)

Pour obtenir une croissance polynomiale : plusieurs têtes !

- sans restriction, ça donne Logspace [Hartmanis 1972]
- on met une condition de *pile* sur les têtes
  $\longrightarrow$ *transducteurs à jetons*, définition historique des polyrégulières

Plein de définitions équivalentes ; on en verra quelques-unes
toutes récentes pour les polyrég. [Bojańczyk 2018 ; Bojańczyk, Kiefer & Lhote 2019]

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$

Sortie :

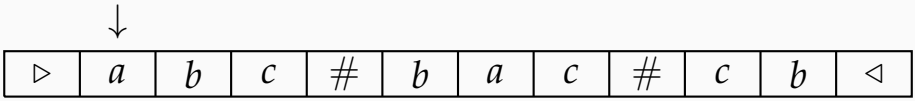
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

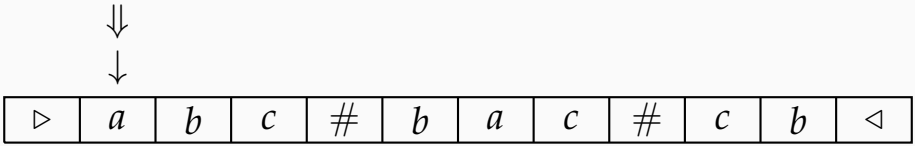
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

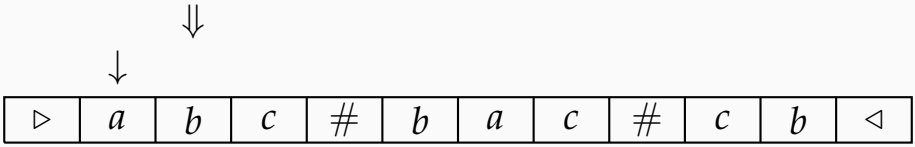
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

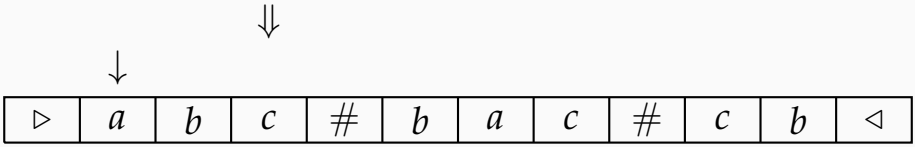
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

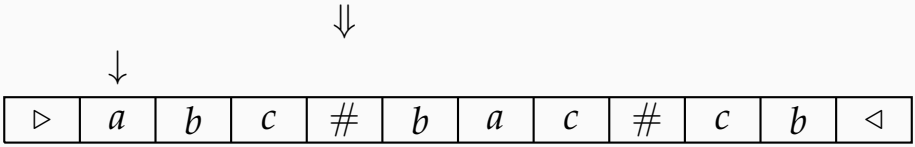
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

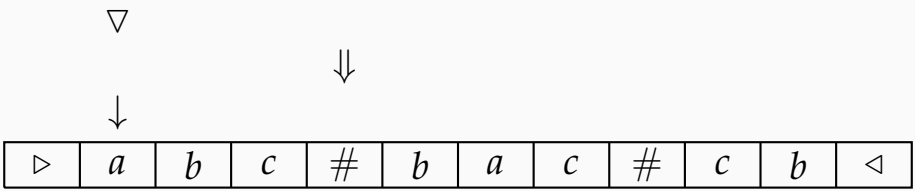
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie :

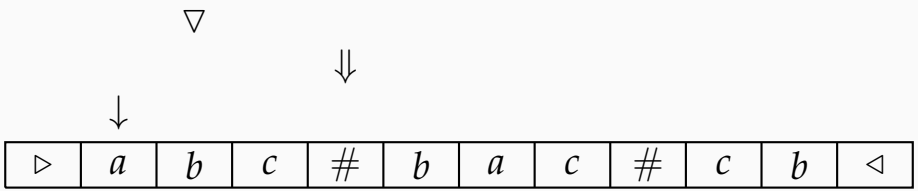
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : a

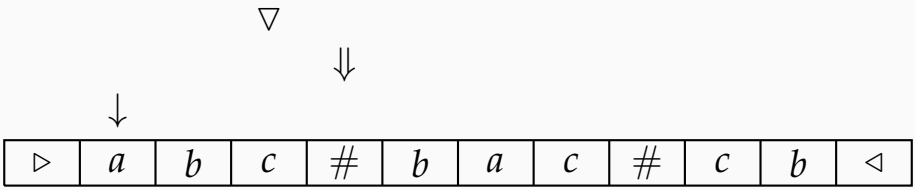
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : ab

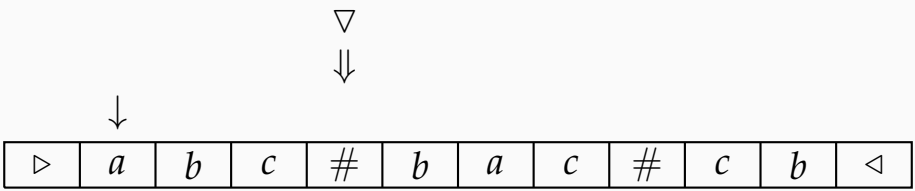
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

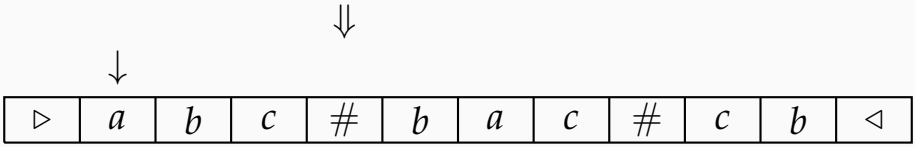
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

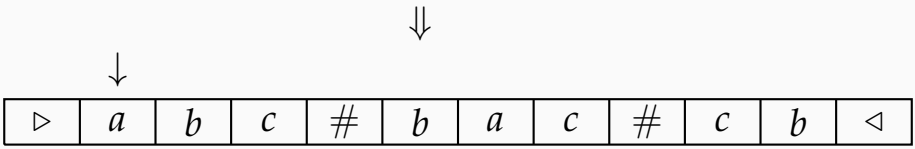
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

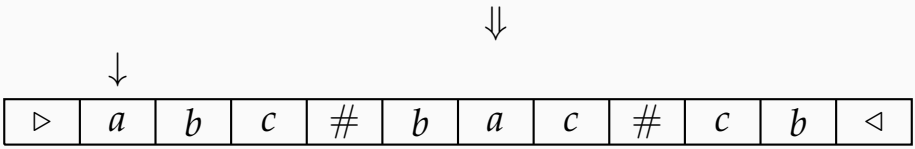
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

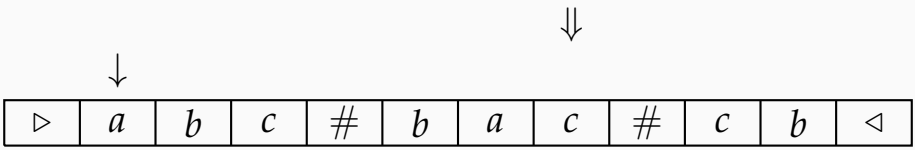
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

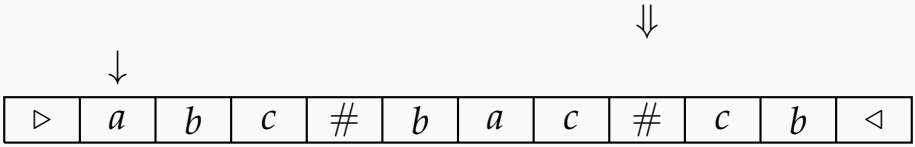
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

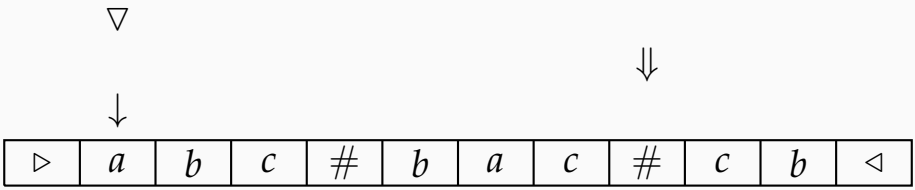
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : abc

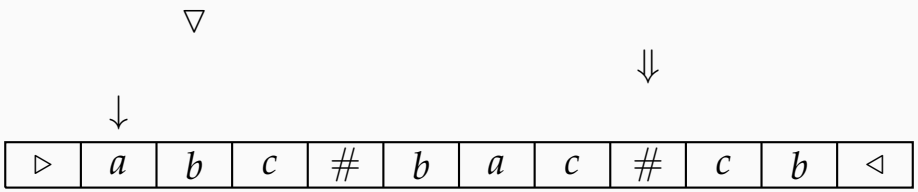
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abca*

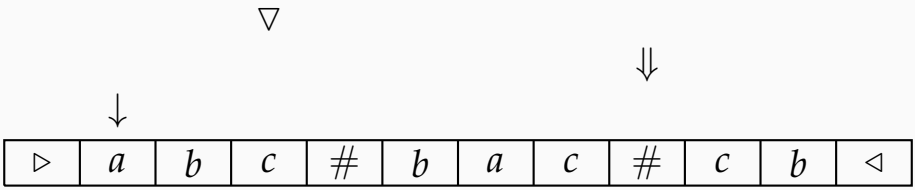
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcab*

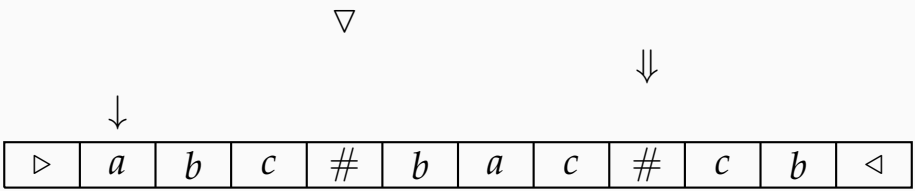
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc*

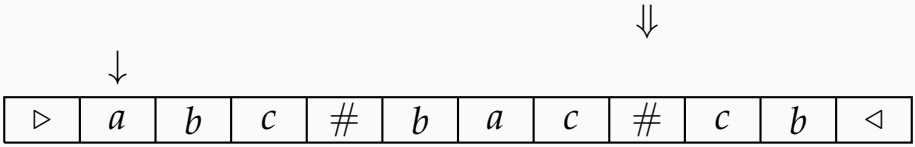
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc*

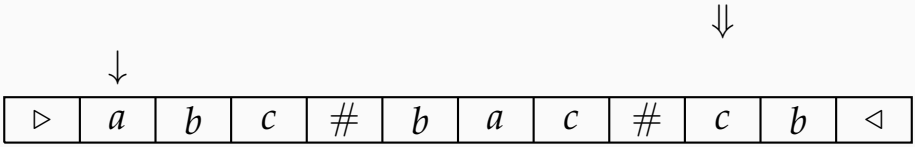
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc*

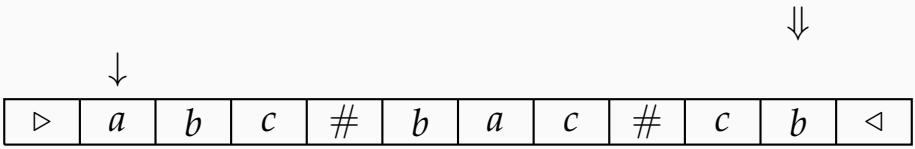
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc*

