Automate à pile simple : exemple introductif

- (0 + 1) * (2 + 1)
- * + 0 1 + 2 1
- * + b | b + | | b | b
- Grammaire de ces expressions

$$G : \begin{cases} S \rightarrow +SS \;\; / \;\; *SS \;\; / \;\; | T \;\; / \;\; b \\ T \rightarrow | \; T \;\; / \;\; b \end{cases}$$

Quiz - le mot + b | b + | | b | b est engendré par G vrai faux

Analyseur descendant procédural

on a une chaîne de caractères à analyser de gauche à droite

- variable lu (renvoie le prochain caractère sans avancer, ce peut être EOF)
- procédure lire (avance sur le prochain caractère et met à jour la variable lu)
- fonction fin-de-chaîne? (teste lu = =EOF)
- procédure erreur(n) (signale une erreur et arrête l'analyse)

- une procédure par non-terminal
  - 2 procédures S et T (la procédure W avance dans la chaîne d'un mot f tel que W $\rightarrow^* f$)

- l'analyseur est un programme (déterministe ?)

  debut lire ; S ; si fin-de-chaîne? alors OK sinon PAS-OK **fin**

Analyse de 3 exemples

On va gérer à la main la pile des appels récursifs en empilant ce qui « reste à faire »

- * + | b b b $\longrightarrow$ succès
- * + | b b $\longrightarrow$ échec mot vide, pile non vide
- * + | b b b b $\longrightarrow$ échec mot non vide, pile vide

On va généraliser ce procédé en introduisant les

Automates à pile simples

- Un automate à pile simple est un quadruplet

  $a = < X , Y, y_0 , \lambda >$ où :

  - X est un alphabet, dit alphabet d'entrée

  - Y est un alphabet, dit alphabet de pile

  - $y_0 \in$ Y est le symbole initial de pile

  - $\lambda : X \times Y \rightarrow P_f (Y^*)$ est la fonction de transition de l'automate

    $\lambda(x, y)$ est un ensemble **fini** de mots de Y*

- on peut voir λ comme une partie finie de $(X \times Y) \times Y^*$

Exemple d'a.p.s.

- $X = \{ |, b, *, + \}$ $Y = \{ S, T \}$ $y_0 = S$

λ :

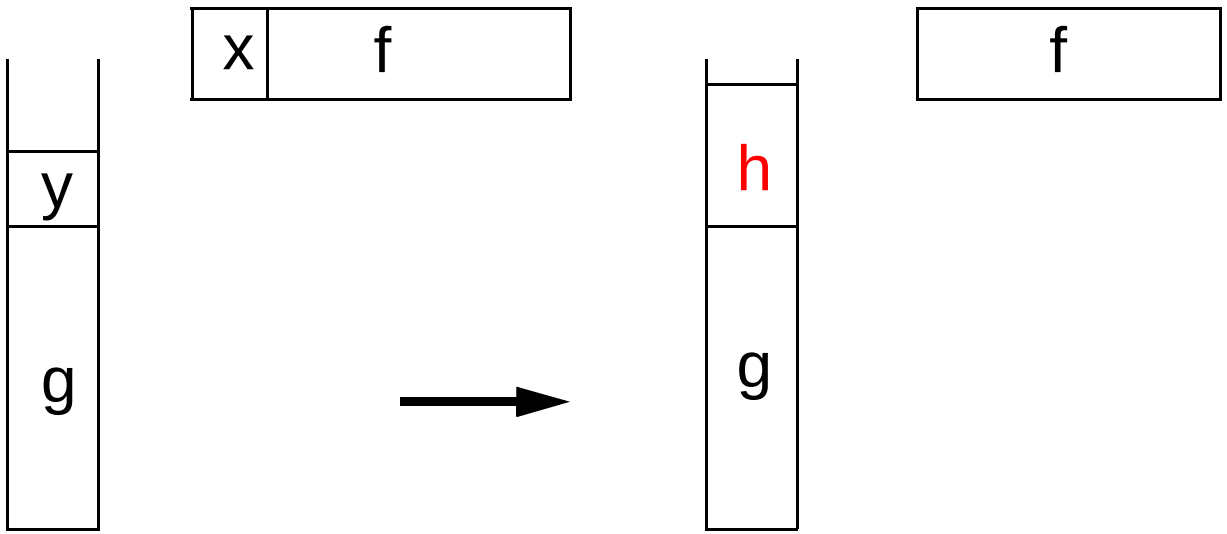
$*, S \rightarrow \{ SS \}$
$+, S \rightarrow \{ SS \}$
$|, S \rightarrow \{ T \}$
$b, S \rightarrow \{ \varepsilon \}$

