Automates a n-piles

David Gontier

ENS

8 janvier 2009

Qu'est ce qu'un automate a $n-$ *piles* ?

- Automates $\longrightarrow$ langages rationnels

Qu'est ce qu'un automate a $n-$ *piles* ?

- Automates $\longrightarrow$ langages rationnels
- Automates à pile $\longrightarrow$ grammaire algébrique

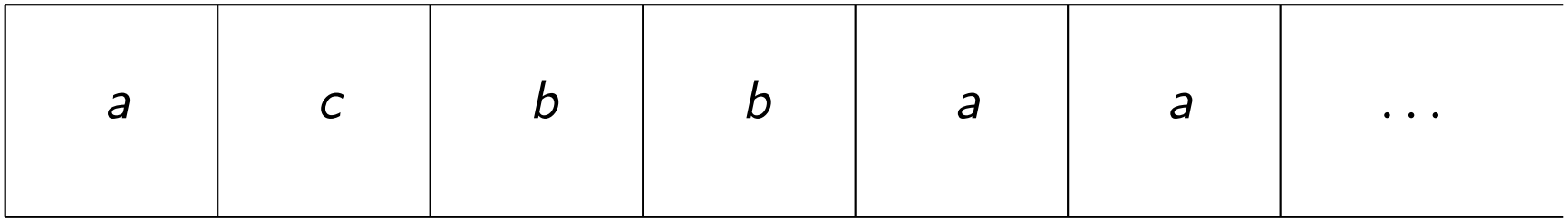
Qu'est ce qu'un automate a $n-$ *piles* ?

- Automates $\longrightarrow$ langages rationnels
- Automates à pile $\longrightarrow$ grammaire algébrique
- Automates à 2 piles $\longrightarrow$ équivalent à une machine de Turing

Qu'est ce qu'un automate a $n-$ *piles* ?

- Automates $\longrightarrow$ langages rationnels
- Automates à pile $\longrightarrow$ grammaire algébrique
- Automates à 2 piles $\longrightarrow$ équivalent à une machine de Turing

Et si on n'autorise qu'une seule pile, qui elle même contient des piles ?



$\omega = acbbaa...$

Trois instructions sont possibles sur ces piles :

- lecture du sommet de pile
- ajout d'une lettre au sommet de la pile
- suppression du sommet de pile

En particulier, on se s'occupe que du sommet de la pile...

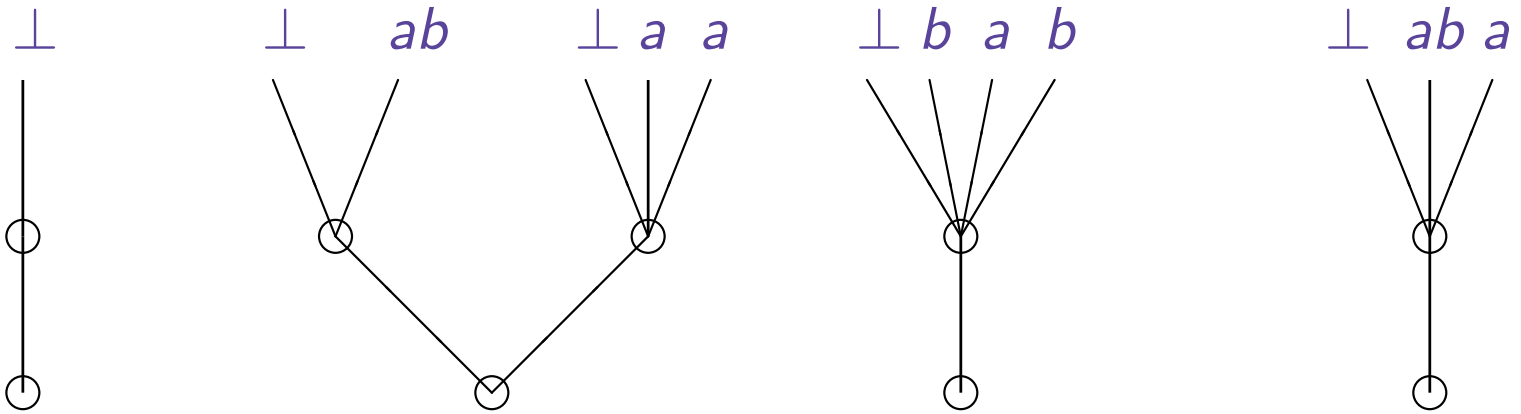
Definition

On définit récursivement une $n-$*pile* de la manière suivante :

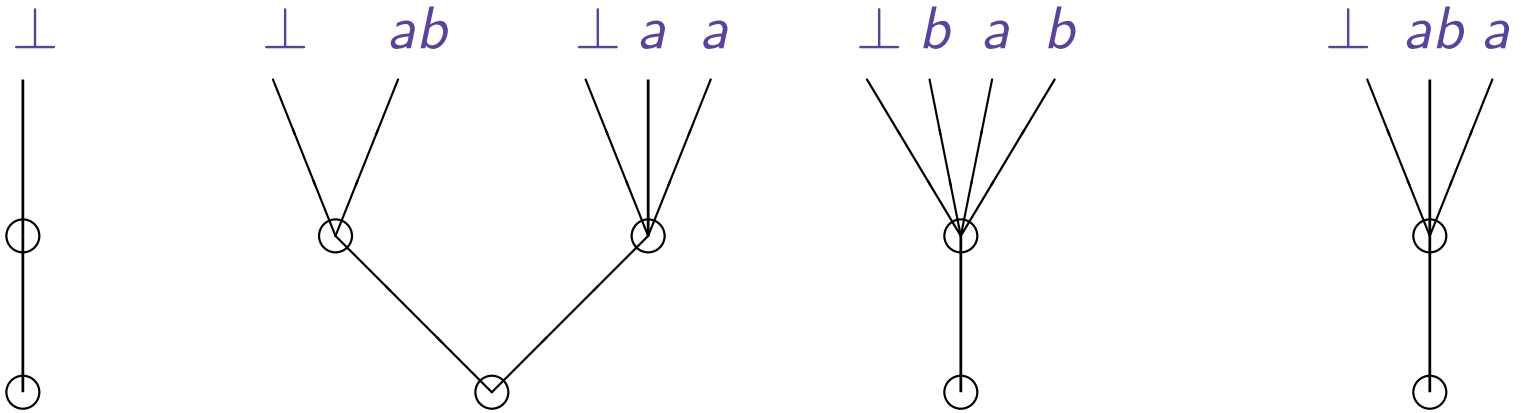
- Une $1-$*pile* est une suite finie $[a_1, \ldots, a_n]$ de mots de A^*
- Une $(n+1)-$*pile* est une suite finie $[p_1, \ldots, p_l]$ de $n-$*piles*