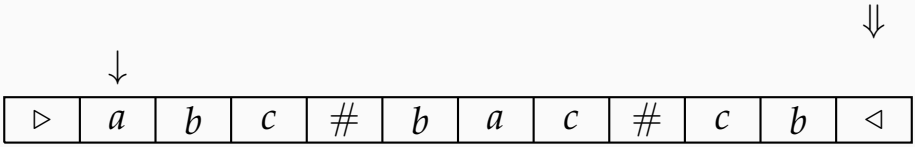
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc*

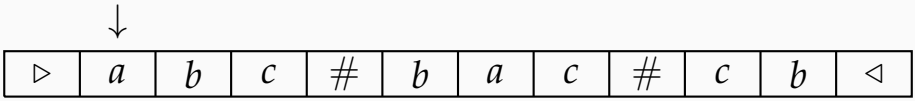
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

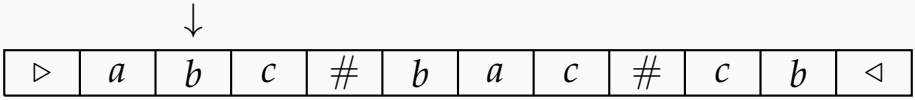
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

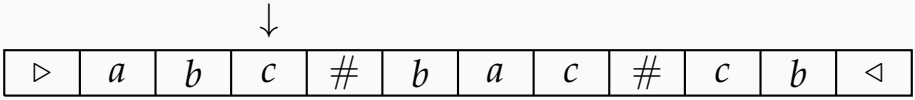
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

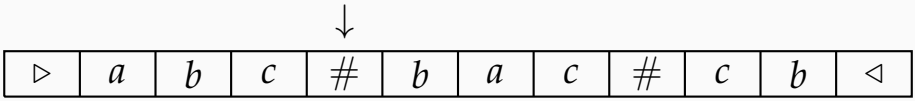
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

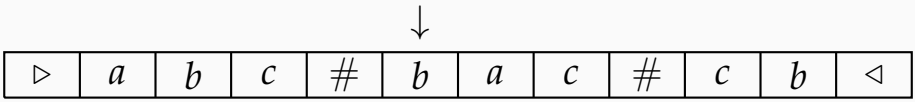
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

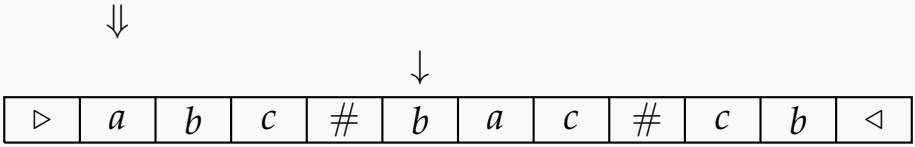
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

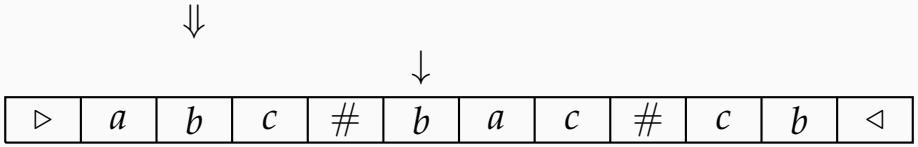
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

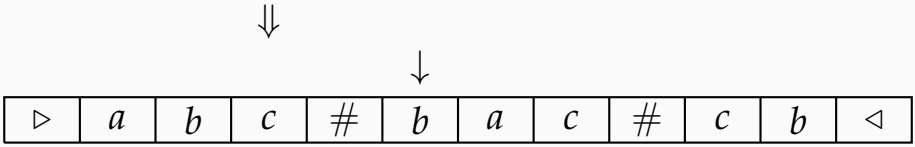
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

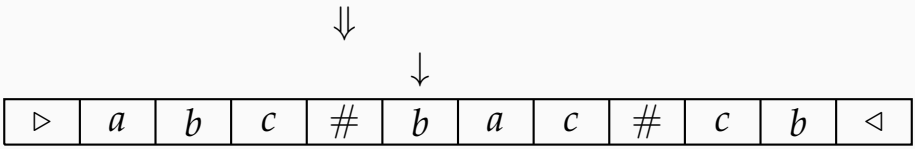
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

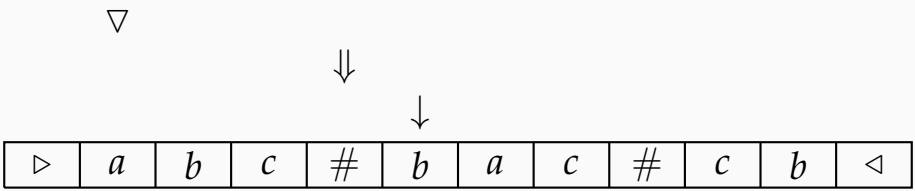
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

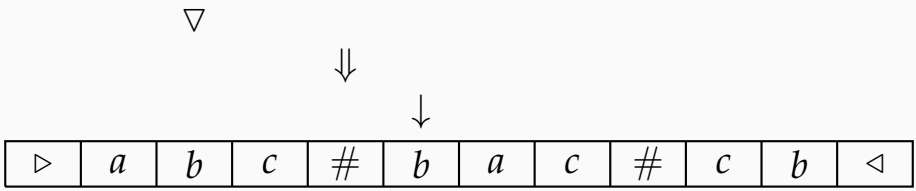
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

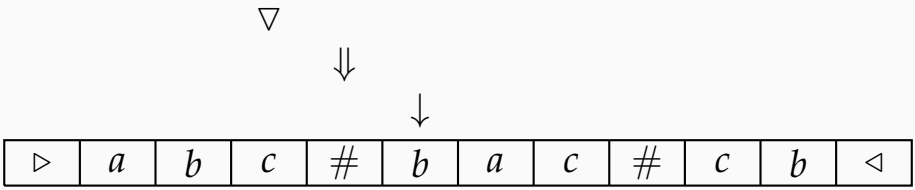
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

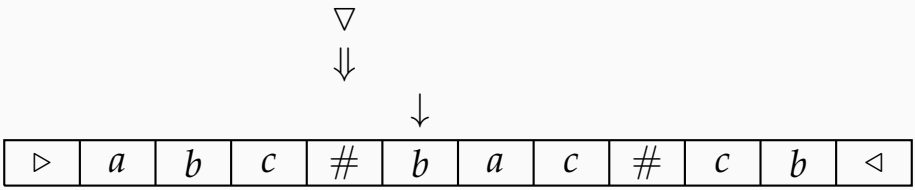
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

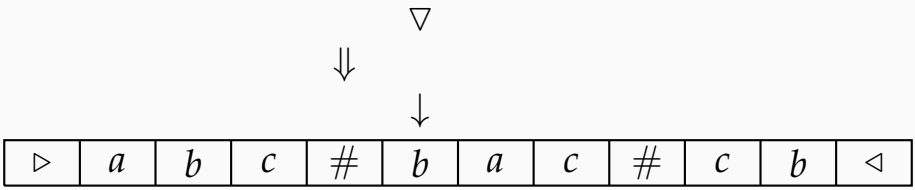
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#$

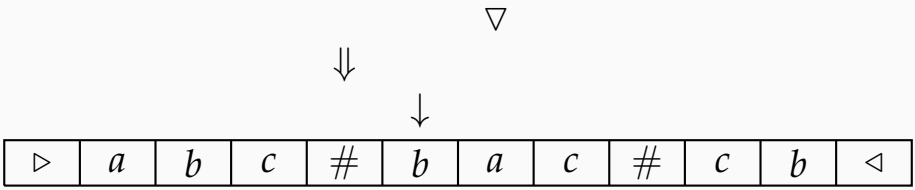
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



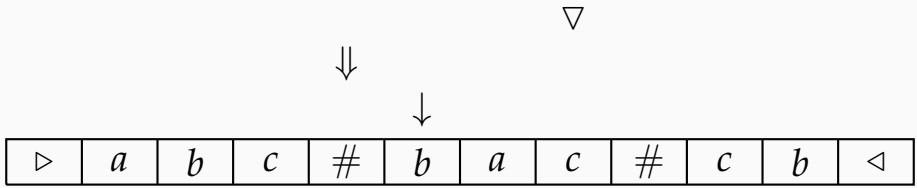
Sortie : $abcabc\#b$

[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)



Sortie : $abcabc\#ba$

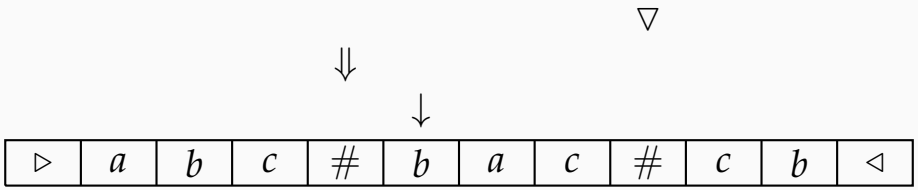
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

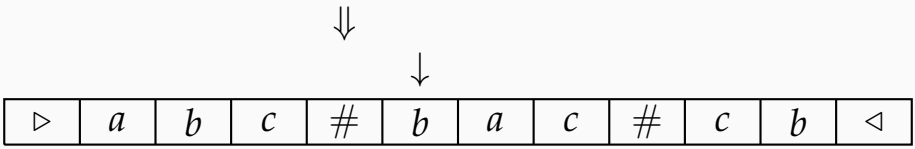
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

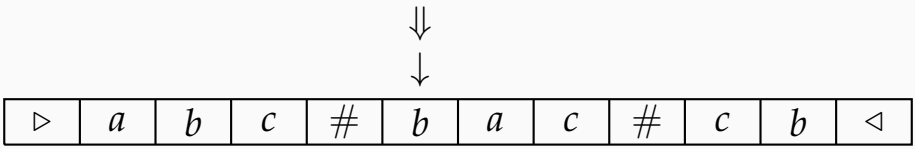
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

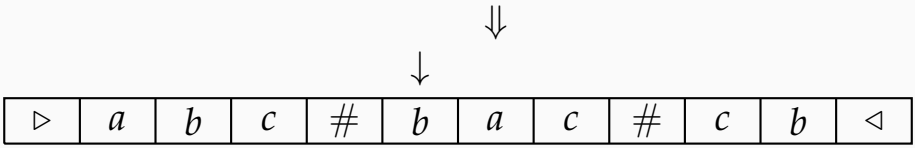
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

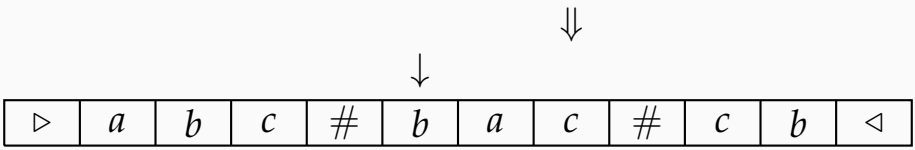
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

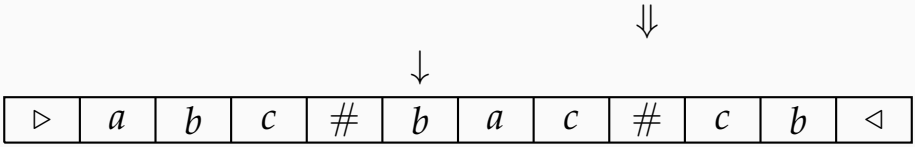
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

