UE Langages Formels - IN4U06

Licence 2 - Informatique

Philippe Jégou

Département Informatique et Interactions

et

Laboratoire LIS
(Laboratoire de Recherche CNRS)
Équipe de recherche COALA
(Intelligence Artificielle et Programmation par Contraintes)
http://www.lis-lab.fr/recherche/calcul/coala/

Faculté des Sciences

Université d'Aix-Marseille

philippe.jegou@univ-amu.fr

22 janvier 2019

Informations pratiques

- Emploi du temps : consulter l'ENT
  - 6 séances CM de 1h30
  - 15h de TD (dès la semaine prochaine)
  - 3 séances de TP de 2h (en fin de semestre)

Informations pratiques

- Emploi du temps : consulter l'ENT
  - 6 séances CM de 1h30
  - 15h de TD (dès la semaine prochaine)
  - 3 séances de TP de 2h (en fin de semestre)
- Évaluation (MCC) :
  - Session 1 : NF = 0,2 x CC + 0,8 x ET

    Le CC (Contrôle Continu) sera lié à :
    - une interrogation écrite en TD
    - le rendu des TPs
  - Session 2 : NF = ET (Examen Terminal)

Informations pratiques

- Emploi du temps : consulter l'ENT
  - 6 séances CM de 1h30
  - 15h de TD (dès la semaine prochaine)
  - 3 séances de TP de 2h (en fin de semestre)
- Évaluation (MCC) :
  - Session 1 : NF = 0,2 x CC + 0,8 x ET

    Le CC (Contrôle Continu) sera lié à :
    - une interrogation écrite en TD
    - le rendu des TPs
  - Session 2 : NF = ET (Examen Terminal)
- Page web du cours : sur AMETICE

  où vous trouverez tout ce que j'y mettrai

Description de l'enseignement

Dans le programme de l'UE :

- Expressions régulières, langages rationnels
- Clôture rationnelle des AFN (concaténation, étoile de Kleene)
- Théorème de Kleene
- Résiduels, automate canonique et lemme de l'étoile
- Grammaire algébrique (modélisation des expressions arithmétiques)
- Algorithme CYK
- Automates à pile
- Evocation de la hiérarchie de Chomsky

UE Automates finis : un prérequis

- Notions de base sur les langages formels... bien sûr

et le contenu officiel que vous maîtrisez :

- Automates Finis Déterministes et Non-déterministes (AFD-AFN) et leur utilisation en modélisation
- Produit cartésien d'automates
- Déterminisation
- Clôture booléenne des AFN (union, intersection, complémentaire)
- Minimisation
- Test du vide et équivalence/inclusion de langages réguliers
- Recherche de motif (automate de Simon)

Plan

Plan

Introduction : retour vers les langages réguliers (rationnels)

Contenu : rappel du cours de l'UE "Automates finis"

- Ce qu'il faut vraiment maîtriser pour suivre cette UE
- Utilisation des mêmes notations...
- avec reprise des mêmes diapositives (une sélection)

Introduction : notions de base

Symboles et mots

- Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.
- Un alphabet est un ensemble fini de symboles. On désigne conventionnellement un alphabet par la lettre grecque Σ.
- Une suite de symboles, appartenant à un alphabet Σ, mis bout à bout est appelé un mot (ou une *chaîne*) sur Σ.
- On note $|m|$ la longueur du mot m (le nombre de symboles qui le composent).
- On note $|m|_s$ le nombre de symboles s que possède le mot m.
- Le mot de longueur zéro, appelé mot vide, est noté ε.
- Si m est un mot et $1 \leq i \leq |m|$, on note $m[i]$ le i^{eme} symbole de m.

Introduction : notions de base

Symboles et mots

- Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.
- Un alphabet est un ensemble fini de symboles. On désigne conventionnellement un alphabet par la lettre grecque Σ.
- Une suite de symboles, appartenant à un alphabet Σ, mis bout à bout est appelé un mot (ou une *chaîne*) sur Σ.
- On note $|m|$ la longueur du mot m (le nombre de symboles qui le composent).
- On note $|m|_s$ le nombre de symboles s que possède le mot m.
- Le mot de longueur zéro, appelé mot vide, est noté ε.
- Si m est un mot et $1 \leq i \leq |m|$, on note $m[i]$ le i^{eme} symbole de m.

Attention : un mot est suite finie de symboles

Introduction : notions de base

Concaténation

- La concaténation de deux mots α et β, notée $\alpha \cdot \beta$ ou simplement $\alpha\beta$ est le mot obtenu en juxtaposant les symboles de β à la suite de ceux de α.
- La concaténation est associative : $(\alpha\beta)\gamma = \alpha(\beta\gamma)$
- ε est l'élément neutre pour la concaténation : $\varepsilon\alpha = \alpha\varepsilon = \alpha$
- L'itération de l'opération de concaténation d'un mot m permet d'obtenir les puissances de m. Définition récursive :

$$
\begin{array}{rcl}
m^0 & = & \varepsilon \\
\forall n \in \mathbb{N}\ m^{n+1} & = & mm^n
\end{array}
$$

Introduction : notions de base

Concaténation

- La concaténation de deux mots α et β, notée $\alpha \cdot \beta$ ou simplement $\alpha\beta$ est le mot obtenu en juxtaposant les symboles de β à la suite de ceux de α.
- La concaténation est associative : $(\alpha\beta)\gamma = \alpha(\beta\gamma)$
- ε est l'élément neutre pour la concaténation : $\varepsilon\alpha = \alpha\varepsilon = \alpha$
- L'itération de l'opération de concaténation d'un mot m permet d'obtenir les puissances de m. Définition récursive :

$$
\begin{array}{rcl}
m^0 & = & \varepsilon \\
\forall n \in \mathbb{N}\ m^{n+1} & = & mm^n
\end{array}
$$

Attention : ε est un mot et non un symbole

Introduction : notions de base

Langages

- L'ensemble de tous les mots que l'on peut construire sur un alphabet Σ est noté Σ^*.
- Un langage sur un alphabet Σ est un ensemble de mots construits sur Σ.
- Tout langage défini sur Σ est donc un sous-ensemble de Σ^*.
- L'ensemble de tous les langages que l'on peut définir sur Σ^* est l'ensemble des parties de Σ^*, noté $\mathcal{P}(\Sigma^*)$.

Introduction : notions de base

Langages

- L'ensemble de tous les mots que l'on peut construire sur un alphabet Σ est noté Σ^*.
- Un langage sur un alphabet Σ est un ensemble de mots construits sur Σ.
- Tout langage défini sur Σ est donc un sous-ensemble de Σ^*.
- L'ensemble de tous les langages que l'on peut définir sur Σ^* est l'ensemble des parties de Σ^*, noté $\mathcal{P}(\Sigma^*)$.

et Opérations sur les langages

Union	$L_1 \cup L_2$	$\{x \mid x \in L_1 \text{ ou } x \in L_2\}$
Intersection	$L_1 \cap L_2$	$\{x \mid x \in L_1 \text{ et } x \in L_2\}$
Différence	$L_1 - L_2$	$\{x \mid x \in L_1 \text{ et } x \notin L_2\}$
Complément	$\bar{L}$	$\{x \in \Sigma^* \mid x \notin L\}$
Concaténation	$L_1 L_2$	$\{xy \mid x \in L_1 \text{ et } y \in L_2\}$
Auto concaténation	$\overbrace{L \dots L}^{n}$	L^n
Fermeture de Kleene	L^*	$\bigcup_{k \geq 0} L^k$

Introduction : langages réguliers

Langages réguliers

Etant donné un alphabet Σ, on appelle langage régulier sur Σ un langage sur Σ défini de la façon suivante :

1. $\emptyset$ (l'ensemble vide) est un langage régulier sur Σ.
2. $\{\varepsilon\}$ est un langage régulier sur Σ.
3. $\{a\}$ est un langage régulier sur Σ pour tout $a \in \Sigma$.
4. Si P et Q sont des langages réguliers sur Σ, alors les langages suivants sont des langages réguliers :
    1. $P \cup Q$
    2. PQ
    3. P^*
5. rien d'autre n'est un langage régulier.