On note $\perp_k$ la $k-$*pile* vide

Exemple : $[[[\ \perp\ ]], [[\ \perp, ab], [\ \perp, a, a]], [[\ \perp, b, a, b]], [[\ \perp, ab, a]]]$ est une $3-$ *pile*



Exemple : $[[[\ \perp\ ]], [[\ \perp, ab], [\ \perp, a, a]], [[\ \perp, b, a, b]], [[\ \perp, ab, a]]]$ est une $3-$*pile*



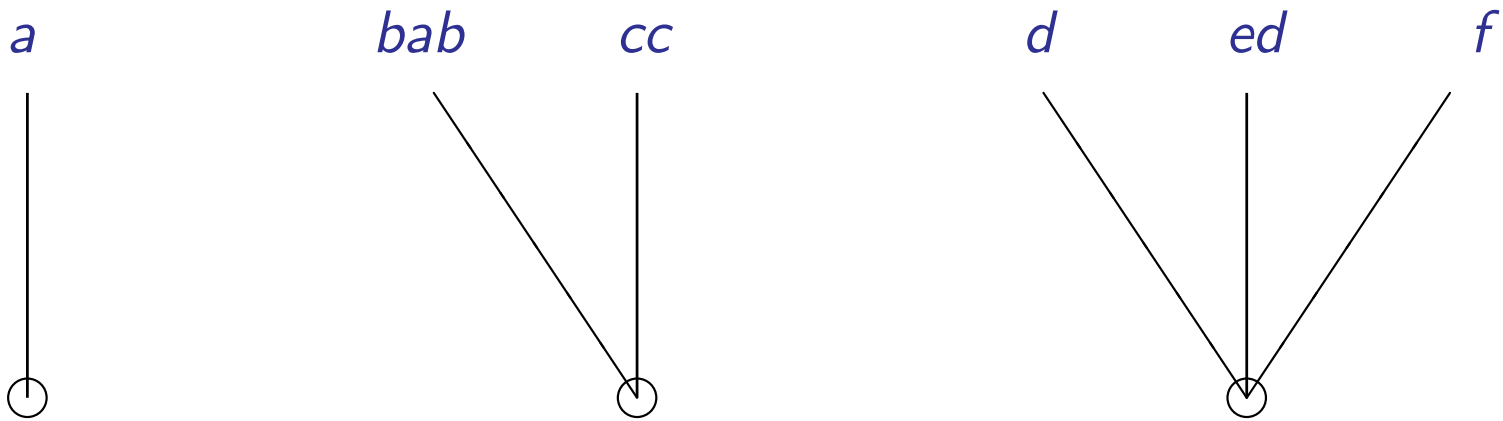
Dans toute la suite, on s'intéressera aux $2-$*piles*, et on appellera "pile 1" la $1-$*pile* accessible.

On s'autorise les opérations suivantes sur les $2-$ *piles*

- top : pour lire le sommet de la pile 1
- pop_1 : pour supprimer ce top
- pop_2 : pour supprimer pile 1.
- $push_1(a)$: ajouter a au sommet de la pile 1
- $push_2$: copier la pile 1 au sommet de la $2-$ *pile*