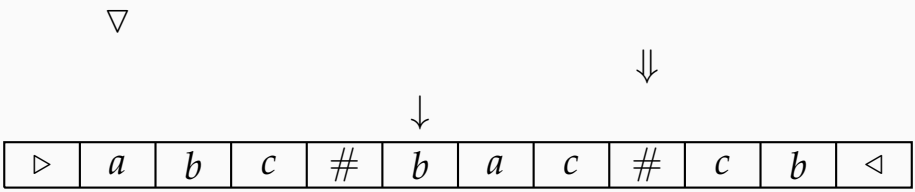
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

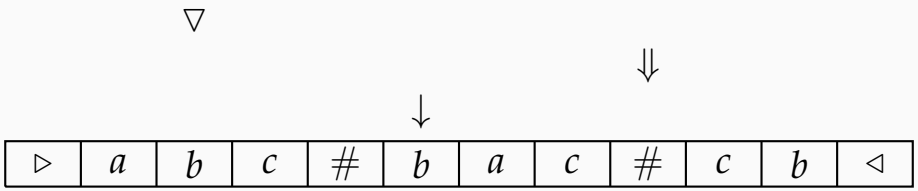
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

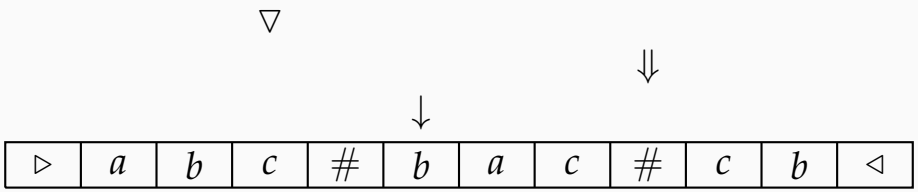
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

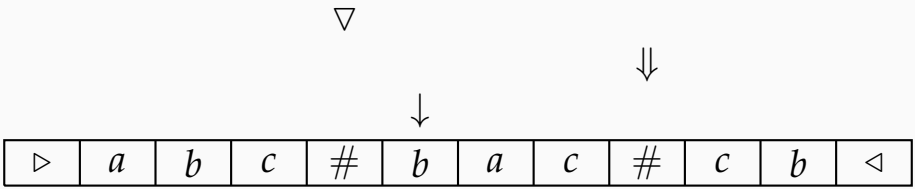
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

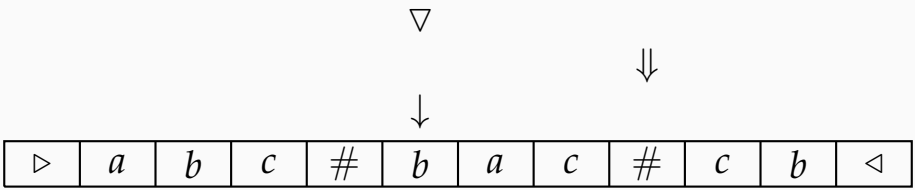
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bac$

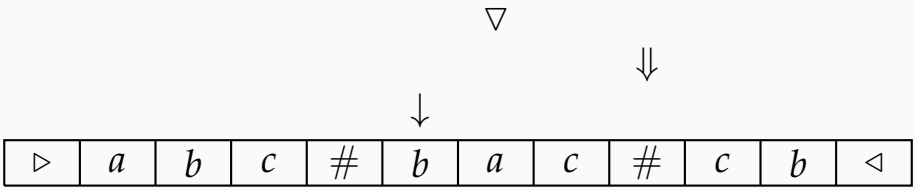
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacb$

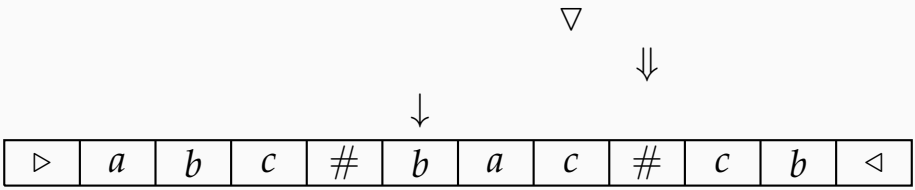
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacba$

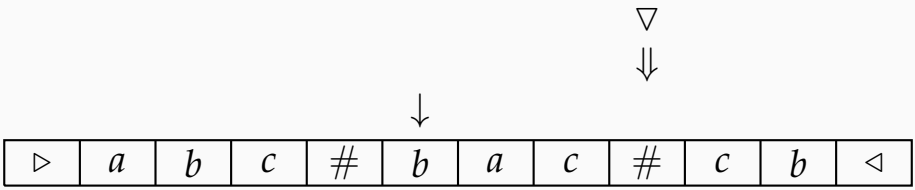
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac$

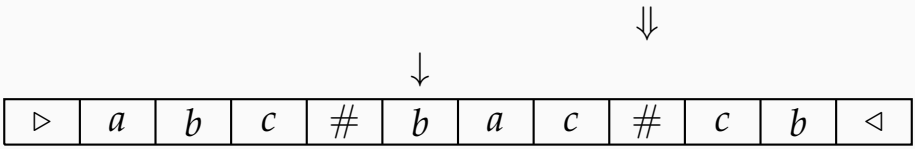
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac$

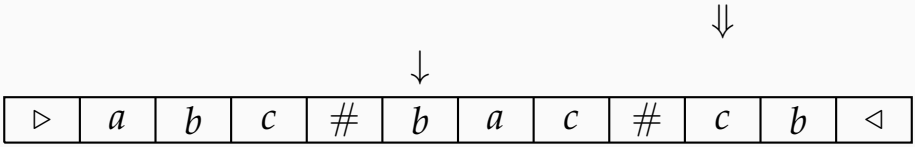
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac$

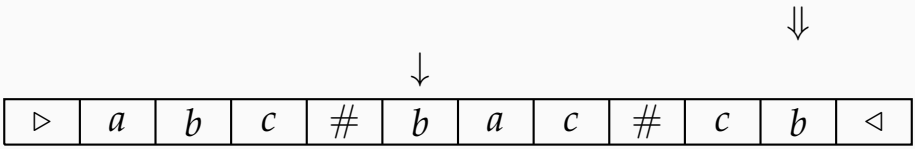
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac$

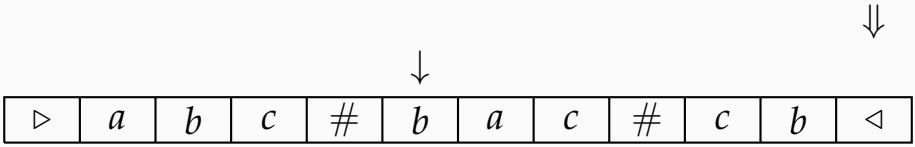
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac$

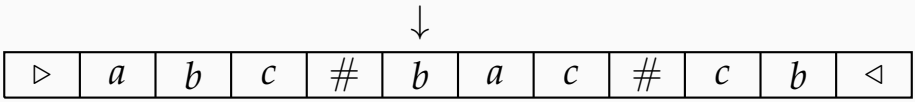
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

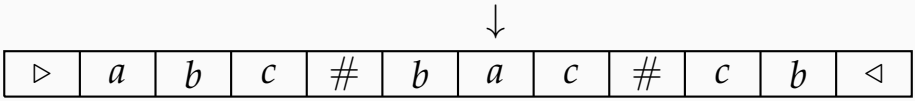
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

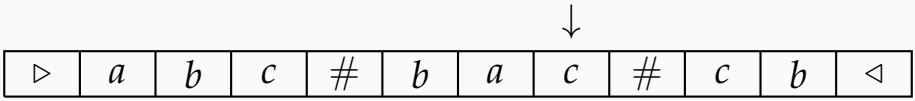
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

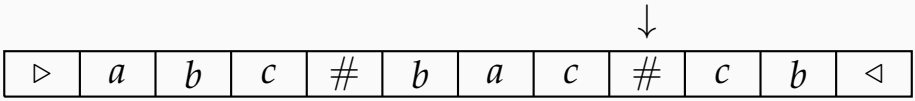
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

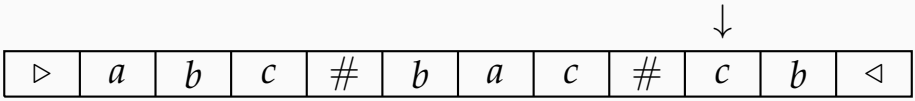
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

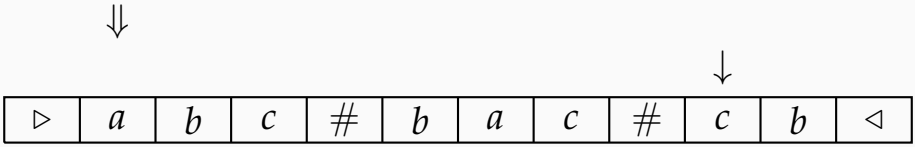
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

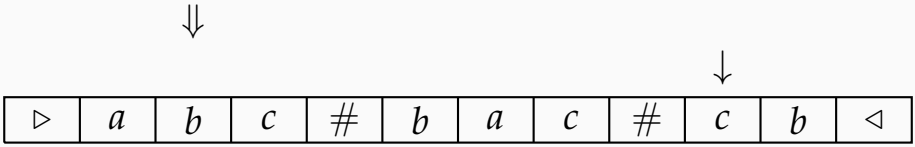
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

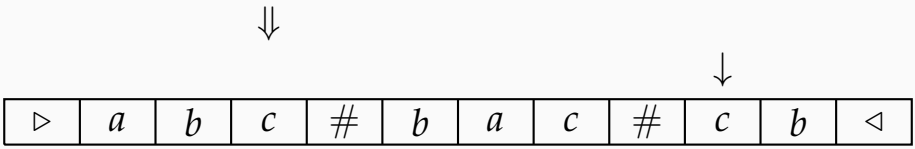
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

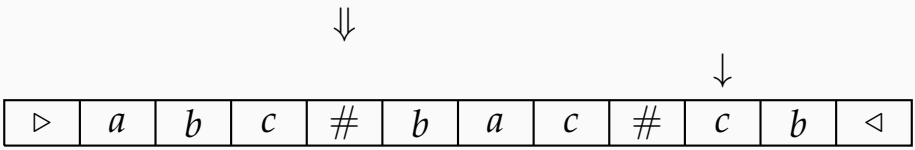
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

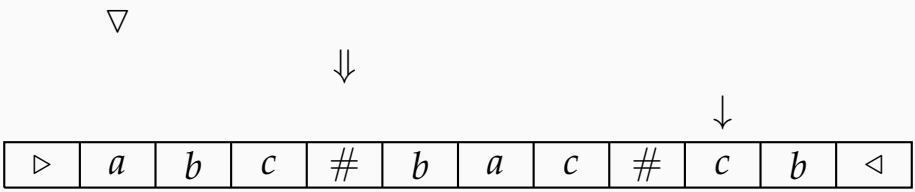
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

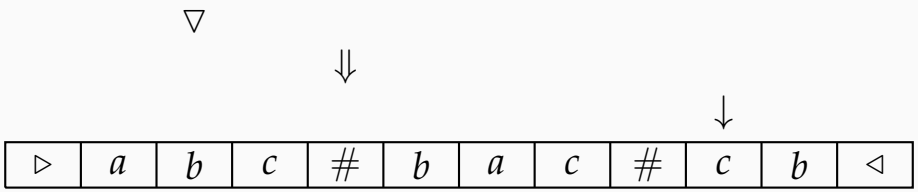
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