Introduction : langages réguliers et expressions

Expressions régulières

- Les symboles sont définis comme suit :
  1. $\emptyset$ est une expression régulière dénotant le langage régulier $\emptyset$.
  2. ε est une expression régulière dénotant le langage régulier $\{\varepsilon\}$.
  3. a (tel que $a \in \Sigma$) est une expression régulière dénotant le langage régulier $\{a\}$.
- Les opérateurs sont définis comme suit :
  - Si p et q sont des expressions régulières dénotant respectivement les ensembles réguliers P et Q alors :
    1. $(p+q)$ est une expression régulière dénotant le langage régulier $P \cup Q$
    2. (pq) est une expression régulière dénotant le langage régulier PQ
    3. $(p)^*$ est une expression régulière dénotant le langage régulier P^*
- rien d'autre n'est une expression régulière.

Introduction : langages réguliers et expressions

Expressions régulières

- Les symboles sont définis comme suit :
  1. $\emptyset$ est une expression régulière dénotant le langage régulier $\emptyset$.
  2. ε est une expression régulière dénotant le langage régulier $\{\varepsilon\}$.
  3. a (tel que $a \in \Sigma$) est une expression régulière dénotant le langage régulier $\{a\}$.
- Les opérateurs sont définis comme suit :
  - Si p et q sont des expressions régulières dénotant respectivement les ensembles réguliers P et Q alors :
    1. $(p+q)$ est une expression régulière dénotant le langage régulier $P \cup Q$
    2. (pq) est une expression régulière dénotant le langage régulier PQ
    3. $(p)^*$ est une expression régulière dénotant le langage régulier P^*
- rien d'autre n'est une expression régulière.

Quelques exemples

$$
\begin{aligned}
0^*10^* &= \{m \in \{0,1\}^* \mid m \text{ a exactement un } 1\} \\
(0+1)^*1(0+1)^* &= \{m \in \{0,1\}^* \mid m \text{ a au moins un } 1\} \\
(0+1)^*001(0+1)^* &= \{m \in \{0,1\}^* \mid m \text{ contient la sous-chaîne } 001\} \\
((0+1)(0+1))^* &= \{m \in \{0,1\}^* \mid |m| \text{ est pair}\}
\end{aligned}
$$

Introduction : langages réguliers et expressions

Expressions régulières

- Les symboles sont définis comme suit :
  1. $\emptyset$ est une expression régulière dénotant le langage régulier $\emptyset$.
  2. ε est une expression régulière dénotant le langage régulier $\{\varepsilon\}$.
  3. a (tel que $a \in \Sigma$) est une expression régulière dénotant le langage régulier $\{a\}$.
- Les opérateurs sont définis comme suit :
  - Si p et q sont des expressions régulières dénotant respectivement les ensembles réguliers P et Q alors :
    1. $(p+q)$ est une expression régulière dénotant le langage régulier $P \cup Q$
    2. (pq) est une expression régulière dénotant le langage régulier PQ
    3. $(p)^*$ est une expression régulière dénotant le langage régulier P^*
- rien d'autre n'est une expression régulière.

Introduction : langages réguliers et expressions

Expressions régulières

- Les symboles sont définis comme suit :
  1. $\emptyset$ est une expression régulière dénotant le langage régulier $\emptyset$.
  2. ε est une expression régulière dénotant le langage régulier $\{\varepsilon\}$.
  3. a (tel que $a \in \Sigma$) est une expression régulière dénotant le langage régulier $\{a\}$.
- Les opérateurs sont définis comme suit :
  - Si p et q sont des expressions régulières dénotant respectivement les ensembles réguliers P et Q alors :
    1. $(p+q)$ est une expression régulière dénotant le langage régulier $P \cup Q$
    2. (pq) est une expression régulière dénotant le langage régulier PQ
    3. $(p)^*$ est une expression régulière dénotant le langage régulier P^*
- rien d'autre n'est une expression régulière.

Théorème :

$$L \text{ est un langage régulier}$$
$$\Leftrightarrow$$
$$L \text{ peut s'exprimer par une expression régulière}$$

Introduction : langages réguliers et expressions

Propriétés et équivalences

$\alpha + (\beta + \gamma)$	$\equiv$	$(\alpha + \beta) + \gamma$	associativité de l'union
$\alpha + \beta$	$\equiv$	$\beta + \alpha$	commutativité de l'union
$\alpha + \emptyset$	$\equiv$	α	$\emptyset$ élément neutre pour l'union
$\alpha + \alpha$	$\equiv$	α	idempotence de l'union
$(\alpha^*)^*$	$\equiv$	α^*	idempotence de l'étoile de kleene
$\alpha(\beta\gamma)$	$\equiv$	$(\alpha\beta)\gamma$	associativité de la concaténation
$\varepsilon\alpha$	$\equiv$	$\alpha\varepsilon \equiv \alpha$	ε élément neutre de la concaténation
$\alpha(\beta + \gamma)$	$\equiv$	$\alpha\beta + \alpha\gamma$	distributivité de la concat. sur l'union
$(\alpha + \beta)\gamma$	$\equiv$	$\alpha\gamma + \beta\gamma$	distributivité de la concat. sur l'union
$\emptyset\alpha$	$\equiv$	$\alpha\emptyset \equiv \emptyset$	$\emptyset$ élément absorbant pour la concat.
$\emptyset^*$	$\equiv$	ε	
$\varepsilon + \alpha\alpha^*$	$\equiv$	α^*	
$\varepsilon + \alpha^*\alpha$	$\equiv$	α^*	
$\alpha + \alpha^*$	$\equiv$	α^*	
$(\varepsilon + \alpha)^*$	$\equiv$	α^*	
$\alpha^* + \alpha\alpha^*$	$\equiv$	α^*	
$(\alpha\beta)^*\alpha$	$\equiv$	$\alpha(\beta\alpha)^*$	
$(\alpha^*\beta)^*\alpha^*$	$\equiv$	$(\alpha + \beta)^*$	
$\alpha^*(\beta\alpha^*)^*$	$\equiv$	$(\alpha + \beta)^*$	

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
    - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
    - Σ est un alphabet contenant des symboles d'entrée

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition :
    $\delta(q_i, s) = q_j$ ssi dans q_i la lecture de s fait passer dans q_j

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to Q$ est la fonction de transition :
    $\delta(q_i, s) = q_j$ ssi dans q_i la lecture de s fait passer dans q_j
  - q_0 est l'état initial

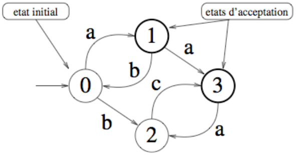
Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \dots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to Q$ est la fonction de transition :
    $\delta(q_i, s) = q_j$ ssi dans q_i la lecture de s fait passer dans q_j
  - q_0 est l'état initial
  - $F \subseteq Q$ est l'ensemble des états finaux (ou acceptants ou terminaux)

Introduction : Automates

- Automates Finis Déterministes (AFD) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to Q$ est la fonction de transition :
    $\delta(q_i, s) = q_j$ ssi dans q_i la lecture de s fait passer dans q_j
  - q_0 est l'état initial
  - $F \subseteq Q$ est l'ensemble des états finaux (ou acceptants ou terminaux)
- Représentations formelle, graphique et par table de transition :

$\langle Q, \Sigma, \delta, q_0, F \rangle$
$Q = \{0, 1, 2, 3\}$
$\Sigma = \{a, b, c\}$
$\delta(0, a) = \{1\}$
$\delta(0, b) = \{2\}$
$\delta(1, a) = \{3\}$
$\delta(1, b) = \{0\}$
$\delta(2, c) = \{3\}$
$\delta(3, a) = \{2\}$
$q_0 = 0$
$F = \{1, 3\}$



		a	b	c
$\rightarrow$	0	1	2	
$\leftarrow$	1	3	0	
	2			3
$\leftarrow$	3	2		

Introduction : Automates

- Langage reconnu par un AFD $A = (Q, \Sigma, \delta, q_0, F)$:

Introduction : Automates

- Langage reconnu par un AFD $A = (Q, \Sigma, \delta, q_0, F)$:
  - Fonction de transition étendue aux mots : $\widehat{\delta} : Q \times \Sigma^* \rightarrow Q$ avec :
    - $\widehat{\delta}(q_i, \varepsilon) = q_i$
    - $\forall m \in \Sigma^*, \forall s \in \Sigma$, on a $\widehat{\delta}(q_i, m.s) = \delta(\widehat{\delta}(q_i, m), s)$.

  ainsi $\widehat{\delta}(q_i, m) = q_j$ signifie que la lecture de m à partir de q_i mène à q_j.