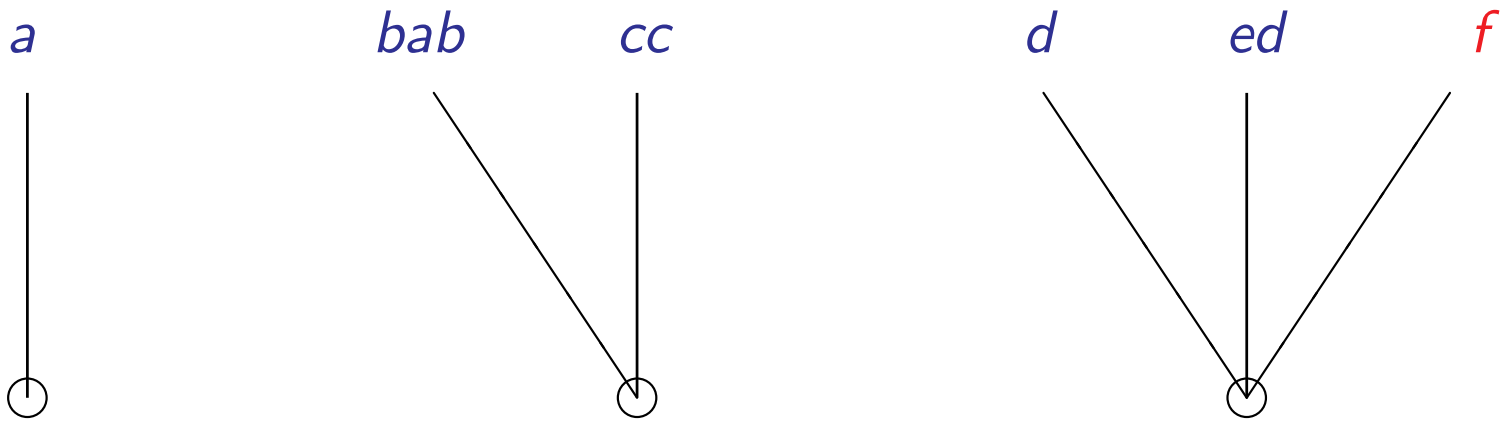
Exemple

$$\omega = [[a], [bab, cc], [d, ed, f]]$$

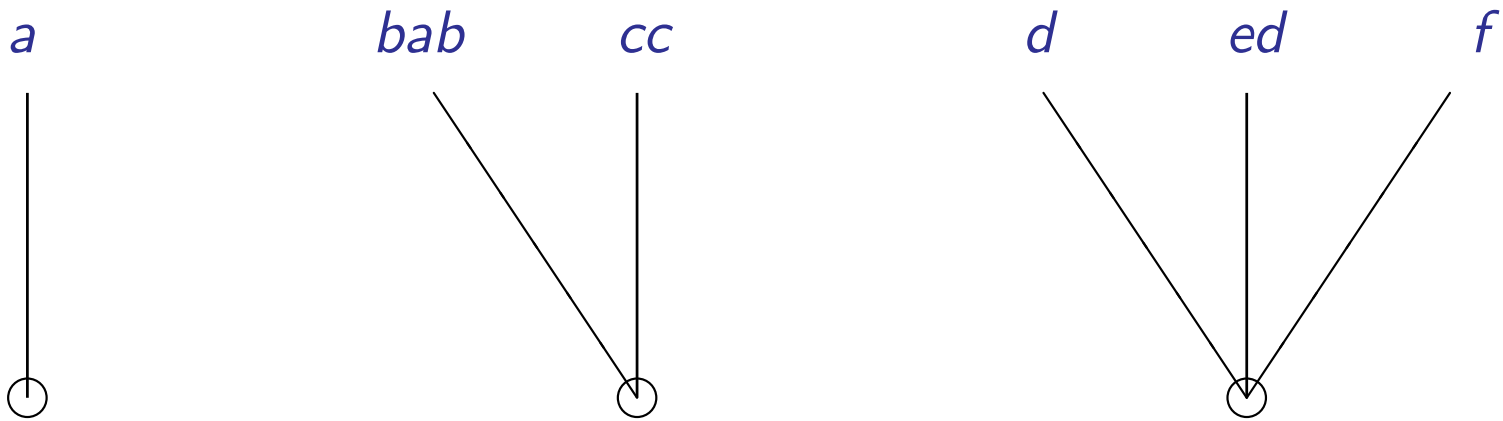


Exemple

$top = f$

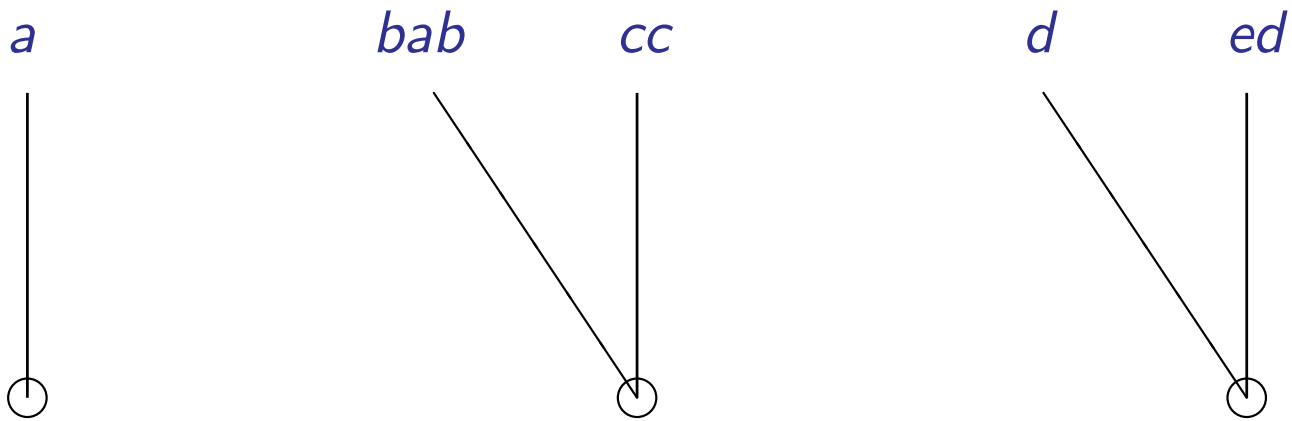


pop_1

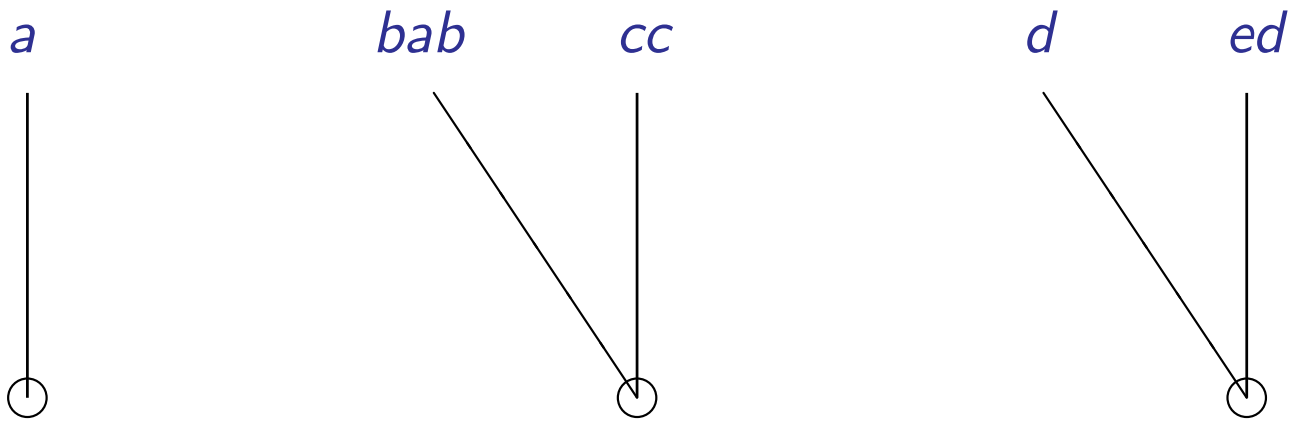


Exemple

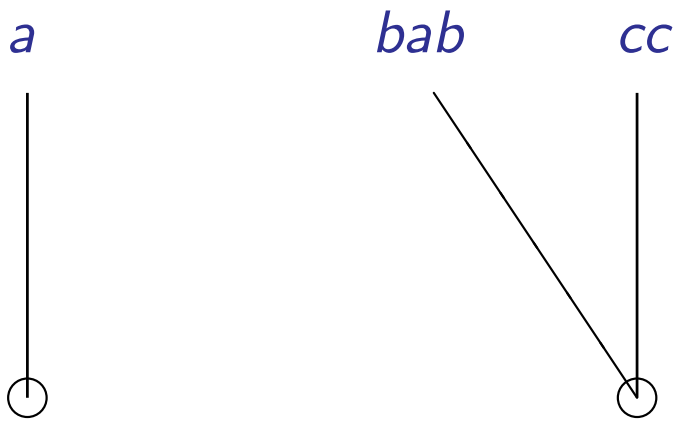
pop_1



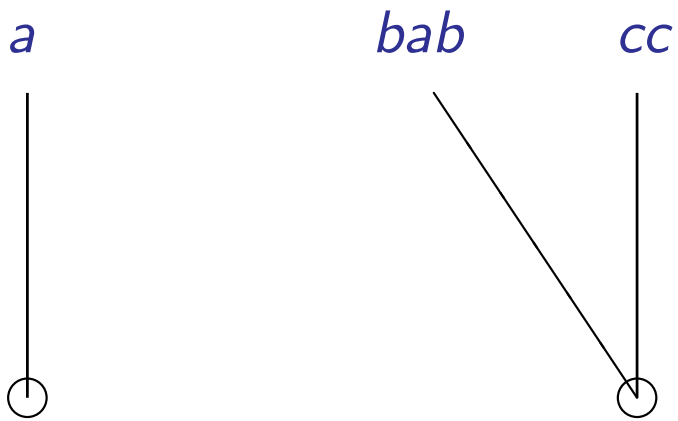
pop_2



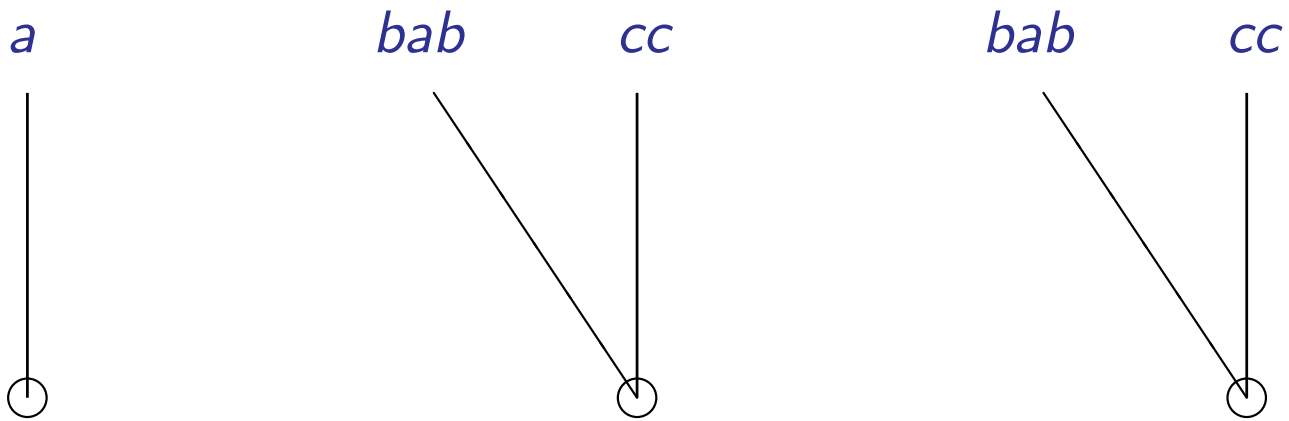