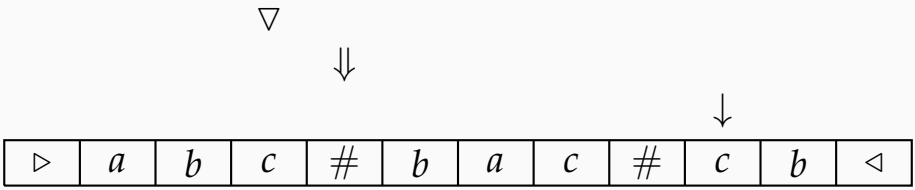
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

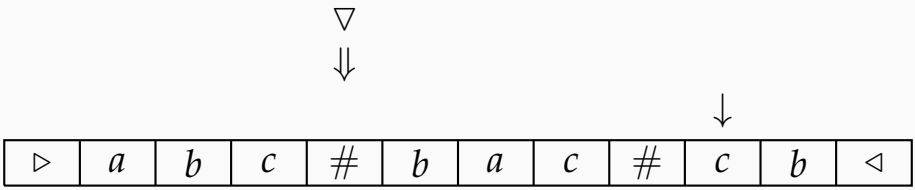
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

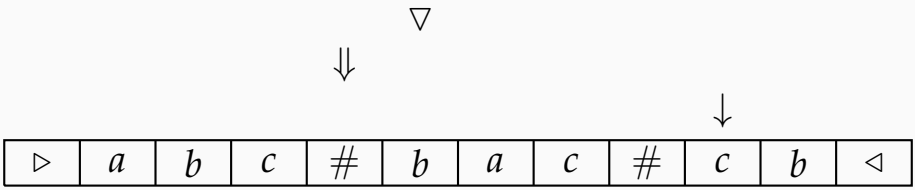
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

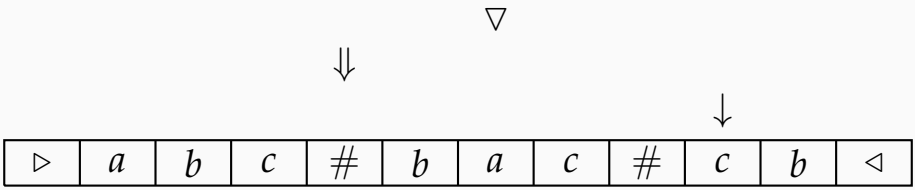
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

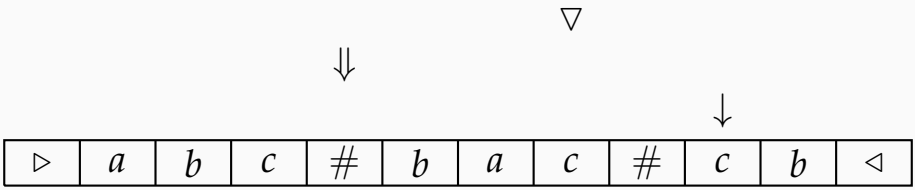
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

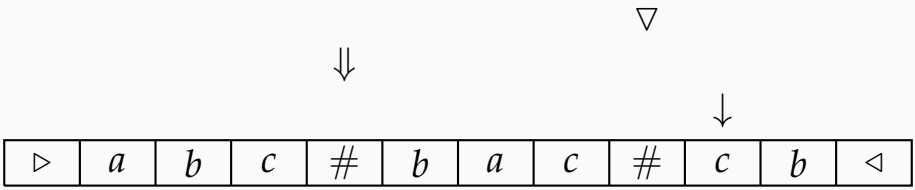
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

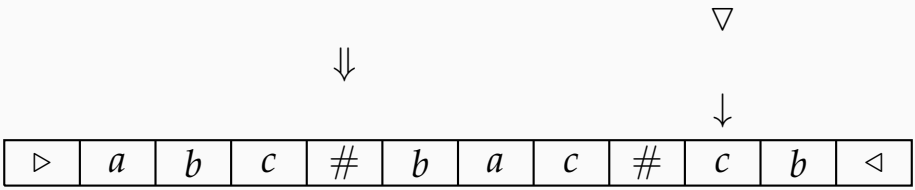
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#$

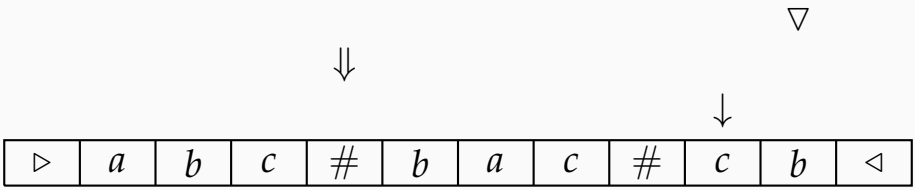
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$



Sortie : $abcabc\#bacbac\#c$

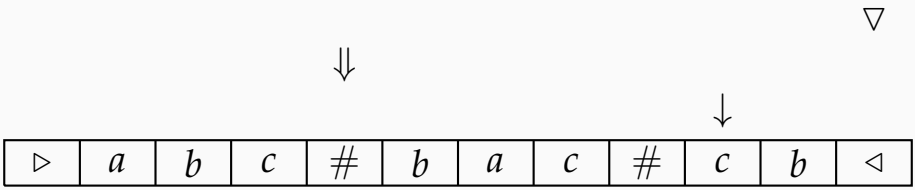
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

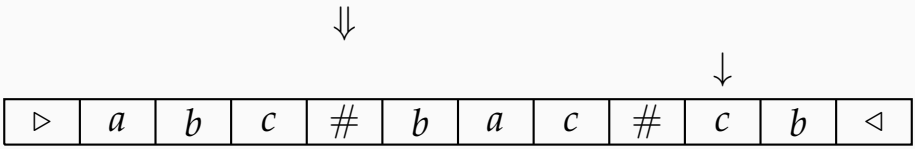
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

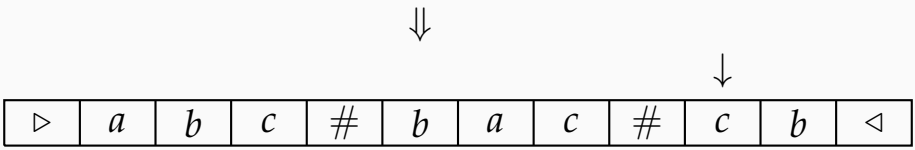
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

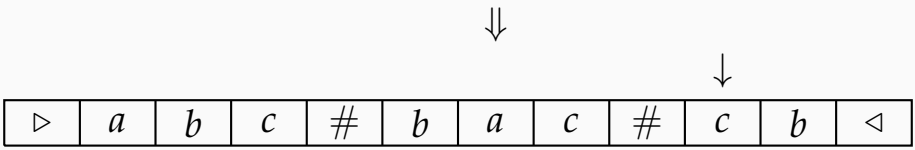
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

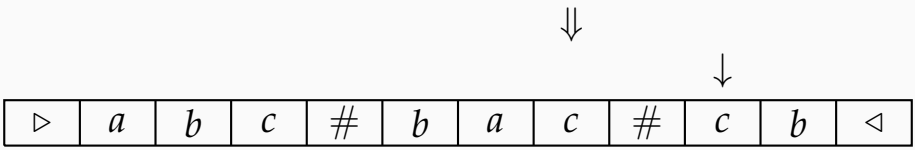
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

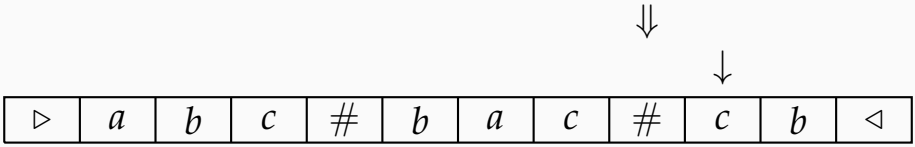
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

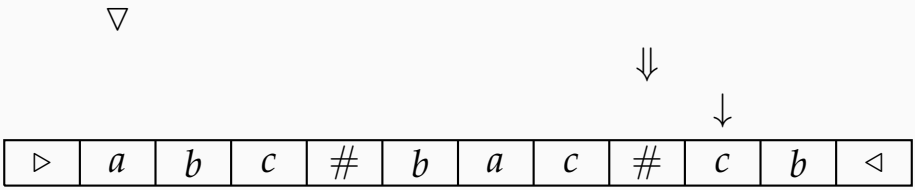
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

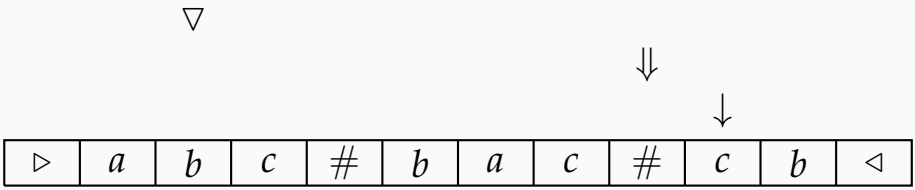
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

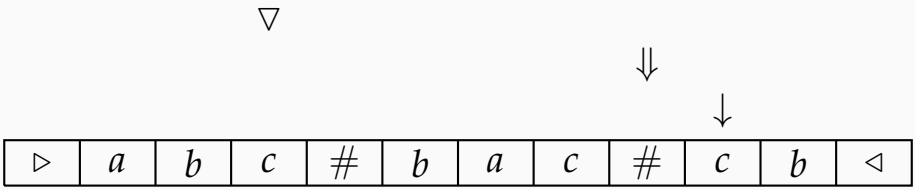
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

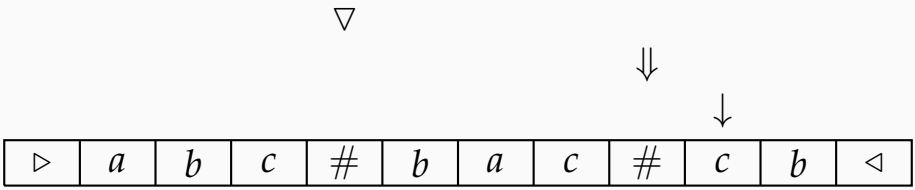
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

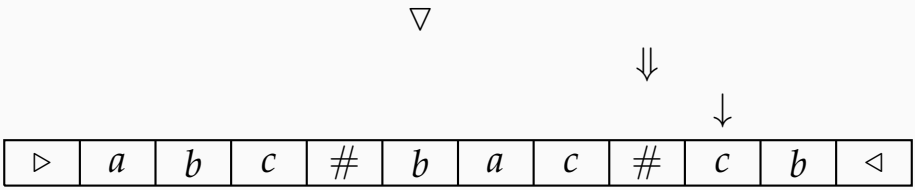
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

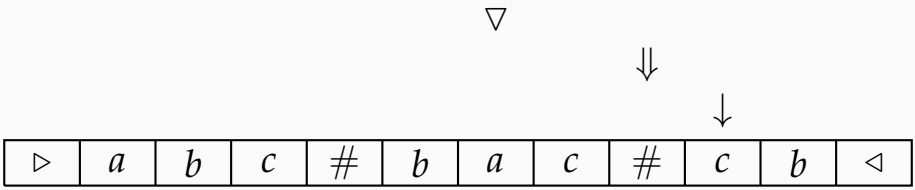
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

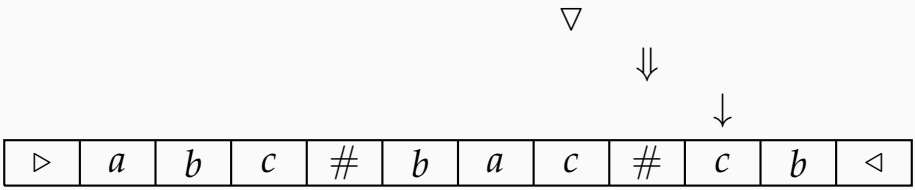
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