Introduction : Automates

- Langage reconnu par un AFD $A = (Q, \Sigma, \delta, q_0, F)$:
  - Fonction de transition étendue aux mots : $\widehat{\delta} : Q \times \Sigma^* \to Q$ avec :
    - $\widehat{\delta}(q_i, \varepsilon) = q_i$
    - $\forall m \in \Sigma^*, \forall s \in \Sigma$, on a $\widehat{\delta}(q_i, m.s) = \delta(\widehat{\delta}(q_i, m), s)$.

    ainsi $\widehat{\delta}(q_i, m) = q_j$ signifie que la lecture de m à partir de q_i mène à q_j.
  - Un mot $m \in \Sigma^*$ est dit accepté par $A = (Q, \Sigma, \delta, q_0, F)$ si $\widehat{\delta}(q_0, m) \in F$

    (un mot m est accepté par un A si la lecture des symboles de m à partir de q_0 mène à un état final)

Introduction : Automates

- Langage reconnu par un AFD $A = (Q, \Sigma, \delta, q_0, F)$:
  - Fonction de transition étendue aux mots : $\widehat{\delta} : Q \times \Sigma^* \rightarrow Q$ avec :
    - $\widehat{\delta}(q_i, \varepsilon) = q_i$
    - $\forall m \in \Sigma^*, \forall s \in \Sigma$, on a $\widehat{\delta}(q_i, m.s) = \delta(\widehat{\delta}(q_i, m), s)$.

    ainsi $\widehat{\delta}(q_i, m) = q_j$ signifie que la lecture de m à partir de q_i mène à q_j.
  - Un mot $m \in \Sigma^*$ est dit accepté par $A = (Q, \Sigma, \delta, q_0, F)$ si $\widehat{\delta}(q_0, m) \in F$

    (un mot m est accepté par un A si la lecture des symboles de m à partir de q_0 mène à un état final)
  - Le langage reconnu par un automate A est défini par :

    $L(A) = \{m \in \Sigma^* \mid \widehat{\delta}(q_0, m) \in F\}$

    (le langage reconnu par A est l'ensemble des mots qu'il accepte)

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to 2^Q = \mathcal{P}(Q)$ est la fonction de transition :
    $\delta(q_i, s) = Q_j \subseteq Q$ ssi dans q_i la lecture de s fait passer dans les états de Q_j

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \dots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to 2^Q = \mathcal{P}(Q)$ est la fonction de transition : $\delta(q_i, s) = Q_j \subseteq Q$ ssi dans q_i la lecture de s fait passer dans les états de Q_j
  - q_0 est l'état initial

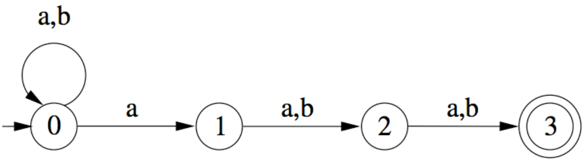
Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \to 2^Q = \mathcal{P}(Q)$ est la fonction de transition : $\delta(q_i, s) = Q_j \subseteq Q$ ssi dans q_i la lecture de s fait passer dans les états de Q_j
  - q_0 est l'état initial
  - $F \subseteq Q$ est l'ensemble des états finaux (ou acceptants ou terminaux)

Introduction : Automates

- Automates Finis Non Déterministes (AFND) : $A = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots q_n\}$ est un ensemble fini d'états
  - Σ est un alphabet contenant des symboles d'entrée
  - $\delta : Q \times \Sigma \rightarrow 2^Q = \mathcal{P}(Q)$ est la fonction de transition :
    $\delta(q_i, s) = Q_j \subseteq Q$ ssi dans q_i la lecture de s fait passer dans les états de Q_j
  - q_0 est l'état initial
  - $F \subseteq Q$ est l'ensemble des états finaux (ou acceptants ou terminaux)

- Représentations graphique :



où on remarque que $\delta(q_0, a) = \{q_0, q_1\} \subseteq Q = \{q_0, q_1, q_2, q_3\}$

Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"

Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"
  - $A = (Q, \Sigma, \delta, q_0, F)$ avec mêmes définitions pour Q, Σ, q_0 et F

Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"
  - $A = (Q, \Sigma, \delta, q_0, F)$ avec mêmes définitions pour Q, Σ, q_0 et F
  - Ce qui change :

    ajout d'un symbole epsilon à l'alphabet pour δ :

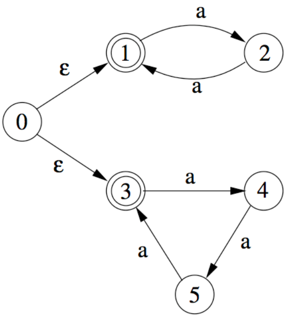
    $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q = \mathcal{P}(Q)$

Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"
  - $A = (Q, \Sigma, \delta, q_0, F)$ avec mêmes définitions pour Q, Σ, q_0 et F
  - Ce qui change :

    ajout d'un symbole epsilon à l'alphabet pour δ :

    $\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q = \mathcal{P}(Q)$
- Représentation graphique :



Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"
  - $A = (Q, \Sigma, \delta, q_0, F)$ avec mêmes définitions pour Q, Σ, q_0 et F
  - Ce qui change :

    ajout d'un symbole epsilon à l'alphabet pour δ :

    $\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q = \mathcal{P}(Q)$
- Langage reconnu par AFND : même principe que pour AFD sauf que

  plutôt que $L(A) = \{m \in \Sigma^* \mid \widehat{\delta}(q_0, m) \in F\}$

  on a : $L(A) = \{m \in \Sigma^* \mid \widehat{\delta}(q_0, m) \cap F \neq \emptyset\}$

Introduction : Automates

- Automates Finis Non Déterministes (AFND) avec "epsilon-transitions"
  - $A = (Q, \Sigma, \delta, q_0, F)$ avec mêmes définitions pour Q, Σ, q_0 et F
  - Ce qui change :

    ajout d'un symbole epsilon à l'alphabet pour δ :

    $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q = \mathcal{P}(Q)$
- Langage reconnu par AFND : même principe que pour AFD sauf que

  plutôt que $L(A) = \{m \in \Sigma^* \mid \widehat{\delta}(q_0, m) \in F\}$

  on a : $L(A) = \{m \in \Sigma^* \mid \widehat{\delta}(q_0, m) \cap F \neq \emptyset\}$

  Le langage reconnu par A est l'ensemble des mots qui permettent d'atteindre au moins un état final

Introduction : Automates

Ce que vous devez connaître (vous en souvenir) :

- Langage reconnaissable :

  langage L tel qu'il existe un AFD A avec $L = L(A)$

Introduction : Automates

Ce que vous devez connaître (vous en souvenir) :

- Langage reconnaissable :

  langage L tel qu'il existe un AFD A avec $L = L(A)$

- AFD complété :

  pour les transitions non-autorisées : création d'un état "puits"

Introduction : Automates

Ce que vous devez connaître (vous en souvenir) :

- Langage reconnaissable :

  langage L tel qu'il existe un AFD A avec $L = L(A)$

- AFD complété :

  pour les transitions non-autorisées : création d'un état "puits"

- AFD "canonique" dit aussi "automate minimal" pour un langage L
  - AFD complet reconnaissant L et possédant le moins d'états
  - cet AFD est unique à un renommage près des états
  - cet AFD est obtenu avec l'algorithme de Moore en $O(|\Sigma|.|Q|^2)$

Introduction : Automates

Ce que vous devez connaître (vous en souvenir) :

- Langage reconnaissable :

  langage L tel qu'il existe un AFD A avec $L = L(A)$
- AFD complété :

  pour les transitions non-autorisées : création d'un état "puits"
- AFD "canonique" dit aussi "automate minimal" pour un langage L
  - AFD complet reconnaissant L et possédant le moins d'états
  - cet AFD est unique à un renommage près des états
  - cet AFD est obtenu avec l'algorithme de Moore en $O(|\Sigma|.|Q|^2)$
- Les AFD et les AFND reconnaissent la même classe de langages
  - les AFND ne sont donc pas plus puissants que les AFD
  - résultat prouvé par "déterminisation" d'AFND en AFD équivalent