pop_2



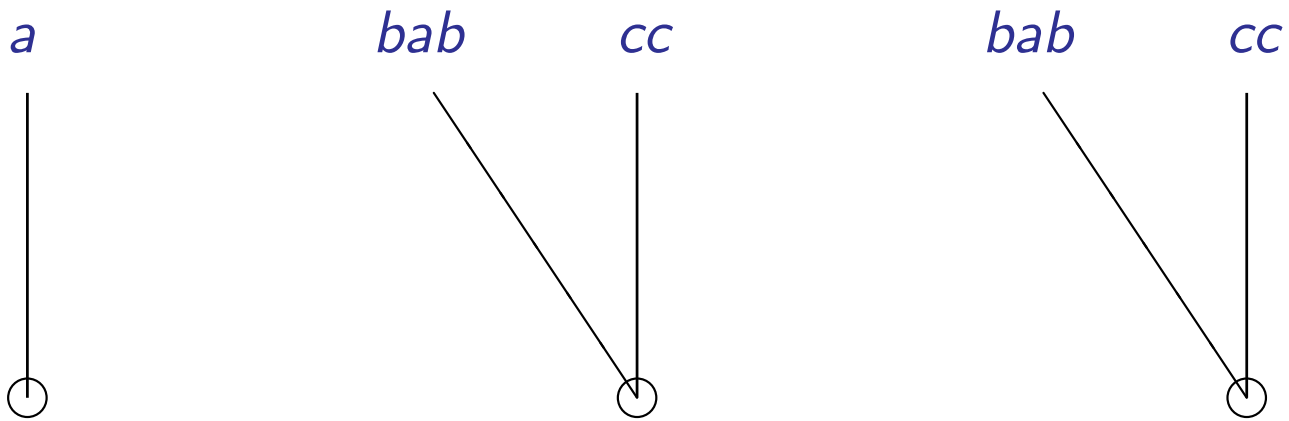
$push_2$



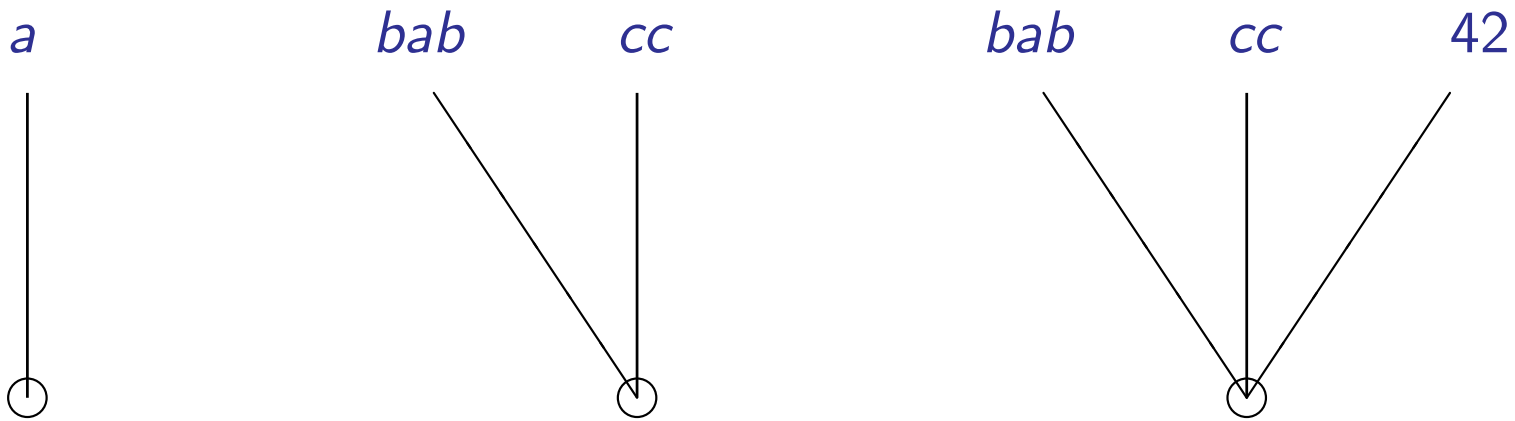
$push_2$



$push_1(42)$



$push_1(42)$



Definition

Un automate à $2-pile$ ($2-PDA$) est un automate "muni" d'une $2-pile$ P, et dont les transitions sont de la forme

$$\delta_\alpha(q, a) = (p, action)$$

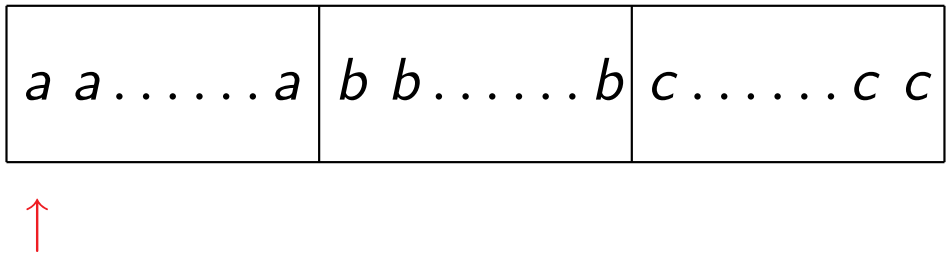