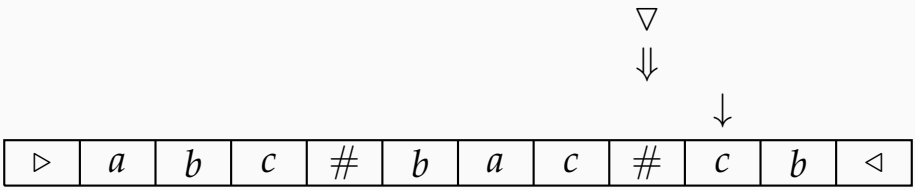
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cb$

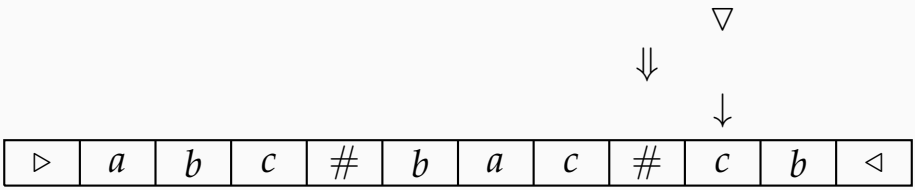
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cb*

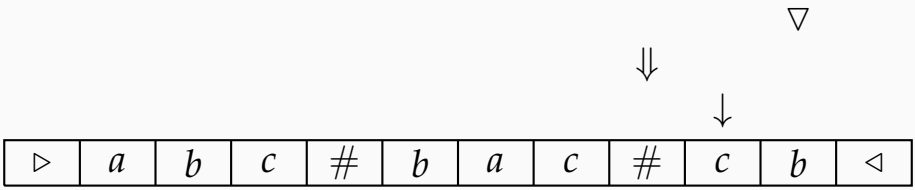
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbc*

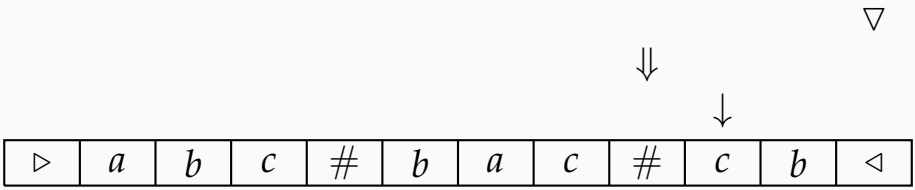
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

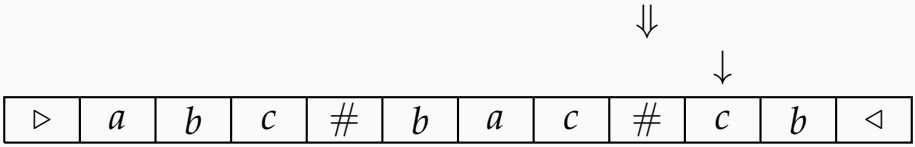
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

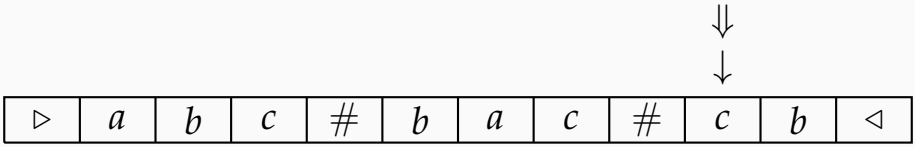
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

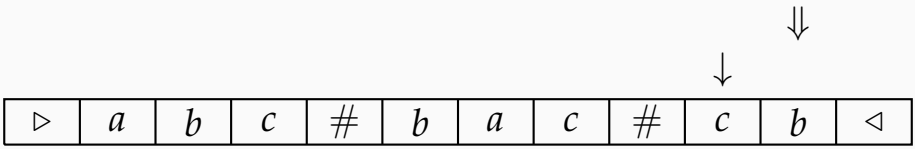
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

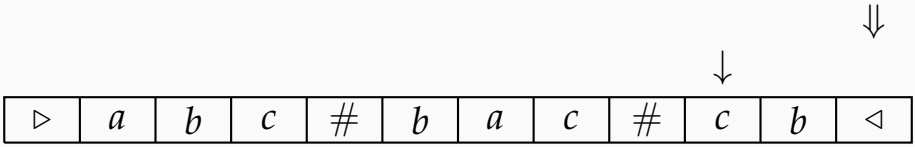
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

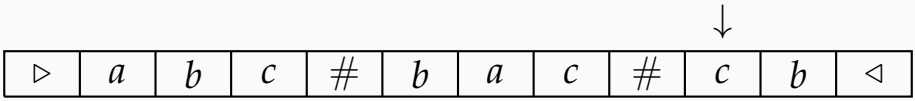
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

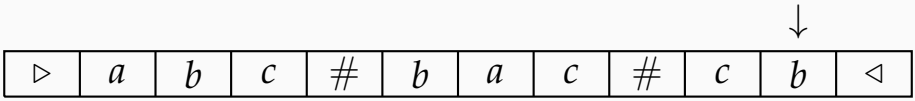
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : *abcabc#bacbac#cbcb*

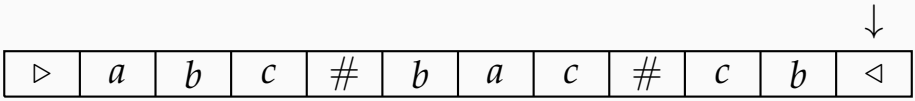
[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Transducteurs à jetons[1] (EN : pebble / PL : kamyki) [Milo, Suciu & Vianu 2000]

Fonctions polyrégulières = calculées par des transducteurs à k jetons ($k \geq 1$)

Ensemble fini d'états + *pile* de hauteur $\leqslant k$ de têtes bidirectionnelles («jetons»)

$$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$$



Sortie : $abcabc\#bacbac\#cbcb$

[1] https://fr.wikipedia.org/wiki/Automate_à_jetons

Remarques et questions sur les transducteurs à jetons

`mapPower` : $w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Le transducteur exemple regarde si le 1^{er} et le $3^{\text{è}}$ jeton sont au même endroit
  $\longrightarrow$ ces comparaisons sont-elles nécessaires pour calculer `mapPower` ?

Remarques et questions sur les transducteurs à jetons

$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Le transducteur exemple regarde si le 1er et le 3è jeton sont au même endroit
  $\longrightarrow$ ces comparaisons sont-elles nécessaires pour calculer `mapPower` ?
- $|\texttt{mapPower}(w)| = O(|w|^2)$, or avec k jetons on peut croître jusqu'en $|w|^k$
  $\longrightarrow$ pourrait-on calculer `mapPower` avec seulement 2 jetons ?

Remarques et questions sur les transducteurs à jetons

$\texttt{mapPower} : w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$

- Le transducteur exemple regarde si le 1er et le 3è jeton sont au même endroit
  $\longrightarrow$ ces comparaisons sont-elles nécessaires pour calculer `mapPower` ?
- $|\texttt{mapPower}(w)| = O(|w|^2)$, or avec k jetons on peut croître jusqu'en $|w|^k$
  $\longrightarrow$ pourrait-on calculer `mapPower` avec seulement 2 jetons ?

Contributions présentées dans cet exposé

- Fonctions polyrégulières *sans comparaison* [N., Noûs & Pradic, ICALP'21]
  déf : calculées par des transducteurs à jetons sans comp. (ou « aveugles »)
- *Croissance* des fonctions polyrégulières vs usage de « ressources »
  [Bojańczyk, Douéneau-Tabot, Kiefer, N. & Pradic, en cours de rédaction]

Les comparaisons sont nécessaires

Théorème : résultats de séparation

Les fonctions suivantes sont polyrégulières, mais *ne sont pas* sans comparaison :

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [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. ; résultat plus fort dans [Douéneau-Tabot MFCS'21]
- `mapPower` $\to$ nouveau ; en lien avec les questions de croissance

Les comparaisons sont nécessaires

Théorème : résultats de séparation

Les fonctions suivantes sont polyrégulières, mais *ne sont pas* sans comparaison :

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [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. ; résultat plus fort dans [Douéneau-Tabot MFCS'21]
- `mapPower` $\to$ nouveau ; en lien avec les questions de croissance

$\longrightarrow$ Une nouvelle classe de fonctions, définie assez naturellement. Exemple :

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

Les comparaisons sont nécessaires

Théorème : résultats de séparation

Les fonctions suivantes sont polyrégulières, mais *ne sont pas* sans comparaison :

- $a^n \mapsto a\#aa\#\ldots\#a^n$ [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. ; résultat plus fort dans [Douéneau-Tabot MFCS'21]
- `mapPower` $\to$ nouveau ; en lien avec les questions de croissance

$\longrightarrow$ Une nouvelle classe de fonctions, définie assez naturellement. Exemple :

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

Mais est-elle robuste/canonique ? A-t-elle de bonnes propriétés ?

Théorème : stabilité par composition

Si f et g sont polyrégulières sans comparaison, alors $g \circ f$ aussi.

Fonctions polyrégulières sans comparaison et composition

Théorème : stabilité par composition

Si f et g sont polyrégulières sans comparaison, alors $g \circ f$ aussi.

Mais plus encore :

Définition alternative des fonctions polyrégulières sans comparaison

C'est la plus petite classe stable par composition contenant les régulières et `cfsquaring` (rappel : $\texttt{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$).

Fonctions polyrégulières sans comparaison et composition

Théorème : stabilité par composition

Si f et g sont polyrégulières sans comparaison, alors $g \circ f$ aussi.

Mais plus encore :

Définition alternative des fonctions polyrégulières sans comparaison

C'est la plus petite classe stable par composition contenant les régulières et `cfsquaring` (rappel : $\texttt{cfsquaring}(abb) = \underline{a}abb\underline{b}abb\underline{b}abb$).

Analogue à une caractérisation des polyrégulières générales où `cfsquaring` est remplacé par $abb \mapsto \underline{a}bba\underline{b}bab\underline{b}$ [Bojańczyk 2018]

Fonctions polyrégulières sans comparaison et λ-calcul

La vraie motivation pour introduire les polyrég. sans comp. :

- Mes travaux de thèse avec P. Pradic (cf. exposé à l'I2M dans 1 semaine !) :
  caractérisation des fonctions régulières dans un λ-calcul à types *linéaires*
  (linéarité $\simeq$ sans copie / « single use restriction »)

Fonctions polyrégulières sans comparaison et λ-calcul

La vraie motivation pour introduire les polyrég. sans comp. :

- Mes travaux de thèse avec P. Pradic (cf. exposé à l'I2M dans 1 semaine !) :
  caractérisation des fonctions régulières dans un λ-calcul à types *linéaires*
  (linéarité $\simeq$ sans copie / « single use restriction »)
- En relâchant un peu la linéarité, on tombe sur une classe de fonctions qui ne préexistait pas...