Introduction : Langages et Clôtures

Ce que vous devez connaître (vous en souvenir) :

- Clôture par opérations régulières (on "reste" dans les réguliers) :

  Si L_1 et L_2 sont des langages réguliers et <op> un "opérateur réguliers" (opérateur réguliers : $\cup, \cap, ., *, \ldots$)

  alors L_1 <op> L_2 est un langage régulier

Introduction : Langages et Clôtures

Ce que vous devez connaître (vous en souvenir) :

- Clôture par opérations régulières (on "reste" dans les réguliers) :

  Si L_1 et L_2 sont des langages réguliers et <op> un "opérateur réguliers" (opérateur réguliers : $\cup, \cap, ., *, \dots$)

  alors L_1 <op> L_2 est un langage régulier
- Ce qui a été vu avec les automates :
  - Clos par intersection : $L(A_1) \cap L(A_2) = L(A_1 \times A_2)$

    ($\times$ = produit d'automates)
  - Clos par union : $L(A_1) \cup L(A_2) = L(A_1 \cup A_2)$

    (union d'automates)
  - Clos par concaténation : $L(A_1).L(A_2) = L(A_1.A_2)$

    (concaténation d'automates)
  - Clos par l'Étoile de Kleene : $L(A^*) = L(A)^*$ (Étoile de Kleene)

Introduction : Langages réguliers et Automates

Ce que vous devez connaître (vous en souvenir) :

- Langages réguliers $\subseteq$ Langages reconnaissables
  - tout langage exprimable par une expression régulière est un langage reconnaissable
  - preuve constructive : à partir d'une expression régulière, on construit un AFND reconnaissant le même langage
    (Algorithme de Thompson en $O(n)$ où n est la taille de l'expression)
- Théorème de Kleene : Langages réguliers = Langages reconnaissables
  - L'inclusion [*Langages réguliers* $\subseteq$ *Langages reconnaissables*] : première partie de la preuve du théorème de Kleene
  - L'inclusion dans l'autre sens : termine la preuve (sera vue en cours)

Pour conclure : Description de l'enseignement

Dans le programme de l'UE :

- Expressions régulières, langages rationnels
- Clôture rationnelle des AFN (concaténation, étoile de Kleene)
- Théorème de Kleene
- Résiduels, automate canonique et lemme de l'étoile
- Grammaire algébrique (modélisation des expressions arithmétiques)
- Algorithme CYK
- Automates à pile
- Évocation de la hiérarchie de Chomsky

Pour conclure : Description de l'enseignement

Ce que l'on verra dans l'UE :

- Expressions régulières, langages rationnels
- Clôture rationnelle des AFN (concaténation, étoile de Kleene)
- Théorème de Kleene
- Résiduels, automate canonique et lemme de l'étoile
- Grammaire algébrique (modélisation des expressions arithmétiques)
- Algorithme CYK
- Automates à pile
- Evocation de la hiérarchie de Chomsky

Plan

Plan

Théorème de Kleene

Théorème de Kleene

Langages réguliers = Langages reconnaissables

Théorème de Kleene

Théorème de Kleene

$$\text{Langages réguliers} = \text{Langages reconnaissables}$$

Preuve : on montre l'inclusion dans les deux sens

Théorème de Kleene

Théorème de Kleene

$$\text{Langages réguliers} = \text{Langages reconnaissables}$$

Preuve : on montre l'inclusion dans les deux sens

(1) Langages réguliers $\subseteq$ Langages reconnaissables

Déjà fait en S3 !

(par construction d'automates à partir d'expressions régulières)

Théorème de Kleene

Théorème de Kleene

$$\text{Langages réguliers} = \text{Langages reconnaissables}$$

Preuve : on montre l'inclusion dans les deux sens

(1) Langages réguliers $\subseteq$ Langages reconnaissables

Déjà fait en S3 !

(par construction d'automates à partir d'expressions régulières)

(2) Langages reconnaissables $\subseteq$ Langages reguliers

On va le faire.

Théorème de Kleene : Preuve

On va montrer que tout langage L reconnaissable (par un AFD) peut être exprimé par une expressions régulière.

On considère donc un langage $L(A)$ défini par l'AFD A :

- Soit donc un langage $L(A)$ avec $A = (\{q_1, q_2, \ldots \ldots q_n\}, \Sigma, \delta, q_1, F)$ (pour simplifier les notations, on note ici l'état initial q_1 et non q_0)
- **Idée de la preuve :**
  - à partir des transitions de A, on construit l'expression régulière associée
  - par exemple, si on a $\delta(q_i, s) = q_j$ alors l'expression associée est s
- mais on doit introduire des notations assez complexes...

Théorème de Kleene : Preuve

- Soit R_{ij}^{k} l'ensemble des chaînes x partant de q_i et allant à q_j en passant par des états intermédiaires q_{int} d'indice $int \leq k$

  Ce sont les chaînes x telles que :

  - $\widehat{\delta}(q_i, x) = q_j$ et
  - pour tout préfixe propre y (ni ε ni x), on a $\widehat{\delta}(q_i, y) = q_{int} \Rightarrow int \leq k$

  Cela n'empêche par d'avoir $i > k$ et/ou $j > k$

Théorème de Kleene : Preuve

- Soit R_{ij}^{k} l'ensemble des chaînes x partant de q_i et allant à q_j en passant par des états intermédiaires q_{int} d'indice $int \leq k$

  Ce sont les chaînes x telles que :

  - $\widehat{\delta}(q_i, x) = q_j$ et
  - pour tout préfixe propre y (ni ε ni x), on a $\widehat{\delta}(q_i, y) = q_{int} \Rightarrow int \leq k$

  Cela n'empêche par d'avoir $i > k$ et/ou $j > k$

- Ainsi, on a :

  R_{ij}^{n} l'ensemble de toutes les chaînes x partant de q_i et allant à q_j

  car q_n est le plus grand indice d'état

Théorème de Kleene : Preuve

- Soit R_{ij}^k l'ensemble des chaînes x partant de q_i et allant à q_j en passant par des états intermédiaires q_{int} d'indice $int \leq k$

  Ce sont les chaînes x telles que :
  - $\widehat{\delta}(q_i, x) = q_j$ et
  - pour tout préfixe propre y (ni ε ni x), on a $\widehat{\delta}(q_i, y) = q_{int} \Rightarrow int \leq k$

  Cela n'empêche par d'avoir $i > k$ et/ou $j > k$

- Ainsi, on a :

  R_{ij}^n l'ensemble de toutes les chaînes x partant de q_i et allant à q_j

  car q_n est le plus grand indice d'état

- R_{ij}^k correspond en fait à une expression régulière notée r_{ij}^k

  r_{ij}^k est l'expression associée au langage des chaînes de R_{ij}^k

Théorème de Kleene : Preuve

Lien avec le langage reconnu $L(A)$:

- $L(A) = \bigcup_{q_j \in F} R^n_{1j}$

  car tous les mots reconnus par A sont les chaînes x telles que :

  - $\widehat{\delta}(q_1, x) = q_j$ avec $q_j \in F$ et
  - dont les états de passage sont dans $Q = \{q_1, q_2, \ldots ...q_n\}$

Théorème de Kleene : Preuve

Lien avec le langage reconnu $L(A)$:

- $L(A) = \bigcup_{q_j \in F} R^n_{1j}$

  car tous les mots reconnus par A sont les chaînes x telles que :

  - $\widehat{\delta}(q_1, x) = q_j$ avec $q_j \in F$ et
  - dont les états de passage sont dans $Q = \{q_1, q_2, \ldots\ldots q_n\}$

- Ainsi, $L(A)$ est représenté par l'expression $r^n_{1j_1} + r^n_{1j_2} + \cdots + r^n_{1j_p}$

  avec $F = \{q_{j_1}, q_{j_2}, \ldots\ldots q_{j_p}\}$

Théorème de Kleene : Preuve

Lien avec le langage reconnu $L(A)$:

- $L(A) = \bigcup_{q_j \in F} R^n_{1j}$

  car tous les mots reconnus par A sont les chaînes x telles que :
  - $\widehat{\delta}(q_1, x) = q_j$ avec $q_j \in F$ et
  - dont les états de passage sont dans $Q = \{q_1, q_2, \ldots \ldots q_n\}$