où $top(P) = a$, α est le symbole lu par l'automate, et où *action* est une ou plusieurs des opérations définies précédemment sur P.

On travaille avec le mode d'acceptation par pile vide

Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$

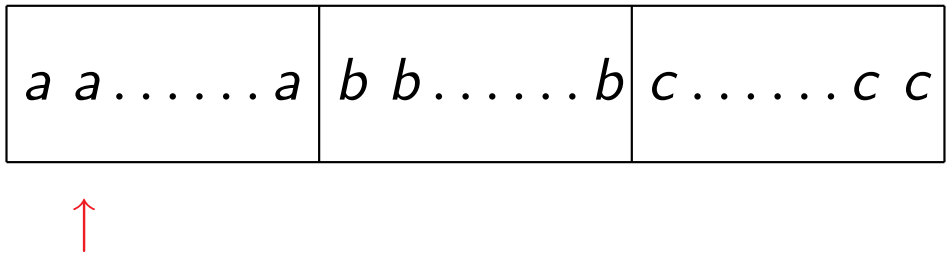


q_0,

Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

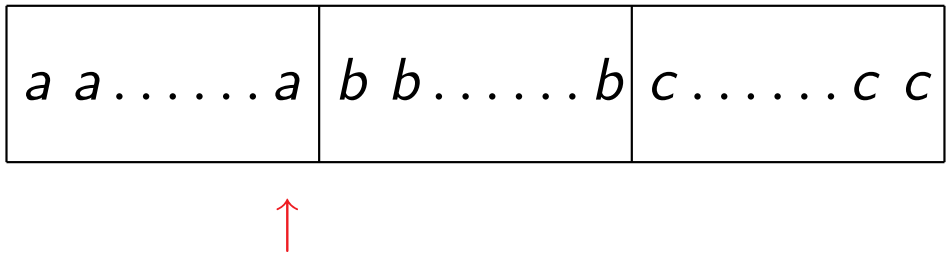
$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$