Fonctions polyrégulières sans comparaison et λ-calcul

La vraie motivation pour introduire les polyrég. sans comp. :

- Mes travaux de thèse avec P. Pradic (cf. exposé à l'I2M dans 1 semaine !) :
  caractérisation des fonctions régulières dans un λ-calcul à types *linéaires*
  (linéarité $\simeq$ sans copie / « single use restriction »)
- En relâchant un peu la linéarité, on tombe sur une classe de fonctions qui ne préexistait pas...
- Découverte (aussi dans ma thèse) :
  ça correspond à enlever les comparaisons des transducteurs à jetons !
  $\longrightarrow$ argument supplémentaire pour la canonicité de ces fonctions

Fonctions polyrégulières sans comparaison et λ-calcul

La vraie motivation pour introduire les polyrég. sans comp. :

- Mes travaux de thèse avec P. Pradic (cf. exposé à l'I2M dans 1 semaine !) :
  caractérisation des fonctions régulières dans un λ-calcul à types *linéaires*
  (linéarité $\simeq$ sans copie / « single use restriction »)
- En relâchant un peu la linéarité, on tombe sur une classe de fonctions qui ne préexistait pas...
- Découverte (aussi dans ma thèse) :
  ça correspond à enlever les comparaisons des transducteurs à jetons !
  $\longrightarrow$ argument supplémentaire pour la canonicité de ces fonctions

Remarque : la stabilité par composition est *automatique* avec la déf. λ-calcul
(derrière tout ça, structure commune de catégorie monoïdale close)

Croissance vs nombre de jetons

Théorème : minimisation de jetons **(le plus technique dans l'article ICALP'21)**

$f : \Gamma^* \to \Sigma^*$ est polyrégulière sans comparaison et $|f(w)| = O(|w|^k)$

$\iff$

f est calculable par un transducteur à k jetons sans comparaison

Inspiration : min. de jetons pour les polyrég. générales [Lhote LICS'20]

Croissance vs nombre de jetons

Théorème : minimisation de jetons **(le plus technique dans l'article ICALP'21)**

$f : \Gamma^* \to \Sigma^*$ est polyrégulière sans comparaison et $|f(w)| = O(|w|^k)$

$\iff$

f est calculable par un transducteur à k jetons sans comparaison

Inspiration : min. de jetons pour les polyrég. générales [Lhote LICS'20]

... on adapte l'approche de l'article alors que son résultat est *faux* :

`mapPower`, à croissance $O(n^2)$, nécessite 3 jetons !

Croissance vs nombre de jetons

Théorème : minimisation de jetons **(le plus technique dans l'article ICALP'21)**

$f : \Gamma^* \to \Sigma^*$ est polyrégulière sans comparaison et $|f(w)| = O(|w|^k)$

$\iff$

f est calculable par un transducteur à k jetons sans comparaison

Inspiration : min. de jetons pour les polyrég. générales [Lhote LICS'20]

... on adapte l'approche de l'article alors que son résultat est *faux* :

`mapPower`, à croissance $O(n^2)$, nécessite 3 jetons !

Théorème

Pour tout $k \geq 2$ il existe f polyrég. nécessitant k jetons telle que $|f(w)| = O(|w|^2)$.

Par contre, polyrégulière + $O(n) \implies$ régulière i.e. 1 jeton

(corollaire de [Engelfriet, Inaba & Maneth 2021] sur les transducteurs d'arbres)

Minorations sur le nombre de jetons

Théorème

`mapPower` : $w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$ nécessite 3 jetons

Deux démonstrations différentes :

- élémentaire : utilisation du lemme de pompage sur les *langages de sortie* des fonctions régulières (k-itérativité) [Rozoy 1986, redécouvert par Smith 2014]
  esquive une analyse détaillée du déroulement du calcul
- conceptuelle : détour par des *atomes* (alphabets infinis)
  + théorème (technique) de désatomisation

Minorations sur le nombre de jetons

Théorème

`mapPower` : $w_0\#\ldots\#w_n \longmapsto (w_0)^n\#\ldots\#(w_n)^n$ nécessite 3 jetons

Deux démonstrations différentes :

- élémentaire : utilisation du lemme de pompage sur les *langages de sortie* des fonctions régulières (k-itérativité) [Rozoy 1986, redécouvert par Smith 2014]
  esquive une analyse détaillée du déroulement du calcul
- conceptuelle : détour par des *atomes* (alphabets infinis)
  + théorème (technique) de désatomisation

Pour l'instant seule la seconde approche s'étend à :

Théorème

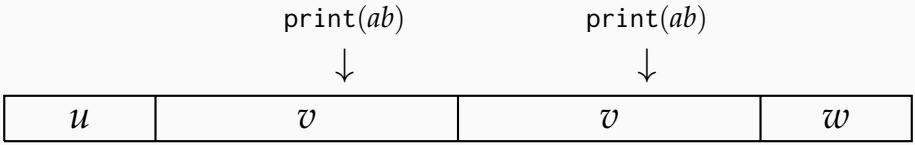
Pour tout $k \geq 2$ il existe f polyrég. nécessitant k jetons telle que $|f(w)| = O(|w|^2)$.

Minimisation de jetons : vague idée du cas 0-aire [Lhote 2020]

Soit un transducteur bidirectionnel, calculant $f : \Gamma^* \to \Sigma^*$ régulière.

On cherche à « pomper » des triplets $(u, v, w) \in (\Gamma^*)^3$ particuliers :
« $\forall n, m \in \mathbb{N}$ le transducteur se comporte pareil dans le facteur v de $(uv^m)v(v^nw)$ »

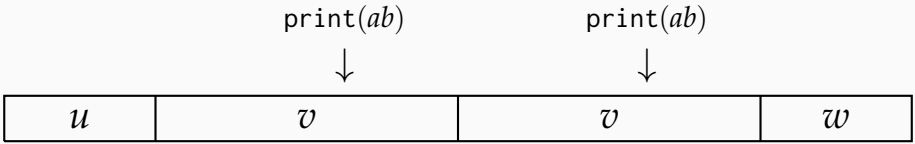
pour les trouver : monoïde fini de comportements + théorème de Ramsey



Minimisation de jetons : vague idée du cas 0-aire [Lhote 2020]

Soit un transducteur bidirectionnel, calculant $f : \Gamma^* \to \Sigma^*$ régulière.

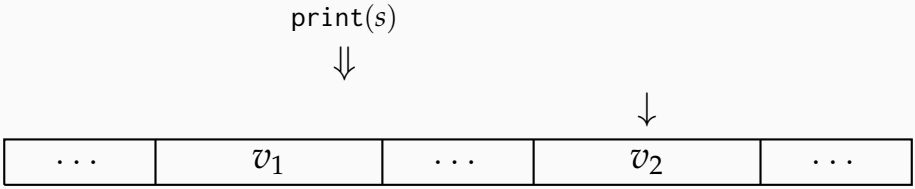
On cherche à « pomper » des triplets $(u, v, w) \in (\Gamma^*)^3$ particuliers :
« $\forall n, m \in \mathbb{N}$ le transducteur se comporte pareil dans le facteur v de $(uv^m)v(v^nw)$ »
pour les trouver : monoïde fini de comportements + théorème de Ramsey



- ci-dessus : $|f(uv^nw)| \geq n|ab| = 2n$
- si aucun triplet n'est « productif » : $|f(w)| = O(1)$

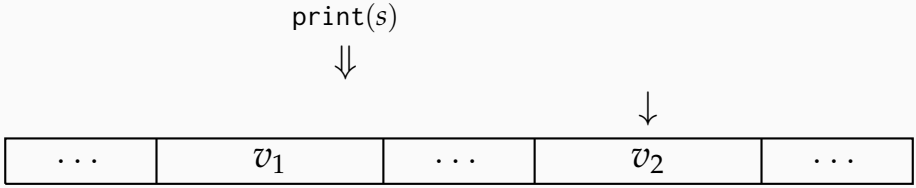
Minimisation de jetons : vague idée (suite)

Soit un transducteur à 2 jetons, calculant $f : \Gamma^* \to \Sigma^*$ polyrégulière.
On cherche à pomper des motifs (avec condition de comportement contextuel ?) :



Minimisation de jetons : vague idée (suite)

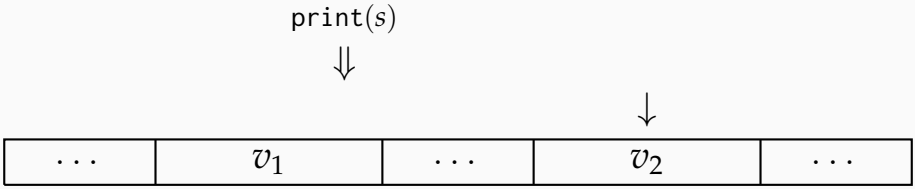
Soit un transducteur à 2 jetons, calculant $f : \Gamma^* \to \Sigma^*$ polyrégulière.
On cherche à pomper des motifs (avec condition de comportement contextuel ?) :



- ci-dessus : $s \neq \varepsilon \implies |f(\dots (v_1)^n \dots (v_2)^m \dots)| \geq n \times m$
- Si pas de tel 5-uplet productif : $|f(w)| = O(|w|)$??

Minimisation de jetons : vague idée (suite)

Soit un transducteur à 2 jetons, calculant $f : \Gamma^* \to \Sigma^*$ polyrégulière.
On cherche à pomper des motifs (avec condition de comportement contextuel ?) :



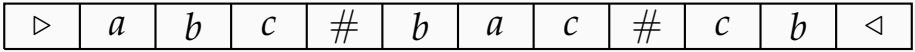
- ci-dessus : $s \neq \varepsilon \implies |f(\dots(v_1)^n \dots (v_2)^m \dots)| \geq n \times m$
- Si pas de tel 5-uplet productif : $|f(w)| = O(|w|)$??

  Fonctionne (non-trivialement) dans le cas *sans comparaison*

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

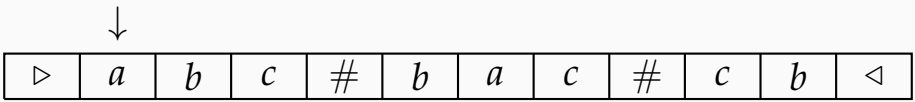


Sortie :

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

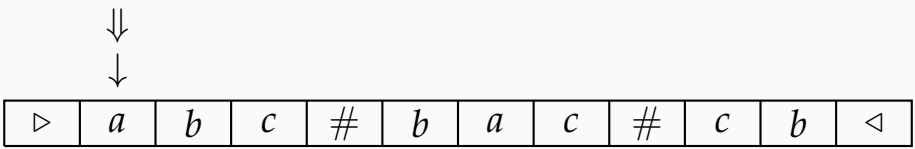


Sortie :

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

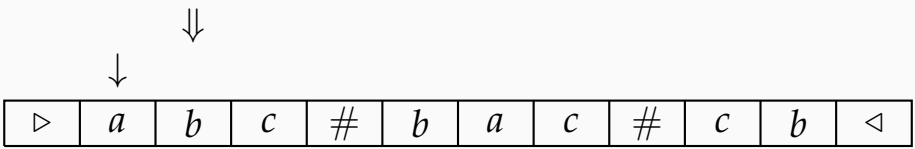


Sortie :