- Ainsi, $L(A)$ est représenté par l'expression $r^n_{1j_1} + r^n_{1j_2} + \cdots + r^n_{1j_p}$

  avec $F = \{q_{j_1}, q_{j_2}, \ldots \ldots q_{j_p}\}$

Il faut maintenant montrer comment on peut construire les différents R^n_{1j}

La construction se fait par induction en définissant tous les R^k_{ij}

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Base de l'induction : $k = 0$

(on ne passe par aucun état intermédiaire pour définir R_{ij}^{0})

Théorème de Kleene : Preuve

Construction des R_{ij}^k

Base de l'induction : $k = 0$

(on ne passe par aucun état intermédiaire pour définir R_{ij}^0)

- Soit une transition (arc) de l'état q_i vers l'état q_j, ou

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Base de l'induction : $k = 0$

(on ne passe par aucun état intermédiaire pour définir R_{ij}^{0})

- Soit une transition (arc) de l'état q_i vers l'état q_j, ou
- Aucune transition (chemin de longueur nulle) avec juste l'état q_i

Et donc R_{ij}^{0} est l'ensemble des chaînes d'un symbole s ou aucun (ε)

Théorème de Kleene : Preuve

Construction des R_{ij}^k

Base de l'induction : $k = 0$

(on ne passe par aucun état intermédiaire pour définir R_{ij}^0)

- Soit une transition (arc) de l'état q_i vers l'état q_j, ou
- Aucune transition (chemin de longueur nulle) avec juste l'état q_i

Et donc R_{ij}^0 est l'ensemble des chaînes d'un symbole s ou aucun (ε)

r_{ij}^0 peut être formulée par l'expression $s_1 + s_2 + \cdots + s_\alpha$ $(+\varepsilon$ si $i = j)$

avec $\{s_1, s_2, \ldots s_\alpha\} = \{s \in \Sigma \mid \delta(q_i, s) = q_j\}$

Théorème de Kleene : Preuve

Construction des R_{ij}^k

Étape d'induction : $k > 0$

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois
  - suivie de zéro ou plusieurs chaînes de R_{kk}^{k-1}

Théorème de Kleene : Preuve

Construction des R_{ij}^k

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois
  - suivie de zéro ou plusieurs chaînes de R_{kk}^{k-1}
  - suivie(s) d'une chaîne de R_{kj}^{k-1} pour atteindre q_j

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois
  - suivie de zéro ou plusieurs chaînes de R_{kk}^{k-1}
  - suivie(s) d'une chaîne de R_{kj}^{k-1} pour atteindre q_j

  ce qui correspond aux chaînes de $R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1}$

Théorème de Kleene : Preuve

Construction des R_{ij}^{k}

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois
  - suivie de zéro ou plusieurs chaînes de R_{kk}^{k-1}
  - suivie(s) d'une chaîne de R_{kj}^{k-1} pour atteindre q_j

  ce qui correspond aux chaînes de $R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1}$

On arrive ainsi à $R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1}$

Théorème de Kleene : Preuve

Construction des R_{ij}^k

Étape d'induction : $k > 0$

On a deux possibilités :

- Les chaînes passant "dessous" q_k : de la forme R_{ij}^{k-1}
- Les chaînes passant "par" q_k : elles sont composées par
  - une chaîne de R_{ik}^{k-1} allant de q_i à q_k pour la première fois
  - suivie de zéro ou plusieurs chaînes de R_{kk}^{k-1}
  - suivie(s) d'une chaîne de R_{kj}^{k-1} pour atteindre q_j

  ce qui correspond aux chaînes de $R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

On arrive ainsi à $R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

Cela correspond à l'expression régulière $r_{ij}^{k-1} + r_{ik}^{k-1}(r_{kk}^{k-1})^* r_{kj}^{k-1}$ $\square$

Théorème de Kleene

Pour conclure :

Théorème de Kleene

$$\text{Langages réguliers} = \text{Langages reconnaissables}$$

Théorème de Kleene

Pour conclure :

Théorème de Kleene

$$\text{Langages réguliers} = \text{Langages reconnaissables}$$

Mais... existe-t-il des langages qui ne sont pas réguliers
et donc non reconnaissables par les AFD ?

C'est l'objet du chapitre suivant...

Plan

Plan

Lemme de l'étoile

Motivations :

- Le langage $L = \{a^n b^n \mid n \geq 0\}$ est-il régulier ?

Lemme de l'étoile

Motivations :

- Le langage $L = \{a^n b^n \mid n \geq 0\}$ est-il régulier ?

  On peut chercher un AFD le reconnaissant...

  Mais on n'en trouvera pas !!!

Lemme de l'étoile

Motivations :

- Le langage $L = \{a^n b^n \mid n \geq 0\}$ est-il régulier ?

  On peut chercher un AFD le reconnaissant...

  Mais on n'en trouvera pas !!!

- En fait : tous les langages ne sont pas réguliers !

  Mais comment le prouver ?

Lemme de l'étoile

Motivations :

- Le langage $L = \{a^n b^n \mid n \geq 0\}$ est-il régulier ?

  On peut chercher un AFD le reconnaissant...

  Mais on n'en trouvera pas !!!

- En fait : tous les langages ne sont pas réguliers !

  Mais comment le prouver ?

- Grâce au "lemme de l'étoile"... qui possède plusieurs noms :
  - lemme du facteur itérant ou lemme d'itération
  - lemme de gonflement
  - lemme d'extraction
  - lemme de pompage ("pumping lemma")

Lemme de l'étoile

Motivations :

- Le langage $L = \{a^n b^n \mid n \geq 0\}$ est-il régulier ?

  On peut chercher un AFD le reconnaissant...

  Mais on n'en trouvera pas !!!

- En fait : tous les langages ne sont pas réguliers !

  Mais comment le prouver ?

- Grâce au "lemme de l'étoile"... qui possède plusieurs noms :
  - lemme du facteur itérant ou lemme d'itération
  - lemme de gonflement
  - lemme d'extraction
  - lemme de pompage ("pumping lemma")

- Attention : ce lemme exprime une condition nécessaire (pas suffisante)

  Ne permet pas de prouver que certains langages ne sont pas réguliers

L'approche

- Soit L un langage régulier

L'approche

- Soit L un langage régulier
- D'apres le théorème de Kleene :

  il existe un AFD $A = (Q, \Sigma, \delta, q_0, F)$ tel que $L = L(A)$

  et on suppose que $|Q| = n$ (l'automate A possède n états)

L'approche

- Soit L un langage régulier
- D'apres le théorème de Kleene :

  il existe un AFD $A = (Q, \Sigma, \delta, q_0, F)$ tel que $L = L(A)$

  et on suppose que $|Q| = n$ (l'automate A possède n états)
- Soit $m \in L$ tel que $|m| = z \geq n$ (m est de longueur z)

  on a donc $m = s_1 s_2 \ldots s_z$

L'approche

- Soit L un langage régulier
- D'apres le théorème de Kleene :

  il existe un AFD $A = (Q, \Sigma, \delta, q_0, F)$ tel que $L = L(A)$

  et on suppose que $|Q| = n$ (l'automate A possède n états)
- Soit $m \in L$ tel que $|m| = z \geq n$ (m est de longueur z)

  on a donc $m = s_1 s_2 \dots s_z$
- Comme $m \in L = L(A)$ on a :

  $\forall i, 1 \leq i \leq z, \quad \widehat{\delta}(q_0, s_1 s_2 \dots s_i) = q'_i \in Q$

  - à partir de q_0, avec les i premiers symboles de m on arrive à un i^{eme} état $q'_i \in Q$

L'approche

- Soit L un langage régulier
- D'apres le théorème de Kleene :

  il existe un AFD $A = (Q, \Sigma, \delta, q_0, F)$ tel que $L = L(A)$

  et on suppose que $|Q| = n$ (l'automate A possède n états)
- Soit $m \in L$ tel que $|m| = z \geq n$ (m est de longueur z)

  on a donc $m = s_1 s_2 \dots s_z$
- Comme $m \in L = L(A)$ on a :

  $\forall i, 1 \leq i \leq z, \quad \widehat{\delta}(q_0, s_1 s_2 \dots s_i) = q'_i \in Q$

  - à partir de q_0, avec les i premiers symboles de m
    on arrive à un i^{eme} état $q'_i \in Q$
  - les états successifs pour la reconnaissance de m
    sont donc successivement les états $q_0, q'_1, q'_2, \dots q'_{z-1}$ et q'_z