$|, T \rightarrow \{ T \}$
$b, T \rightarrow \{ \varepsilon \}$

(ne pas noter) $\lambda(*, T) = \lambda(+, T) = \varnothing$

Transition entre deux configurations

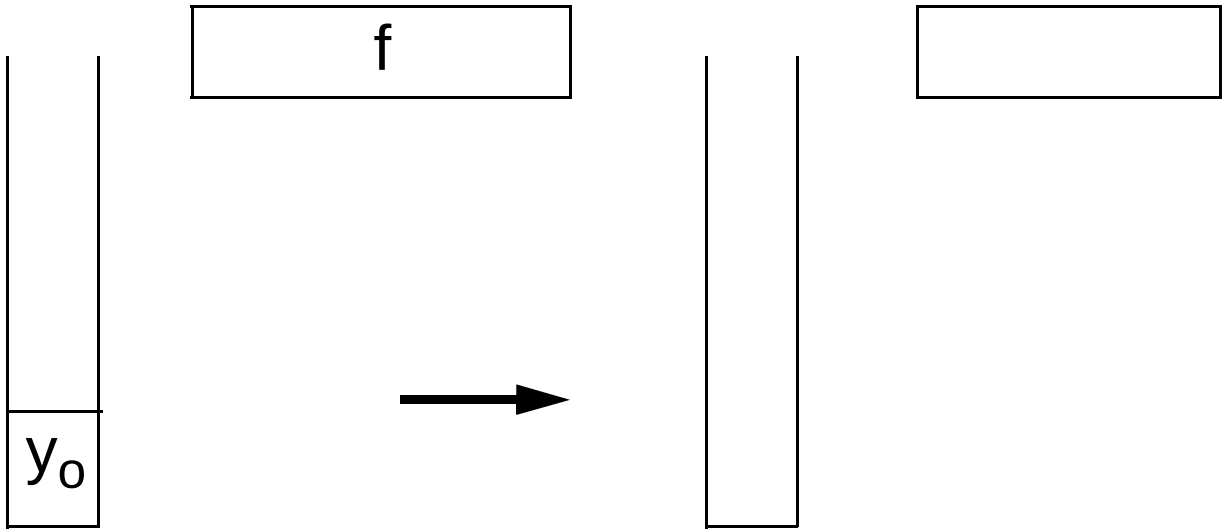
si $x \in X$, $y \in Y$ et $h \in \lambda(x, y)$



- **Calcul valide** = suite de configurations c_1, ... c_n avec des transitions de c_i à c_{i+1}

Langage reconnu par un a.p.s (définition 1)

- L(a) = { f $\in$ X* , il existe un calcul valide menant de la configuration (f, y_o) à la configuration $(1\!\!1_X , 1\!\!1_Y)$ }



Exemple de langage reconnu par a.p.s.