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

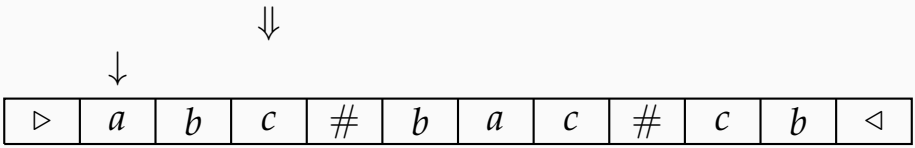


Sortie : a

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

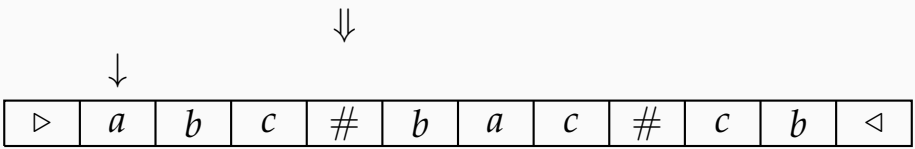


Sortie : ab

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

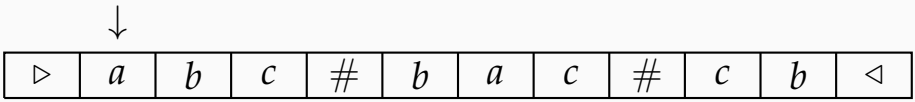


Sortie : *abc*

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

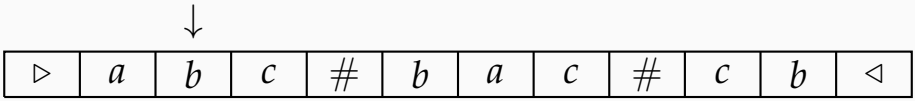


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

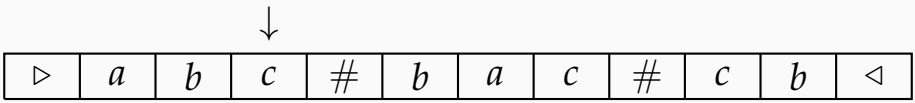


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

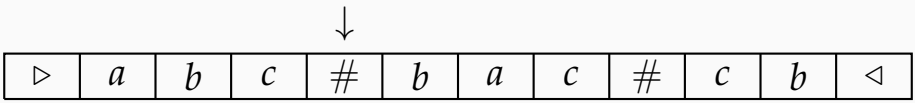


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

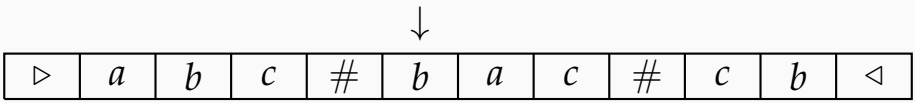


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

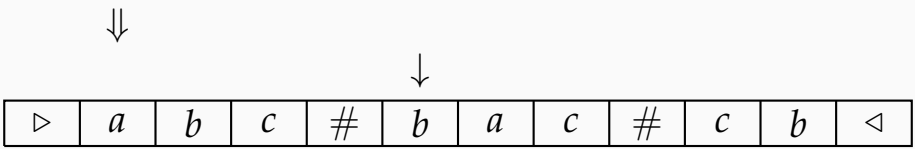


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

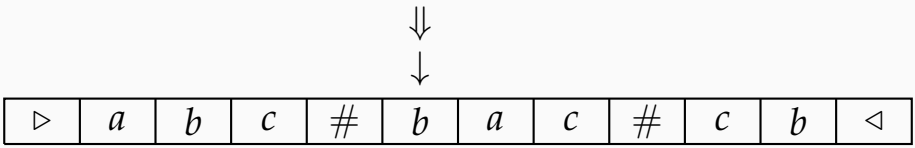


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

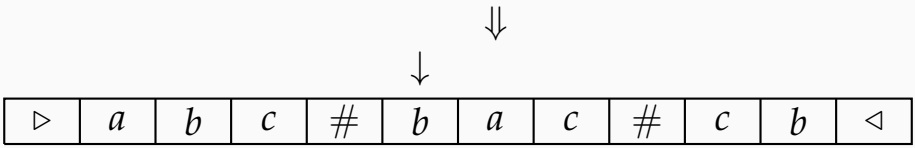


Sortie : $abc\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

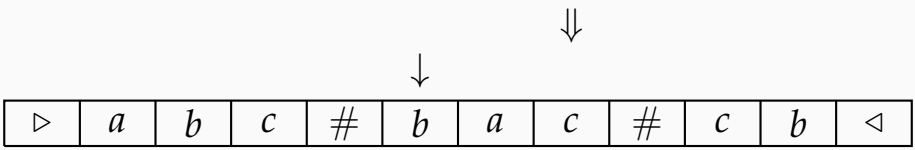


Sortie : $abc\#b$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

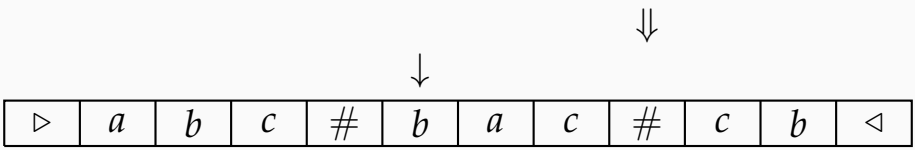


Sortie : $abc\#ba$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

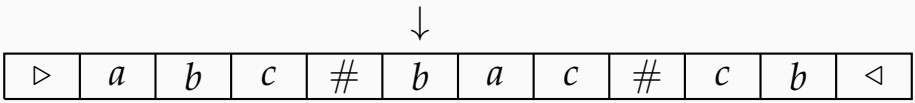


Sortie : $abc\#bac$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

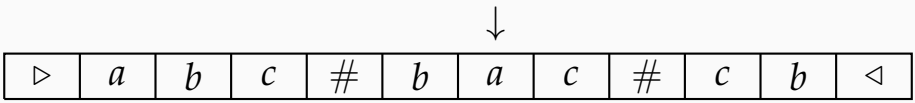


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

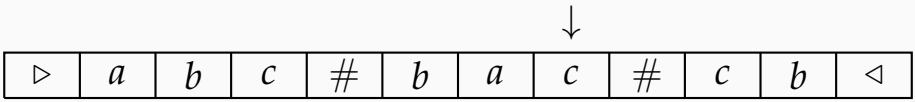


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

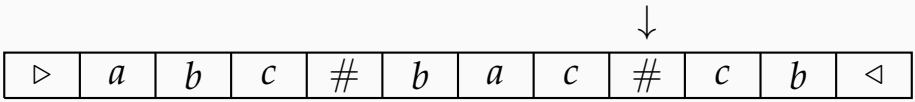


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

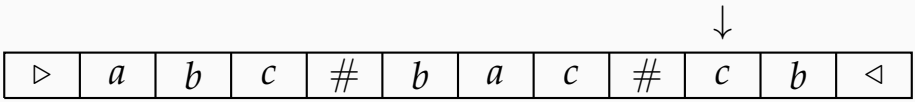


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

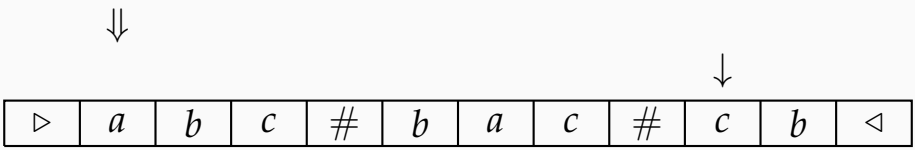


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

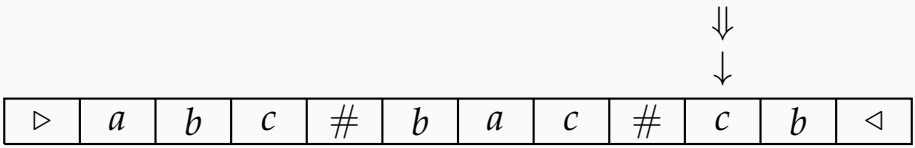


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

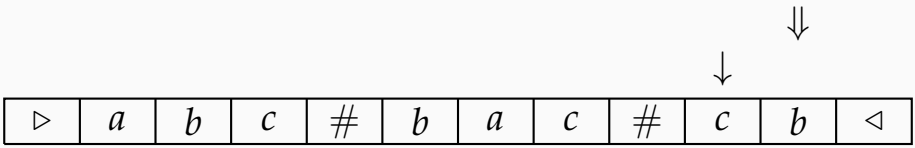


Sortie : $abc\#bac\#$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

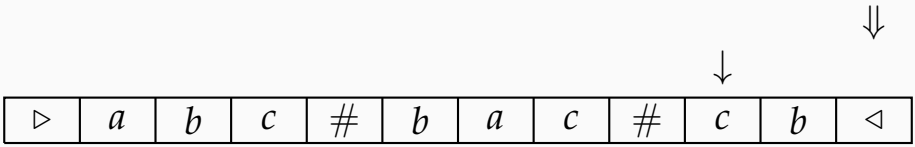


Sortie : $abc\#bac\#c$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

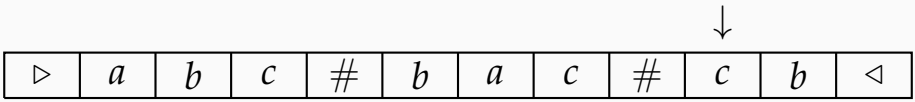


Sortie : $abc\#bac\#cb$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

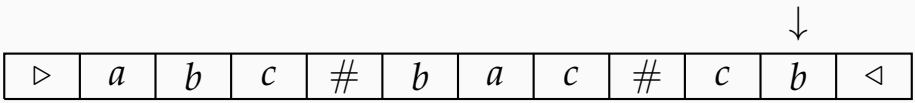


Sortie : $abc\#bac\#cb$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

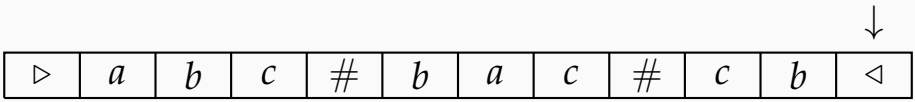


Sortie : $abc\#bac\#cb$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)

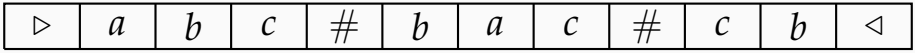


Sortie : $abc\#bac\#cb$

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)



Sortie : $abc\#bac\#cb$

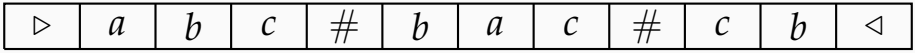
Comparaisons $\Longrightarrow$ *dépendances* possibles entre positions atteintes par les jetons

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

Minimisation de jetons : obstruction avec comparaison

Fonction identité calculée bloc par bloc avec 2 jetons

(c'est `mapPower` sans le jeton du milieu qui compte le $(-)^n$, car $\texttt{map}(\texttt{id}) = \texttt{id}$)



Sortie : $abc\#bac\#cb$

Comparaisons $\Longrightarrow$ *dépendances* possibles entre positions atteintes par les jetons

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

Quand on écrit en sortie, *la position de* $\Downarrow$ *détermine celle de* $\downarrow$: info redondante

Discussion sur les dépendances

De même, sur le transducteur du début pour `mapPower` à 3 jetons $\downarrow$, $\Downarrow$, $\triangledown$:

- quand on écrit en sortie, *la position de* $\triangledown$ *détermine celle de* $\downarrow$
- redondant mais nécessaire car pour bouger $\Downarrow$ il faut *dépiler* $\triangledown$

Discussion sur les dépendances

De même, sur le transducteur du début pour `mapPower` à 3 jetons $\downarrow$, $\Downarrow$, $\triangledown$:

- quand on écrit en sortie, *la position de* $\triangledown$ *détermine celle de* $\downarrow$
- redondant mais nécessaire car pour bouger $\Downarrow$ il faut *dépiler* $\triangledown$

On voudrait un formalisme plus flexible où on n'aurait besoin que de 2 « pointeurs sur l'entrée » $\longrightarrow$ les interprétations MSO

Rappel : formules en logique Monadique du Second Ordre

$\varphi(x_1, \ldots, x_n)$, les x_i sont des *positions* d'un mot w $(1 \leq x_i \leq |w|)$