L'approche

Ce sera plus clair avec des dessins :

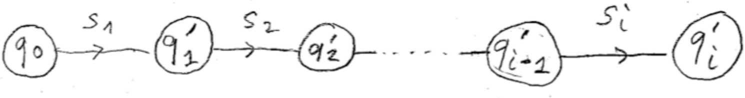
L'approche

Ce sera plus clair avec des dessins :

- Comme $m \in L = L(A)$ on a :

  $\forall i, 1 \leq i \leq z, \quad \widehat{\delta}(q_0, s_1 s_2 \dots s_i) = q_i' \in Q$

  - à partir de q_0, avec les i premiers symboles de $m = s_1 s_2 \dots s_z$ on arrive à un i^{eme} état $q_i' \in Q$



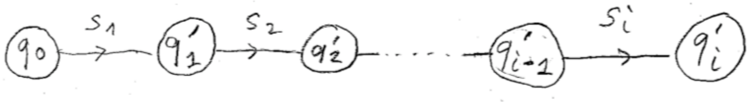
L'approche

Ce sera plus clair avec des dessins :

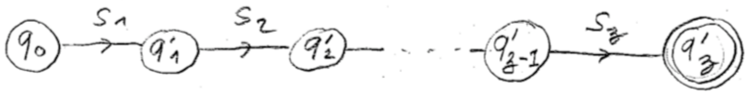
- Comme $m \in L = L(A)$ on a :

  $\forall i, 1 \leq i \leq z, \quad \widehat{\delta}(q_0, s_1 s_2 \ldots s_i) = q'_i \in Q$

  - à partir de q_0, avec les i premiers symboles de $m = s_1 s_2 \ldots s_z$ on arrive à un i^{eme} état $q'_i \in Q$

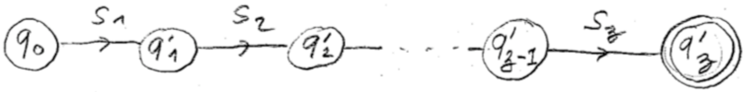


  - les états successifs pour la reconnaissance de $m = s_1 s_2 \ldots s_z$ sont donc successivement les états $q_0, q'_1, q'_2, \ldots q'_{z-1}$ et q'_z



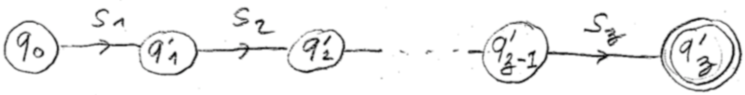
L'approche

- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n + 1$ états considérés



L'approche

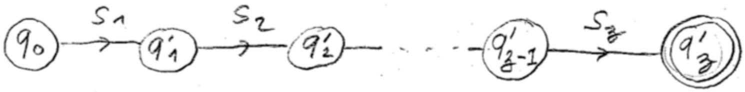
- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n+1$ états considérés



- Avec au moins $n+1$ états considérés, au moins un état est répété

L'approche

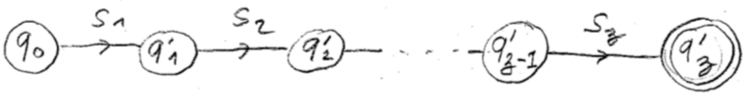
- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n+1$ états considérés



- Avec au moins $n+1$ états considérés, au moins un état est répété donc $\exists\, j, k$ avec $0 \leq j < k \leq z$ tels que $q'_j = q'_k$

L'approche

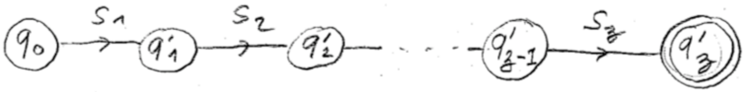
- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n+1$ états considérés



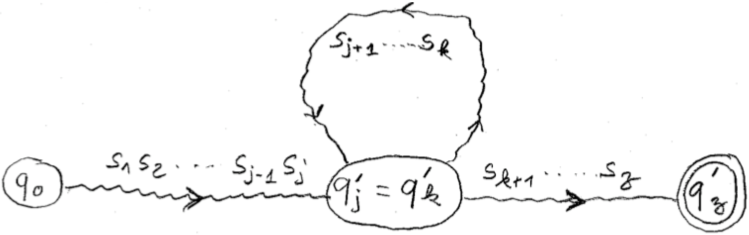
- Avec au moins $n+1$ états considérés, au moins un état est répété donc $\exists\ j, k$ avec $0 \leq j < k \leq z$ tels que $q'_j = q'_k$
- On a une boucle de q'_j vers $q'_j = q'_k$ pour la reconnsaissance de m par A

L'approche

- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n+1$ états considérés

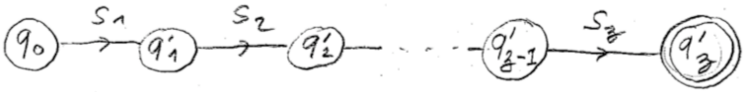


- Avec au moins $n+1$ états considérés, au moins un état est répété donc $\exists\ j, k$ avec $0 \leq j < k \leq z$ tels que $q'_j = q'_k$
- On a une boucle de q'_j vers $q'_j = q'_k$ pour la reconnsaissance de m par A

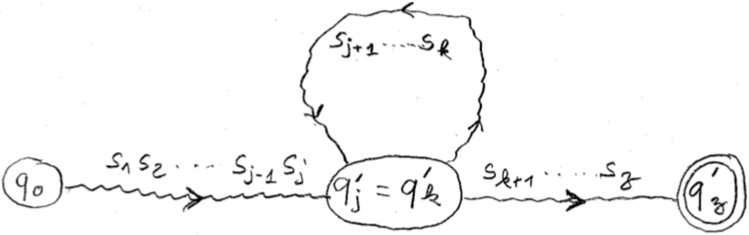


L'approche

- Comme $|m| = z \geq n$, de q_0 à q'_z il y a au moins $n+1$ états considérés



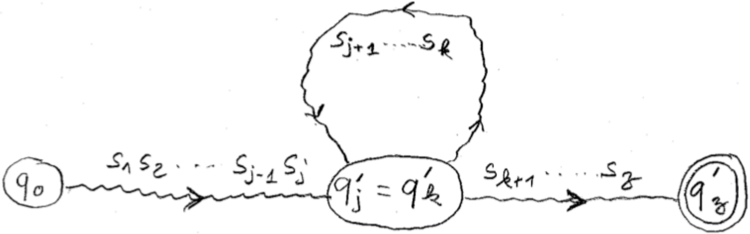
- Avec au moins $n+1$ états considérés, au moins un état est répété donc $\exists\, j, k$ avec $0 \leq j < k \leq z$ tels que $q'_j = q'_k$
- On a une boucle de q'_j vers $q'_j = q'_k$ pour la reconnaissance de m par A



- Donc le mot m est de la forme $m = s_1 s_2 \dots s_j s_{j+1} \dots s_k s_{k+1} \dots s_z$

L'approche

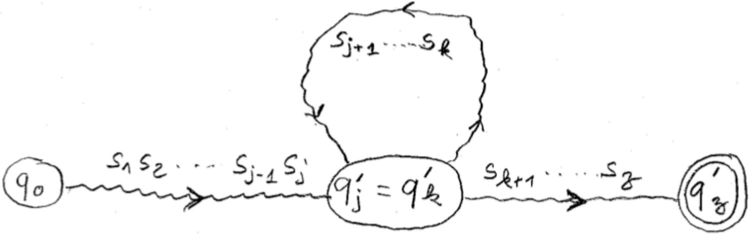
- Le mot m est de la forme $m = s_1 s_2 \dots s_j s_{j+1} \dots s_k s_{k+1} \dots s_z$



avec ce passage dans la boucle

L'approche

- Le mot m est de la forme $m = s_1 s_2 \ldots s_j s_{j+1} \ldots s_k s_{k+1} \ldots s_z$