- $X = \{a, b\}$ $Y = \{y_0, y_1\}$

λ : $a , y_0 \rightarrow \{ y_0\, y_1 \}$ $a , y_1 \rightarrow \varnothing$ (pas de règle)

$b , y_0 \rightarrow \{ \varepsilon \}$

$b , y_1 \rightarrow \{ \varepsilon \}$

- on simule un calcul valide à partir de la configuration (a^2b^3, y_o)

Quiz - l'automate reconnaît $\{a^n\, b^{n+1} , n \geq 0\}$ $\{a^n\, b^{n+1} , n \geq 1\}$

Exemple (bis)

- (on confond les précédents symboles y_0 et y_1)

λ_1 : a , $y_0 \rightarrow$ $y_0\, y_0$

b , $y_0 \rightarrow \varepsilon$

Quiz - l'automate reconnaît $\{a^n\, b^{n+1}\ ,\ n \geq 0\}$ vrai faux

- on simule une suite de transitions à partir de la configuration (aba^2b^3, y_0)

Extension de l'application λ

- Soit $\lambda : X \times Y \rightarrow P_f(Y^*)$, on l'étend par induction en une application

  $\hat{\lambda} : X^* \times P_f(Y^*) \rightarrow P_f(Y^*)$

  de la façon suivante :

  $\hat{\lambda}(\varepsilon , L) = L$

  et pour $x \in X$, $m \in X^*$, $L \in P_f(Y^*)$,

  $\hat{\lambda}(x.m , L) = \hat{\lambda}(m , \bigcup_{y \in Y \text{ et } y.w \in L} \lambda(x,y).\{w\})$

  (remarquer que $\lambda(x,y).\{w\}$ est le produit de 2 langages finis)

Propriétés de $\hat{\lambda}$

- $\hat{\lambda}(x , \{y\}) = \hat{\lambda}(\varepsilon , \lambda(x,y).\{\varepsilon\}) = \lambda(x,y)$ car $y.\varepsilon$ est le seul mot de $\{y\}$
- $\hat{\lambda}(x , \{y.w\}) = \lambda(x,y).\{w\}$
- $\hat{\lambda}(m_1m_2 , L) = \hat{\lambda}(m_2, \hat{\lambda}(m_1 , L))$

- $\hat{\lambda}(f , L)$ est l'ensemble des contenus de pile auxquels on peut arriver en partant d'un quelconque contenu g de L (g est un mot de Y*) après avoir lu successivement les lettres de f ($f \in X^*$).
- $\hat{\lambda}(f, \{y_0\})$ est l'ensemble des contenus de pile auxquels on peut arriver en lisant f depuis le contenu y_0

Langage reconnu par un aps **(définition 2)**

- $L(a) = \{ f \in X^* , 1\!\!1_Y \in \hat{\lambda}(f, \{y_0\})$

- rappel de la définition 1 :

$L(a) = \{ f \in X^*$, il existe un calcul valide menant de la configuration (f, y_0) à la configuration $(1\!\!1_X , 1\!\!1_Y) \}$

Forme normale de Greibach

- Une grammaire algébrique G = < X , V , S , P > est sous F N G si $P \subseteq V \times X\,V^*$

  - rappel grammaire algébrique quelconque : $P \subseteq V \times (X \cup V)^*$

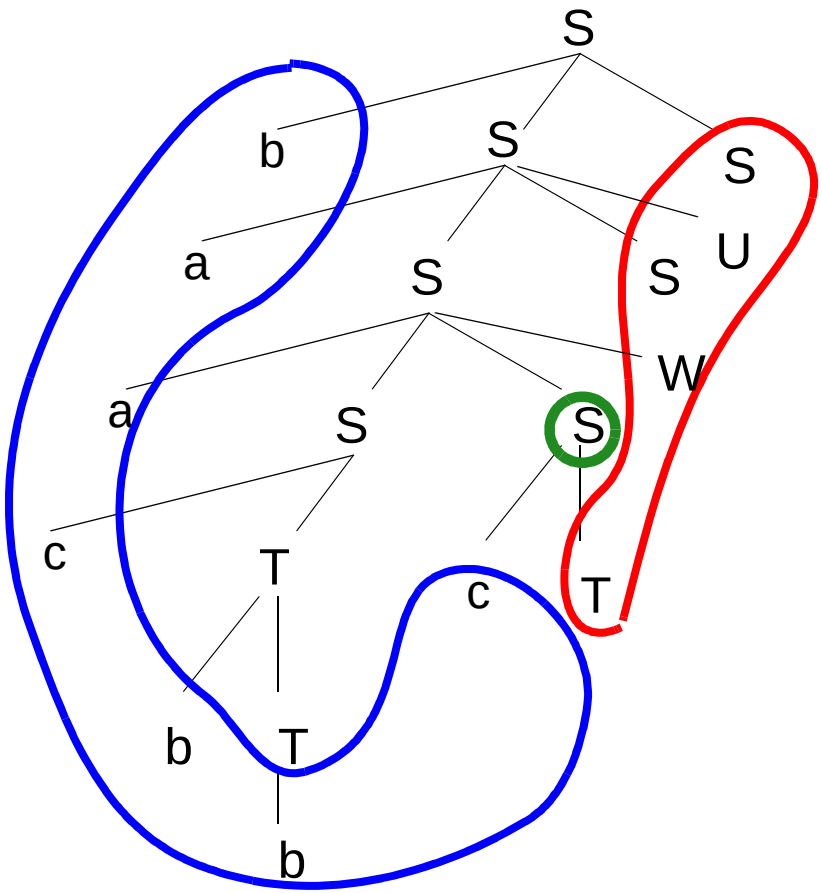
- Exemple :