- $n = 0 : L \subseteq \Gamma^*$ est rationnel $\iff \exists \varphi.\ L = \{w \in \Sigma^* \mid w \vDash \varphi\}$
- $n \geq 1$: généralisation aux mots à n positions marquées

Transductions MSO à la Courcelle

Rappel : formules en logique Monadique du Second Ordre

$\varphi(x_1, \ldots, x_n)$, les x_i sont des *positions* d'un mot w $(1 \leq x_i \leq |w|)$

- $n = 0 : L \subseteq \Gamma^*$ est rationnel $\iff \exists \varphi.\ L = \{w \in \Sigma^* \mid w \vDash \varphi\}$
- $n \geq 1$: généralisation aux mots à n positions marquées

Transductions MSO $\Gamma^* \to \Sigma^*$: choix d'un ensemble fini I et de formules MSO

$$\varphi_a^i(x) \text{ pour } a \in \Sigma \qquad \varphi_\prec^{i,j}(x, y) \qquad i, j \in I$$

Pour un mot d'entrée $w \in \Gamma^*$, définit des relations unaires $a(z)$
et une binaire $z \prec z'$ sur $z, z' \in I \times \{1, \ldots, |w|\}$

Transductions MSO à la Courcelle

Rappel : formules en logique Monadique du Second Ordre

$\varphi(x_1, \ldots, x_n)$, les x_i sont des *positions* d'un mot w $(1 \leq x_i \leq |w|)$

- $n = 0 : L \subseteq \Gamma^*$ est rationnel $\iff \exists \varphi.\ L = \{w \in \Sigma^* \mid w \vDash \varphi\}$
- $n \geq 1$: généralisation aux mots à n positions marquées

Transductions MSO $\Gamma^* \to \Sigma^*$: choix d'un ensemble fini I et de formules MSO

$$\varphi_a^i(x) \text{ pour } a \in \Sigma \qquad \varphi_{\prec}^{i,j}(x, y) \qquad i, j \in I$$

Pour un mot d'entrée $w \in \Gamma^*$, définit des relations unaires $a(z)$
et une binaire $z \prec z'$ sur $z, z' \in I \times \{1, \ldots, |w|\}$

si on a de la chance, c'est isomorphe à un mot de sortie $f(w) \in \Sigma^*$

Transductions MSO à la Courcelle

Transductions MSO $\Gamma^* \to \Sigma^*$: choix d'un ensemble fini I et de formules MSO

$$\varphi_a^i(x) \text{ pour } a \in \Sigma \qquad \varphi_\prec^{i,j}(x,y) \qquad i,j \in I$$

Pour un mot d'entrée $w \in \Gamma^*$, définit des relations unaires $a(z)$
et une binaire $z \prec z'$ sur $z, z' \in I \times \{1, \dots, |w|\}$
si on a de la chance, c'est isomorphe à un mot de sortie $f(w) \in \Sigma^*$

Théorème [Engelfriet & Hoogeboom 2001]

Transductions MSO sur les mots = fonctions régulières (rappel : à 1 jeton)

(Pas trop dur à prouver si on sait que les fonctions rég. sont stables par composition)

Exemple : `reverse` définissable par $|I| = 1$, $\varphi_\prec(x,y) = (y < x)$

Interprétations MSO en dimension supérieure

Interprétation MSO $\Gamma^* \to \Sigma^*$: choix de $k \in \mathbb{N}$, d'un ens. fini I et de formules MSO

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

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

si on a de la chance, c'est isomorphe à un mot de sortie $f(w) \in \Sigma^*$

Interprétations MSO en dimension supérieure

Interprétation MSO $\Gamma^* \to \Sigma^*$: choix de $k \in \mathbb{N}$, d'un ens. fini I et de formules MSO

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

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

si on a de la chance, c'est isomorphe à un mot de sortie $f(w) \in \Sigma^*$

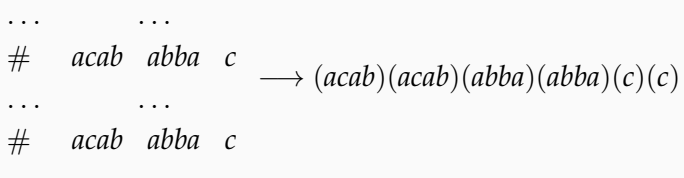
Théorème [Bojańczyk, Kiefer & Lhote 2019]

Interprétations MSO sur les mots = fonctions polyrégulières

(Démonstration très technique utilisant la théorie des modèles finis)

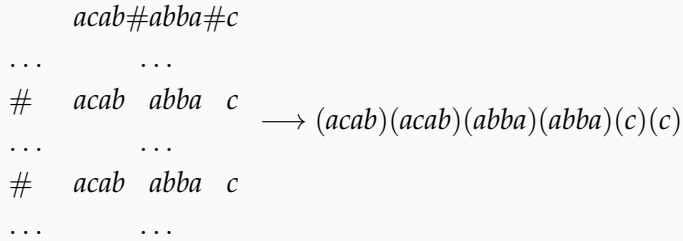
Exemple d'interprétation MSO

$\texttt{mapPower}'$: $w_0\#\ldots\#w_n \longmapsto (w_0)^n\ldots(w_n)^n$ a une interp. de dim 2 (optimale) :



Exemple d'interprétation MSO

`mapPower′` : $w_0\#\ldots\#w_n \longmapsto (w_0)^n\ldots(w_n)^n$ a une interp. de dim 2 (optimale) :



- $\varphi_a(x_1, x_2) = a(x_1) \wedge \#(x_2)$ (on omet les indices de copie : $|I| = 1$)
- $\varphi_{\prec}(x_1, x_2, y_1, y_2) = \exists x_3, y_3.$ qui sont les débuts des blocs contenant resp. x_1, y_1 et $(x_3, x_2, x_1) < (y_3, y_2, y_1)$ lexico $\longrightarrow$ jetons $\downarrow, \Downarrow, \triangledown$

Minimisation de dimension

Théorème (nouveau !) [Bojańczyk, Douéneau-Tabot, Kiefer, N. & Pradic]

Interprétations MSO sur les mots *de dim. k* = fonctions polyrég. *à croissance $O(n^k)$*

Corollaire : polyrégulière + $O(n) \implies$ régulière (déjà établi différemment)

Minimisation de dimension

Théorème (nouveau !) [Bojańczyk, Douéneau-Tabot, Kiefer, N. & Pradic]

Interprétations MSO sur les mots *de dim. k* = fonctions polyrég. *à croissance* $O(n^k)$

Corollaire : polyrégulière + $O(n) \implies$ régulière (déjà établi différemment)

Lemme principal : reflète le phénomène de dépendances entre positions

Toute formule MSO $\varphi(x_1, \ldots, x_n)$ est équivalente à $\bigvee_{i \in I} \varphi_i$ (I fini) où $\forall i \in I$,

- $\exists Y_i \subseteq \{1, \ldots, n\}$ tel que pour tout mot marqué, lorsque φ_i est vraie, les valeurs des x_m pour $m \in Y_i$ déterminent toutes les autres
- de plus $\max\{\text{nb de tuples de pos. dans } w \text{ validant } \varphi_i \mid |w| = n\} = \Theta(n^{|Y_i|})$

Outils de démonstration : compositionnalité de MSO + forêts de factorisation

Cas particuliers pour la minimisation de jetons

La minimisation de jetons échoue en général. Elle marche :

- dans le cas sans comparaison

Cas particuliers pour la minimisation de jetons

La minimisation de jetons échoue en général. Elle marche :

- dans le cas sans comparaison
- pour les sorties unaires ($\{a\}^* \cong \mathbb{N}$), en utilisant : [Douéneau-Tabot 2021]
  polyrégulière $\Sigma^* \to \mathbb{N}$ $\iff$ automate $\mathbb{N}$-pondéré à valeurs $|w|^{O(1)}$
  $\iff$ automate $\mathbb{N}$-pondéré « stratifié » (layered)

(en fait, la minimisation de strates remonte à Schützenberger années 1960)

Cas particuliers pour la minimisation de jetons

La minimisation de jetons échoue en général. Elle marche :

- dans le cas sans comparaison
- pour les sorties unaires ($\{a\}^* \cong \mathbb{N}$), en utilisant : [Douéneau-Tabot 2021]
  polyrégulière $\Sigma^* \to \mathbb{N} \iff$ automate $\mathbb{N}$-pondéré à valeurs $|w|^{O(1)}$
  $\iff$ automate $\mathbb{N}$-pondéré « stratifié » (layered)
  (en fait, la minimisation de strates remonte à Schützenberger années 1960)
- plus généralement, pour les HDT0L à croissance polynomiale
  ($\iff$ transducteurs à registres stratifiés [Douéneau-Tabot, Filiot & Gastin 2020])

Cas particuliers pour la minimisation de jetons

La minimisation de jetons échoue en général. Elle marche :

- dans le cas sans comparaison
- pour les sorties unaires ($\{a\}^* \cong \mathbb{N}$), en utilisant : [Douéneau-Tabot 2021]
  polyrégulière $\Sigma^* \to \mathbb{N}$ $\iff$ automate $\mathbb{N}$-pondéré à valeurs $|w|^{O(1)}$
  $\iff$ automate $\mathbb{N}$-pondéré « stratifié » (layered)
  (en fait, la minimisation de strates remonte à Schützenberger années 1960)
- plus généralement, pour les HDT0L à croissance polynomiale
  ($\iff$ transducteurs à registres stratifiés [Douéneau-Tabot, Filiot & Gastin 2020])
- pour les entrées unaires ? work-in-progress (N. & Pradic)
  idée : obtenir des expressions explicites des suites de mots polyrégulières
  déjà fait dans le cas sans comp. (ICALP'21) pour prouver des séparations

Conclusion

Fonctions (poly)régulières : classes de transductions avec de nombreuses caractérisations (automates, logiques, composition de fonctions de base)

Contributions présentées dans cet exposé

- Fonctions polyrégulières *sans comparaison*
- *Croissance* des fonctions polyrégulières vs usage de « ressources »

Conclusion

Fonctions (poly)régulières : classes de transductions avec de nombreuses caractérisations (automates, logiques, composition de fonctions de base)

Contributions présentées dans cet exposé

- Fonctions polyrégulières *sans comparaison*
- *Croissance* des fonctions polyrégulières vs usage de « ressources »

- Logique pour les polyrég. sans comp. ? `mapPower` réfute une 1ère tentative
- Lemme principal pour la minimisation de dimension MSO
  $\longrightarrow$ vers une sémantique à origines pour les polyrégulières ?
- Apériodicité (note : $n(n+1)/2$ sans comp. et FO-polyrég. mais pas FO-polyrég-sans-comp)
- Problèmes de décision, par ex. appartenance [Douéneau-Tabot 2022]

Conclusion (Merci de votre attention !)

Fonctions (poly)régulières : classes de transductions avec de nombreuses caractérisations (automates, logiques, composition de fonctions de base)

Contributions présentées dans cet exposé

- Fonctions polyrégulières *sans comparaison*
- *Croissance* des fonctions polyrégulières vs usage de « ressources »

- Logique pour les polyrég. sans comp. ? `mapPower` réfute une 1ère tentative
- Lemme principal pour la minimisation de dimension MSO
  $\longrightarrow$ vers une sémantique à origines pour les polyrégulières ?
- Apériodicité (note : $n(n+1)/2$ sans comp. et FO-polyrég. mais pas FO-polyrég-sans-comp)
- Problèmes de décision, par ex. appartenance [Douéneau-Tabot 2022]