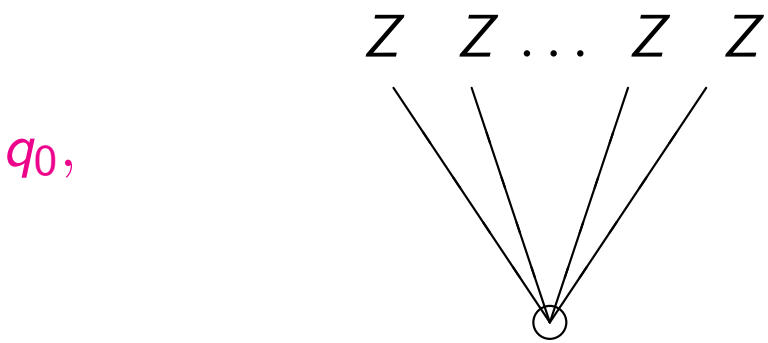


Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$

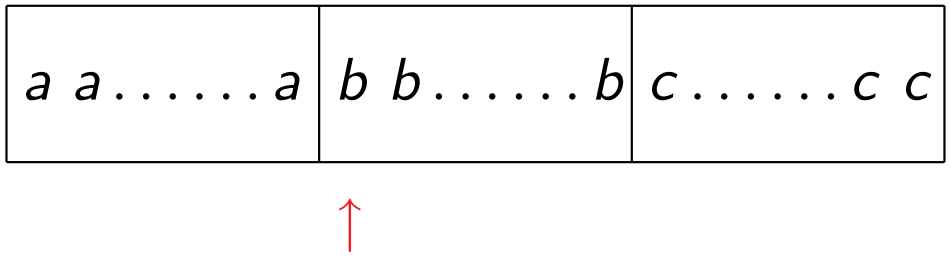




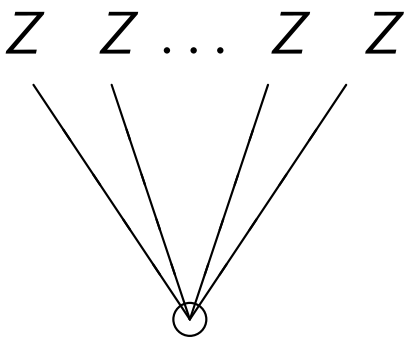
Exemple :

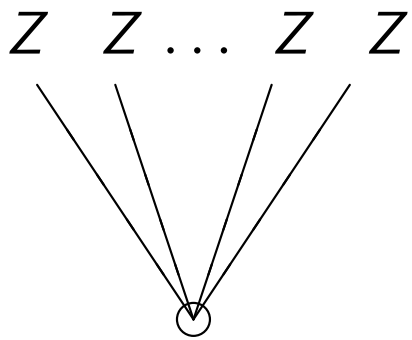
Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$



q_1,

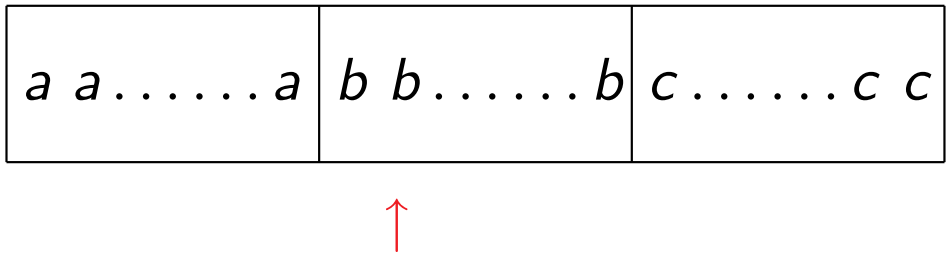




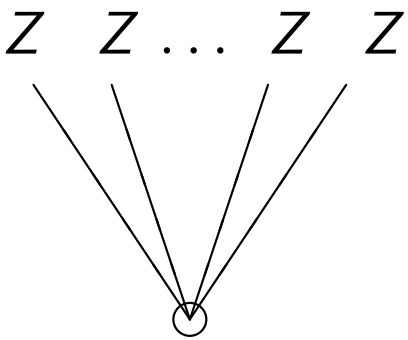
Exemple :

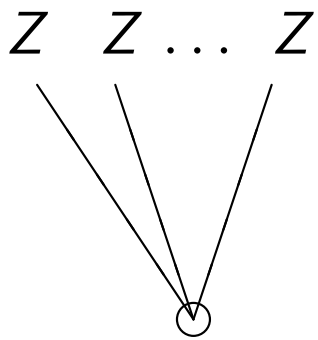
Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$