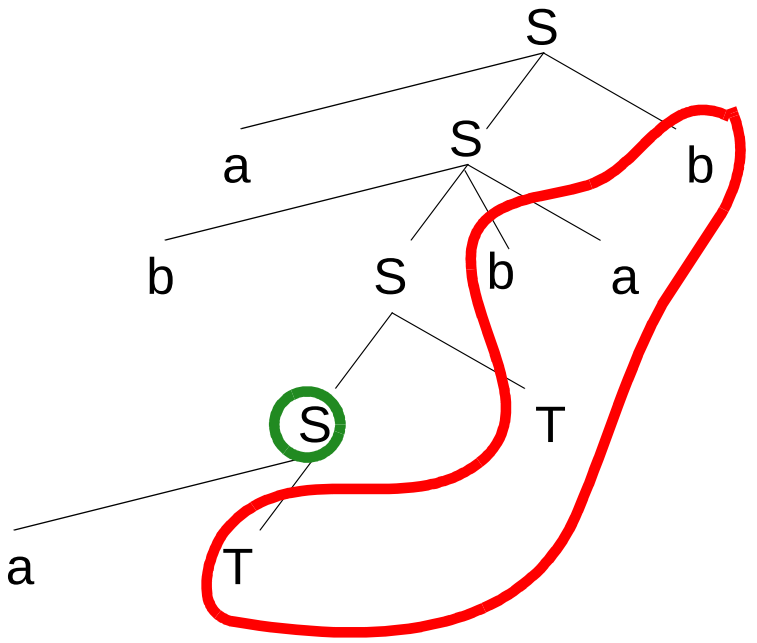
$$G : \begin{cases} S \rightarrow +SS \;\; / \;\; *SS \;\; / \;\; |T \;\; / \;\; b \\ T \rightarrow |T \;\; / \;\; b \end{cases}$$

est sous FNG.

Quiz - une grammaire sous FNG peut engendrer le mot vide vrai faux

Arbre de dérivation correspondant à une dérivation gauche, dans une grammaire sous FNG





Cas d'une grammaire quelconque

- en vert : le dernier non-terminal auquel on a appliqué une règle

Équivalence grammaire sous FNG / automate à pile simple.

- def : un langage est propre s'il ne contient pas le mot vide
- on admet : tout langage algébrique propre possède une grammaire sous FNG (cf poly)

- Un langage propre L est algébrique si et seulement si il existe un automate à pile simple qui le reconnaît.

sens: “seulement si”

Montrons que si un langage propre L est algébrique, alors il existe un automate à pile simple qui le reconnaît.