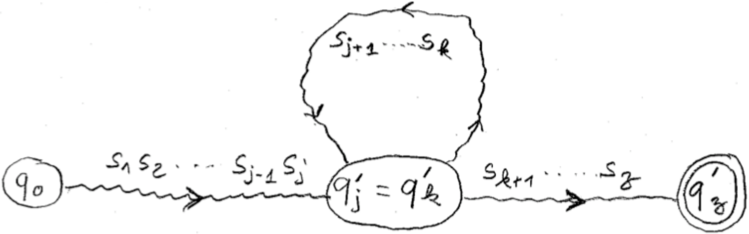
avec ce passage dans la boucle

- Et si on ne passe pas dans la boucle, on arrive quand même à q'_z

  donc le mot $s_1 s_2 \ldots s_j s_{k+1} \ldots s_z \in L$ car il est aussi reconnu par A

L'approche

- Le mot m est de la forme $m = s_1 s_2 \dots s_j s_{j+1} \dots s_k s_{k+1} \dots s_z$



  avec ce passage dans la boucle
- Et si on ne passe pas dans la boucle, on arrive quand même à q'_z donc le mot $s_1 s_2 \dots s_j s_{k+1} \dots s_z \in L$ car il est aussi reconnu par A
- Mais aussi que la répétition i fois de cette boucle est aussi dans L, à savoir que $m = s_1 s_2 \dots s_j (s_{j+1} \dots s_k)^i s_{k+1} \dots s_z \in L$ car il est aussi reconnu par A, et cela, $\forall i \in \mathbb{N}$

Lemme de l'étoile

Lemme du facteur itérant

Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}^*$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

Lemme de l'étoile

Lemme du facteur itérant

Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}^*$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v et w tels que $m = u.v.w$,

Lemme de l'étoile

Lemme du facteur itérant

Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}^*$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$

Lemme de l'étoile

Lemme du facteur itérant

Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}^*$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

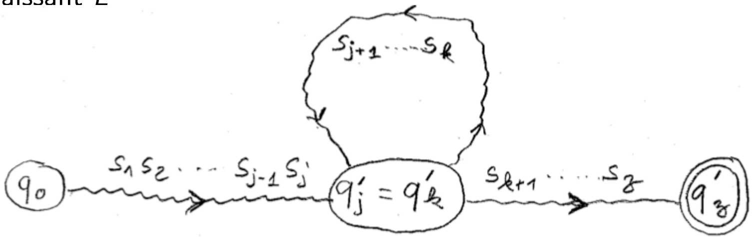
Lemme de l'étoile

Lemme du facteur itérant

Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}^*$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L



Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve :

Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve : en plus de ce qui a été dit avant :

- $m = s_1 s_2 \ldots s_j s_{j+1} \ldots s_k s_{k+1} \ldots s_z$

Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve : en plus de ce qui a été dit avant :

- $m = s_1 s_2 \dots s_j s_{j+1} \dots s_k s_{k+1} \dots s_z$
- $u = s_1 s_2 \dots s_j$

Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve : en plus de ce qui a été dit avant :

- $m = s_1 s_2 \ldots s_j s_{j+1} \ldots s_k s_{k+1} \ldots s_z$
- $u = s_1 s_2 \ldots s_j$
- $v = s_{j+1} \ldots s_k$

Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve : en plus de ce qui a été dit avant :

- $m = s_1 s_2 \ldots s_j s_{j+1} \ldots s_k s_{k+1} \ldots s_z$
- $u = s_1 s_2 \ldots s_j$
- $v = s_{j+1} \ldots s_k$
- $w = s_{k+1} \ldots s_z$ $\square$

Lemme de l'étoile

Lemme du facteur itérant Pour tout langage régulier L, il existe une constante $\alpha \in \mathbb{N}$ (dépend de L) telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe les facteurs u, v et w tels que $m = u.v.w$, et
- (2) $1 \leq |v| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w \in L$

De plus, α n'est pas plus grand que le nombre d'états du plus petit AFD reconnaissant L

Preuve : en plus de ce qui a été dit avant :

- $m = s_1 s_2 \ldots s_j s_{j+1} \ldots s_k s_{k+1} \ldots s_z$
- $u = s_1 s_2 \ldots s_j$
- $v = s_{j+1} \ldots s_k$
- $w = s_{k+1} \ldots s_z$ $\square$

Remarques :

- Si L est fini, tout AFD reconnaissant L est sans boucle et ce lemme n'est pas pas applicable
- Si L est infini, tout AFD reconnaissant L possède au moins une boucle et ce lemme est potentiellement applicable... mais pas toujours (cf. condition nécessaire mais pas suffisante, voir plus loin).

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L\ (|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ ($|m| \geq \alpha$), il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ ($|m| \geq \alpha$), il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent
- $v = a^p.b^q$ avec $1 \leq p, q \leq n$ (donc $m = a^{n-p}.a^p.b^q.b^{n-q}$) :

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ ($|m| \geq \alpha$), il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent
- $v = a^p.b^q$ avec $1 \leq p, q \leq n$ (donc $m = a^{n-p}.a^p.b^q.b^{n-q}$) :
  Donc $v^2 = a^p.b^q.a^p.b^q$

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ ($|m| \geq \alpha$), il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent
- $v = a^p.b^q$ avec $1 \leq p, q \leq n$ (donc $m = a^{n-p}.a^p.b^q.b^{n-q}$) :
  Donc $v^2 = a^p.b^q.a^p.b^q$ et $u.v^2.w = a^{n-p}.a^p.b^q.a^p.b^q.b^{n-q}$

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ $(|m| \geq \alpha)$, il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent
- $v = a^p.b^q$ avec $1 \leq p, q \leq n$ (donc $m = a^{n-p}.a^p.b^q.b^{n-q}$) :
  Donc $v^2 = a^p.b^q.a^p.b^q$ et $u.v^2.w = a^{n-p}.a^p.b^q.a^p.b^q.b^{n-q}$
  ce qui fait que $u.v^2.w \notin L$ d'où une contradiction

Exemple d'application du lemme

Preuve de [*Le langage* $L = \{a^n b^n \mid n \geq 0\}$ *n'est pas régulier*] par l'absurde

Si L est régulier, il existe une constante α telle que : $\forall m \in L$ ($|m| \geq \alpha$), il existe des facteurs u, v et w tels que $m = u.v.w$, avec $\alpha \geq |v| \geq 1$ et $\forall i \geq 0$, on a $u.v^i.w \in L$

On a alors 3 cas (avec $m = a^n b^n$ et $n > 0$ puisque $|v| \geq 1$) :

- $v = a^p$ avec $1 \leq p \leq n$:
  Puisque $|u.v.w|_a = n = |u.v.w|_b$ (car $m = a^n b^n$)
  pour $u.v^2.w$ on aurait $|u.v^2.w|_a = n + p$ et $|u.v^2.w|_b = n$
  et donc $u.v^2.w \notin L$, d'où une contradiction
- $v = b^p$ avec $1 \leq p \leq n$: c'est un cas similaire au précédent
- $v = a^p.b^q$ avec $1 \leq p, q \leq n$ (donc $m = a^{n-p}.a^p.b^q.b^{n-q}$) :
  Donc $v^2 = a^p.b^q.a^p.b^q$ et $u.v^2.w = a^{n-p}.a^p.b^q.a^p.b^q.b^{n-q}$
  ce qui fait que $u.v^2.w \notin L$ d'où une contradiction

On en déduit que n'est pas régulier $\square$.