q_1,

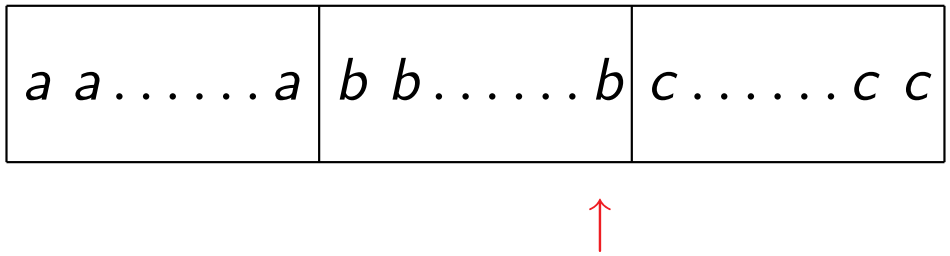


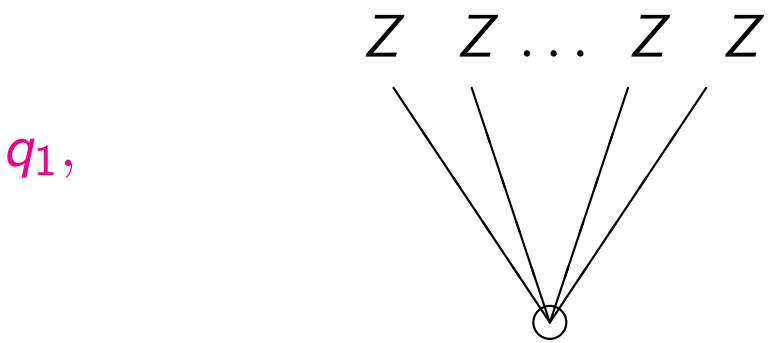


Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$

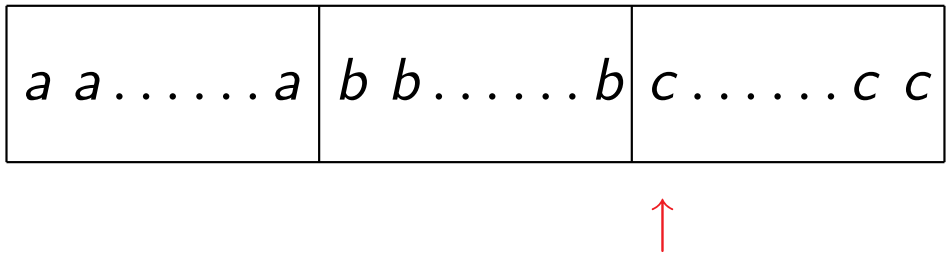


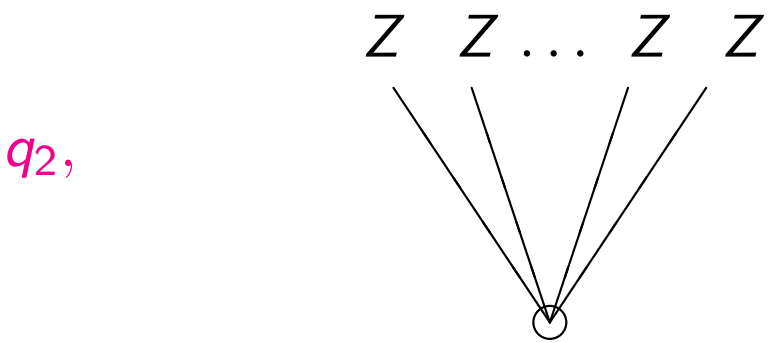


Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$

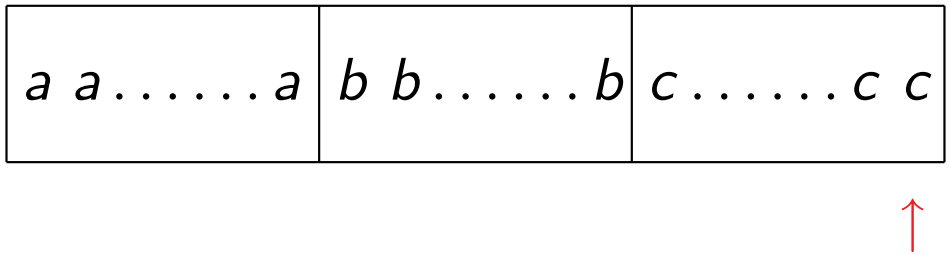




Exemple :

Le langage suivant reconnaît le langage $L = \{a^n b^n c^n, n \geq 0\}$:

$$
\begin{array}{lcl}
\delta_a(q_o, \perp_2) & = & (q_0, push_1(Z)) \\
\delta_a(q_o, Z) & = & (q_0, push_1(Z)) \\
\delta_b(q_0, Z) & = & (q_1, push_2 + pop_1) \\
\delta_b(q_1, Z) & = & (q_1, pop_1) \\
\delta_c(q_1, \perp_1) & = & (q_2, pop_2 + pop_1) \\
\delta_c(q_2, Z) & = & (q_2, pop_1)
\end{array}
$$



q_2,

Plus généralement, les automates à 2-piles peuvent reconnaître les langages polynomiaux :

Definition

Soit $\mathcal{A}$ un alphabet, un langage L est dit polynomial sur $\mathcal{A}$ si il existe $(\omega_{i,j})_{1\leq i\leq m, 1\leq j\leq n} \in \mathcal{A}$, tel que :

$$L = \left\{ \omega_{1,1}^{k_1} \omega_{1,2}^{k_1^2} \dots \omega_{1,n}^{k_1^n} \omega_{2,1}^{k_2} \dots \omega_{m,n}^{k_m^n} | k_1, \dots, k_m \in \mathbb{N}, \omega_{i,1} \neq \epsilon \right\}$$

Plus généralement, les automates à 2-piles peuvent reconnaître les langages polynomiaux :

Definition

Soit $\mathcal{A}$ un alphabet, un langage L est dit polynomial sur $\mathcal{A}$ si il existe $(\omega_{i,j})_{1\leq i\leq m, 1\leq j\leq n} \in \mathcal{A}$, tel que :

$$L = \left\{ \omega_{1,1}^{k_1} \omega_{1,2}^{k_1^2} \dots \omega_{1,n}^{k_1^n} \omega_{2,1}^{k_2} \dots \omega_{m,n}^{k_m^n} | k_1, \dots, k_m \in \mathbb{N}, \omega_{i,1} \neq \epsilon \right\}$$

Exemple :

Plus généralement, les automates à 2-piles peuvent reconnaître les langages polynomiaux :

Definition

Soit $\mathcal{A}$ un alphabet, un langage L est dit polynomial sur $\mathcal{A}$ si il existe $(\omega_{i,j})_{1\leq i\leq m, 1\leq j\leq n} \in \mathcal{A}$, tel que :

$$L = \left\{ \omega_{1,1}^{k_1} \omega_{1,2}^{k_1^2} \dots \omega_{1,n}^{k_1^n} \omega_{2,1}^{k_2} \dots \omega_{m,n}^{k_m^n} | k_1, \dots, k_m \in \mathbb{N}, \omega_{i,1} \neq \epsilon \right\}$$

Exemple :

- Le langage $\left\{ a^n b^{n^2}, n \in \mathbb{N} \right\}$ est polynomiale

Plus généralement, les automates à 2-piles peuvent reconnaître les langages polynomiaux :

Definition

Soit $\mathcal{A}$ un alphabet, un langage L est dit polynomial sur $\mathcal{A}$ si il existe $(\omega_{i,j})_{1\leq i\leq m, 1\leq j\leq n} \in \mathcal{A}$, tel que :

$$L = \left\{ \omega_{1,1}^{k_1} \omega_{1,2}^{k_1^2} \dots \omega_{1,n}^{k_1^n} \omega_{2,1}^{k_2} \dots \omega_{m,n}^{k_m^n} | k_1, \dots, k_m \in \mathbb{N}, \omega_{i,1} \neq \epsilon \right\}$$

Exemple :

- Le langage $\left\{ a^n b^{n^2}, n \in \mathbb{N} \right\}$ est polynomiale
- Le langage $\{a^n b^m c^{nm}, n, m \in \mathbb{N}\}$ ne l'est pas

Le langage de Urzyczyn

Un enfant reçoit dans l'ordre des objets numérotés de 1 à n, qu'il empile.