Soit L un langage algébrique propre et G = <X ,V, S , P> sous FNG telle que $L_G(S) = L$,

posons $\mathcal{a} = < X, V, S, \lambda >$ avec

$(x, T, w) \in \lambda$ si et seulement si $T \rightarrow x\,w \in P$ (on a ici $w \in V^*$)

on devra montrer (plus loin) que $L(\mathcal{a}) = L_G(S)$

- sens $L_G(S) \subseteq L(\mathcal{a})$

- sens $L(\mathcal{a}) \subseteq L_G(S)$

Rappel de l'exemple

(transformation grammaire → automate)

- par la construction (x , T , w) ∈ λ ssi T → xw ∈ P, la grammaire :

$$G : \begin{cases} S \rightarrow +SS \;\; / \;\; *SS \;\; / \;\; |T \;\; / \;\; b \\ T \rightarrow |T \;\; / \;\; b \end{cases}$$

conduit à l'automate :
X = { | , b , ∗ , + } , Y = { S, T },

+ , S → S S
∗ , S → S S
| , S → T
b , S → ε

| , T → T
b , T → ε

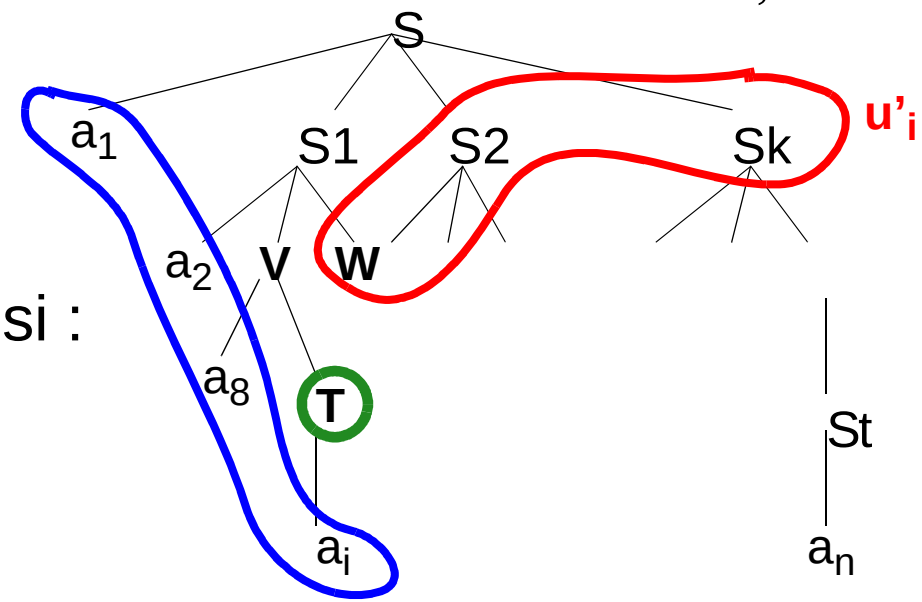
On le reconnaît bien.

sens $L_G(S) \subseteq L(a)$

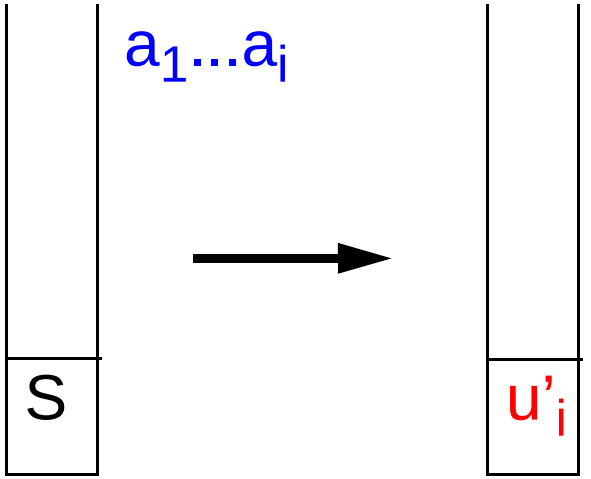
- Soit $f \in L_G(S)$, $f = a_1 a_2 ... a_n$, où les $a_i \in X$

$S \rightarrow_G^* f$ peut s'écrire (dérivation gauche) $S \rightarrow_G u_1 \rightarrow_G u_2 ... \rightarrow_G u_n = f$ avec, pour tout i, $1 \leq i \leq n$, $u_i = a_1 ... a_i \, u'_i$, $u'_i \in V^*$ car la grammaire est de Greibach.

si :



alors :



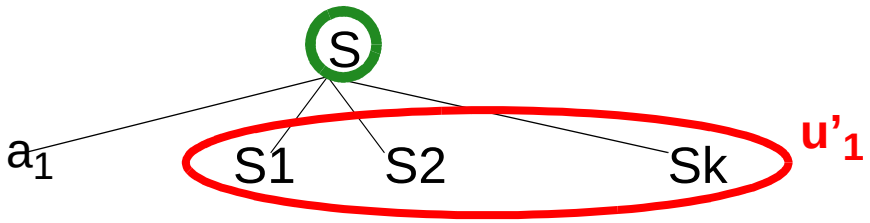
On montre par induction sur i que pour tout i, $1 \leq i \leq n$, $u'_i \in \hat{\lambda}(a_1 ... a_i , \{ S \})$

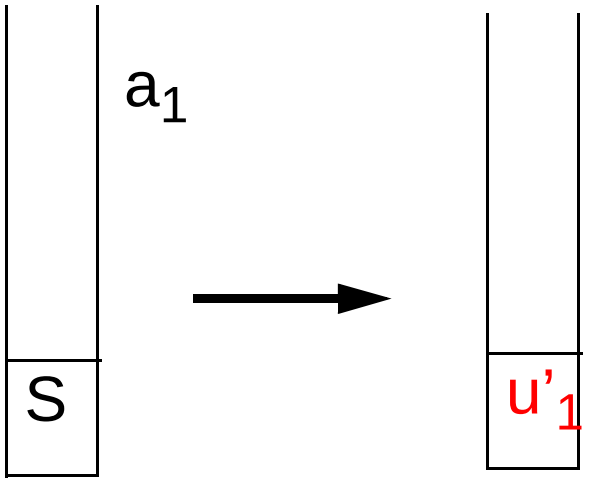
i.e. on trouve dans la pile ce qui reste à dériver pour produire $a_{i+1} ... a_n$.

On montre par induction sur i que pour tout i, $1 \leq i \leq n$, $u'_i \in \hat{\lambda}(a_1 ... a_i, \{ S \})$

On vérifie facilement le cas i = 1 sur le dessin :

$\hat{\lambda}(a_1, \{ S \}) = S_1 ... S_k = u'_1$ car $S \rightarrow_G a_1 S_1 ... S_k$





pour le cas i = n, on aura $u'_n \in \hat{\lambda}(a_1 ... a_n, \{ S \})$

c'est à dire $1l_Y \in \hat{\lambda}(f, \{S\})$ i.e. f est reconnu par a

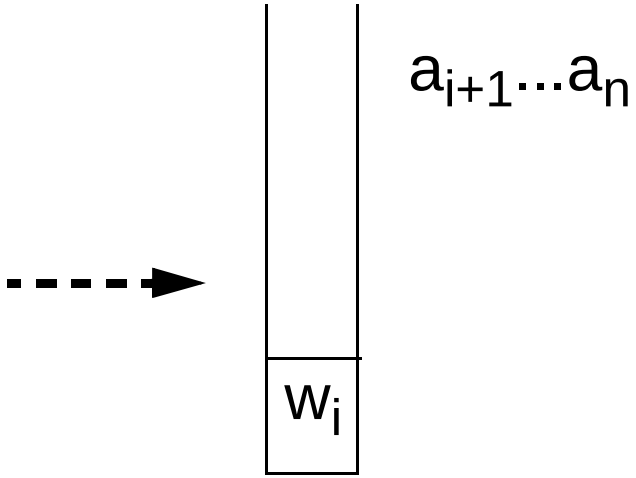
donc $L_G(S) \subseteq L(a)$.