Cet enfant retire de temps en temps pendant le processus le premier objet du sommet de la pile.

Un enfant reçoit dans l'ordre des objets numérotés de 1 à n, qu'il empile.

Cet enfant retire de temps en temps pendant le processus le premier objet du sommet de la pile.

Quel est le numéro du jouet du dessus de la pile à la fin de l'opération ?

Definition

Le langage U de Urzyczyn est défini sur l'alphabet $\mathcal{A} = \{(,), *\}$ et consiste en l'ensemble des mots qui s'ecrivent sous la forme $w*^n$, tel que w soit le préfixe d'une expression bien parenthésée, et qu'aucun préfixe de w soit bien paranthésé. De plus, on donne une étiquette à chaque parenthèse de la manière suivante :

- L'étiquette du n^{eme} (est n
- L'étiquette de) est celle de l'étiquette de la parenthèse précédant celle du (qu'elle ferme.
- n est alors le numéro de la dernière étiquette

Exemple :

$$\underset{1}{(}\ \underset{2}{(}\ \underset{3}{(}\ \underset{4}{(}\ \underset{3}{)}\ \underset{2}{)}\ \underset{5}{(}\ \underset{6}{(}\ \underset{5}{)}\ \underset{7}{(}\ \underset{8}{(}\ \underset{7}{)}\ \underset{5}{)}\ * \ * \ * \ * \ *$$

- $\underset{k}{(}$ $\longrightarrow$ insérer l'objet k dans la pile
- $\underset{k}{)}$ $\longrightarrow$ retirer l'objet de sommet de pile. k est le numéro de l'objet au sommet apres l'operation
- $\underbrace{* \cdots *}_{n}$ $\longrightarrow$ juste pour coder le numéro du dernier objet

Theoreme

Tout mot de U s'écrit de manière unique sous la forme :

$$\underbrace{(}_{(1)}\underbrace{(\ldots(}_{(2)}\underbrace{(\ldots)\ldots(\ldots)}_{(2)}\underbrace{*\cdots*}_{(3)}$$

où :

- *(1) est le préfixe d'un mot bien parenthésé, finissant par (, dont aucun sous préfixe n'est bien paranthésé.*
- *(2) est une expression bien paranthésée.*
- *(3) a le même nombre n de * que le nombre de (dans (1).*

Theoreme

Tout mot de U s'écrit de manière unique sous la forme :

$$\underbrace{(}_{(1)}\underbrace{(\ldots(}_{(2)}(\ldots)\ldots(\ldots)\underbrace{*\cdots*}_{(3)}$$

où :

- *(1) est le préfixe d'un mot bien parenthésé, finissant par (, dont aucun sous préfixe n'est bien paranthésé.*
- *(2) est une expression bien paranthésée.*
- *(3) a le même nombre n de * que le nombre de (dans (1).*

Démonstration :

Theoreme

Tout mot de U s'écrit de manière unique sous la forme :

$$\underbrace{((\ldots((\ldots)\ldots(\ldots)}_{(1)\qquad\qquad(2)}\underbrace{*\cdots*}_{(3)}$$

où :

- *(1) est le préfixe d'un mot bien parenthésé, finissant par (, dont aucun sous préfixe n'est bien paranthésé.*
- *(2) est une expression bien paranthésée.*
- *(3) a le même nombre n de * que le nombre de (dans (1).*

Démonstration :

Comment ca les cubes ne tournent pas dans la salle ?

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

- La partie (1) du mot est bien parenthésée (pile 1)

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

- La partie (1) du mot est bien parenthésée (pile 1)
- On retient le nombre de (qu'on lit $\longrightarrow n$ (pile 2)

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

- La partie (1) du mot est bien parenthésée (pile 1)
- On retient le nombre de (qu'on lit $\longrightarrow n$ (pile 2)
- On vérifie que la partie (2) du mot est bien paranthésée (pile 1)

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à $2-$ piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

- La partie (1) du mot est bien parenthésée (pile 1)
- On retient le nombre de (qu'on lit $\longrightarrow n$ (pile 2)
- On vérifie que la partie (2) du mot est bien paranthésée (pile 1)
- On vérifie enfin qu'il y a bien n * dans la partie (3) du mot (pile 2)

Theoreme

Le langage de Urzyczyn est reconnaissable par Automate à 2 – piles non déterministe

Démonstration :

L'idée est de deviner la partie (1) du mot, et de vérifier les propriétés suivantes :

- La partie (1) du mot est bien parenthésée (pile 1)
- On retient le nombre de (qu'on lit $\longrightarrow n$ (pile 2)
- On vérifie que la partie (2) du mot est bien paranthésée (pile 1)
- On vérifie enfin qu'il y a bien n * dans la partie (3) du mot (pile 2)

Actuellement, on ne sait toujours pas si le langage de Urzyczyn est reconnaissable par un n-PDA déterministe...

- Les 2-PDA sont strictement plus puissants que les automates a pile
- Les n-PDA sont strictement moins puissants que les machines de Turing
- On n'a pas entièrement caractérisé la classe de langage reconnu par un 2-PDA
- Actuellement, on ne sait toujours pas si on peut déterminiser un 2-PDA

Remerciement

Je tiens a remercier Olivier Serre pour son aide précieuse