sens $L(a) \subseteq L_G(S)$

- Soit $f \in L(a)$, $f = a_1 a_2 ... a_n$, $a_i \in X$

il existe un calcul valide
$(f , S = w_0) \rightarrow (a_2...a_n , w_1) \rightarrow ...(a_{i+1}...a_n , w_i) \rightarrow ...(a_n , w_{n-1}) \rightarrow (1\!\!1_X, 1\!\!1_Y)$

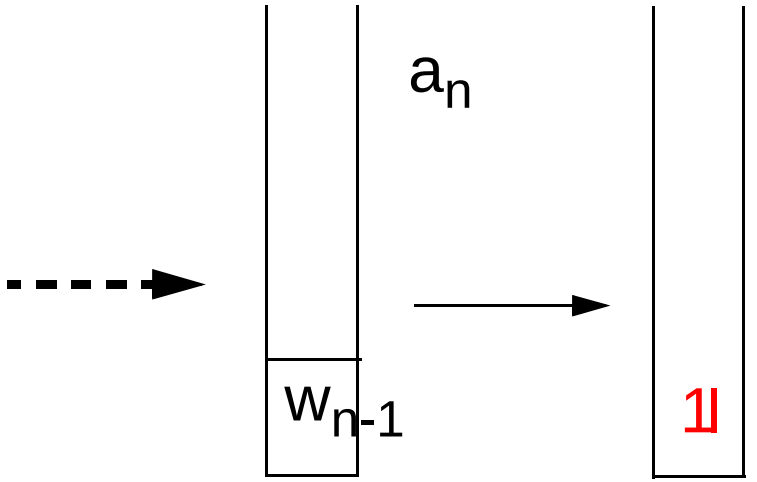
où w_i désigne le contenu de la pile au $i^{ème}$ pas de ce calcul.
On montre par induction sur i que pour tout i, $0 \leq i \leq n-1$,
$w_i \rightarrow_G^* a_{i+1}...a_n$ (l'induction se fait en décroissant)
i.e. on trouve dans la pile ce qui reste à dériver pour produire $a_{i+1}...a_n$.



alors $w_i \rightarrow_G^* a_{i+1}...a_n$

on veut si (f , S = w_0) $\rightarrow$ ...(a_{i+1}...a_n , w_i)...$\rightarrow$ (1l , 1l), alors $w_i \rightarrow_G^* a_{i+1}...a_n$

On vérifie facilement le dernier cas i = n-1



1l X

alors w_{n-1} = T, la règle de λ est

$(a_n, T) \rightarrow \varepsilon$

et elle provient de la règle $T \rightarrow_G a_n$

on a donc bien $w_{n-1} \rightarrow_G^* a_n$

On a finalement pour i = 0 : $S = w_0 \rightarrow_G^* a_1...a_n = f$

et donc f $\in$ $L_G(S)$.

d'où $L(a) \subseteq L_G(S)$

sens: “si”

Montrons que si un langage L est reconnu par un automate à pile simple, alors il est algébrique (et propre).

- Soit $a = < X , Y, y_0 , \lambda >$ un a.p.s. reconnaissant L,

  posons $G = < X ,Y, y_0 , P >$ avec

  $y \rightarrow x\ w \in P$ si et seulement si $(x , y , w) \in \lambda$,

  on montre que $L(a) = L_G(S)$

  C’est exactement la même preuve que précédemment.

Exemple (transformation automate → grammaire)

- l'automate :

  $a , y_0 \rightarrow y_0 y_1$
  $b , y_0 \rightarrow \varepsilon$
  $b , y_1 \rightarrow \varepsilon$ reconnaît $\{a^n b^{n+1}, n \geq 0\}$

- par la construction « $y \rightarrow x\ w \in P$ ssi $(x , y , w) \in \lambda$ », il conduit à la grammaire :

  $G = < \{a , b\}, \{y_0 , y_1\}, y_0 , P >$
  avec pour P : $y_0 \rightarrow a\ y_0\ y_1$
  $y_0 \rightarrow b$
  $y_1 \rightarrow b$ que l'on peut transformer en :

  $$S \rightarrow aSb \ / \ b$$