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$$

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a, b\}^* \text{ et } m' \in \{a, b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

- On prend un mot $m \in L$ tel que $m \in \{a^n b^n \mid n \geq 0\}$ avec $n \geq 1$

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

- On prend un mot $m \in L$ tel que $m \in \{a^n b^n \mid n \geq 0\}$ avec $n \geq 1$
- Ce mot peut se factoriser de la façon suivante :
  $m = uvw$ avec $u = a^{n-1}$, $v = ab$ et $w = b^{n-1}$

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a, b\}^* \text{ et } m' \in \{a, b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

- On prend un mot $m \in L$ tel que $m \in \{a^n b^n \mid n \geq 0\}$ avec $n \geq 1$
- Ce mot peut se factoriser de la façon suivante :
  $m = uvw$ avec $u = a^{n-1}$, $v = ab$ et $w = b^{n-1}$
- Et tout mot de la forme $uv^i w$ avec $i \geq 0$ appartient au langage L car :
  - $uv^0 w$ et $uv^1 w$ appartiennent à $\{a^n b^n \mid n \geq 0\}$

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

- On prend un mot $m \in L$ tel que $m \in \{a^n b^n \mid n \geq 0\}$ avec $n \geq 1$
- Ce mot peut se factoriser de la façon suivante :
  $m = uvw$ avec $u = a^{n-1}$, $v = ab$ et $w = b^{n-1}$
- Et tout mot de la forme uv^iw avec $i \geq 0$ appartient au langage L car :
  - uv^0w et uv^1w appartiennent à $\{a^n b^n \mid n \geq 0\}$
  - et pour $i \geq 2$, $uv^iw \in \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$

Limites d'application du lemme

Un langage qui vérifie les conditions du Lemme sans être régulier :

$$L = \{a^n b^n \mid n \geq 0\} \cup \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$$

Ce langage n'est pas régulier... mais il vérifie les conditions du Lemme :

- On prend un mot $m \in L$ tel que $m \in \{a^n b^n \mid n \geq 0\}$ avec $n \geq 1$
- Ce mot peut se factoriser de la façon suivante :
  $m = uvw$ avec $u = a^{n-1}$, $v = ab$ et $w = b^{n-1}$
- Et tout mot de la forme $uv^i w$ avec $i \geq 0$ appartient au langage L car :
  - $uv^0 w$ et $uv^1 w$ appartiennent à $\{a^n b^n \mid n \geq 0\}$
  - et pour $i \geq 2$, $uv^i w \in \{m.b.a.m' \mid m \in \{a,b\}^* \text{ et } m' \in \{a,b\}^*\}$

Remarque : L n'est pas régulier car :

- $a^* b^*$ est régulier
- si L est régulier, alors $L \cap a^* b^*$ doit être régulier

  (cf. les langages réguliers sont clos par intersection)
- or, $L \cap a^* b^* = \{a^n b^n \mid n \geq 0\}$ qui n'est pas régulier... et donc L ne l'est pas

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$
- On cherche une contradiction en trouvant un entier i tel que l'on peut montrer que $u.v^i.w \notin L$

  (ce i dépend donc de α, u, v et w)

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$
- On cherche une contradiction en trouvant un entier i tel que l'on peut montrer que $u.v^i.w \notin L$ (ce i dépend donc de α, u, v et w)

mais tout cela n'est pas toujours trivial...

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$
- On cherche une contradiction en trouvant un entier i tel que l'on peut montrer que $u.v^i.w \notin L$ (ce i dépend donc de α, u, v et w)

mais tout cela n'est pas toujours trivial... ni toujours possible

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$
- On cherche une contradiction en trouvant un entier i tel que l'on peut montrer que $u.v^i.w \notin L$
  (ce i dépend donc de α, u, v et w)

mais tout cela n'est pas toujours trivial... ni toujours possible
car le lemme exprime une condition nécessaire et pas suffisante

Cas général d'application du lemme

Pour montrer qu'un langage L n'est pas régulier :

- On considère la constante α du lemme
- On choisit un mot $m \in L$ dépendant de la valeur de α
- On écrit m sous la forme $u.v.w$ avec $1 \leq |v| \leq \alpha$
- On cherche une contradiction en trouvant un entier i tel que l'on peut montrer que $u.v^i.w \notin L$
  (ce i dépend donc de α, u, v et w)

mais tout cela n'est pas toujours trivial... ni toujours possible
car le lemme exprime une condition nécessaire et pas suffisante

Mais comment caractériser les langages qui ne sont pas réguliers ?
C'est notamment l'objet de la suite de ce cours !

Plan

Plan

Grammaires formelles

Notions de grammaires formelles

- Un autre moyen de représenter des langages formels que les machines abstraites comme les AFD
- Indiquent comment on peut générer (on dit aussi engendrer) tous les mots du langage qu'elles modélisent.
- Historiquement : ont été définies dans les années 1950 par Noam Chomsky (linguiste professeur au MIT) pour représenter la structure des langues naturelles.
- Appelées à l'origine "grammaires génératives".

Grammaires formelles : Notions relation binaire

Notions relation binaire

- Relation binaire R définie sur $E \times E$:

  Ensemble de couples $(a, b) \in E \times E$
- Une relation binaire R peut être notée $\rightarrow$ ou encore $\underset{R}{\rightarrow}$
- On note $a \underset{R}{\rightarrow} b$ le fait qu'un couple $(a, b) \in R$

Grammaires formelles : Notions relation binaire

Notions relation binaire

- Relation binaire R définie sur $E \times E$:

  Ensemble de couples $(a, b) \in E \times E$
- Une relation binaire R peut être notée $\rightarrow$ ou encore $\underset{R}{\rightarrow}$
- On note $a \underset{R}{\rightarrow} b$ le fait qu'un couple $(a, b) \in R$
- Exemple :
  - $E = \{a, b, c\}$
  - et donc

    $E \times E = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}$
  - et on peut définir une relation $R = \{(a, a), (a, b), (b, a), (b, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermetures de relations binaires

- Soit $\rightarrow$ une relation binaire sur $E \times E$. On définit $\xrightarrow{i}$ par :
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\}$
  - et $\forall i > 0$, on a $\xrightarrow{i} = \xrightarrow{i-1} \mathbf{o} \rightarrow$ où $\mathbf{o}$ est l'opérateur de composition

Grammaires formelles : Fermetures de relations binaires

Fermetures de relations binaires

- Soit $\rightarrow$ une relation binaire sur $E \times E$. On définit $\xrightarrow{i}$ par :
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\}$
  - et $\forall i > 0$, on a $\xrightarrow{i} = \xrightarrow{i-1}$ **o** $\rightarrow$ où **o** est l'opérateur de composition
- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermetures de relations binaires

- Soit $\rightarrow$ une relation binaire sur $E \times E$. On définit $\xrightarrow{i}$ par :
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\}$
  - et $\forall i > 0$, on a $\xrightarrow{i} = \xrightarrow{i-1} \mathbf{o} \rightarrow$ où $\mathbf{o}$ est l'opérateur de composition
- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\} = \{(a, a), (b, b), (c, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermetures de relations binaires

- Soit $\rightarrow$ une relation binaire sur $E \times E$. On définit $\xrightarrow{i}$ par :
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\}$
  - et $\forall i > 0$, on a $\xrightarrow{i} = \xrightarrow{i-1} \mathbf{o} \rightarrow$ où **o** est l'opérateur de composition
- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$
  - $\xrightarrow{0} = \{(x, x) \mid x \in E\} = \{(a, a), (b, b), (c, c)\}$
  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$

Grammaires formelles : Fermetures de relations binaires

Fermetures de relations binaires

- Soit $\rightarrow$ une relation binaire sur $E \times E$. On définit $\stackrel{i}{\rightarrow}$ par :
  - $\stackrel{0}{\rightarrow} = \{(x, x) \mid x \in E\}$
  - et $\forall i > 0$, on a $\stackrel{i}{\rightarrow} = \stackrel{i-1}{\longrightarrow} \mathbf{o} \rightarrow$ où $\mathbf{o}$ est l'opérateur de composition
- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$
  - $\stackrel{0}{\rightarrow} = \{(x, x) \mid x \in E\} = \{(a, a), (b, b), (c, c)\}$
  - $\stackrel{1}{\rightarrow} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$
  - $\stackrel{2}{\rightarrow} = \{(a, a), (a, b), (a, c), (b, a), (b, b)\}$ car :
    - $a \stackrel{2}{\rightarrow} a$ car $a \rightarrow a \rightarrow a$ et aussi $a \rightarrow b \rightarrow a$
    - $a \stackrel{2}{\rightarrow} b$ car $a \rightarrow a \rightarrow b$
    - $a \stackrel{2}{\rightarrow} c$ car $a \rightarrow b \rightarrow c$
    - $b \stackrel{2}{\rightarrow} a$ car $b \rightarrow a \rightarrow a$
    - $b \stackrel{2}{\rightarrow} b$ car $b \rightarrow a \rightarrow b$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a,a),(a,b),(b,a),(b,c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{+} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$ car

  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{+} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$ car

  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$
  - $\xrightarrow{2} = \{(a, a), (a, b), (a, c), (b, a), (b, b)\}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{+} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$ car

  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$
  - $\xrightarrow{2} = \{(a, a), (a, b), (a, c), (b, a), (b, b)\}$
  - $\xrightarrow{3} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{+} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$ car

  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$
  - $\xrightarrow{2} = \{(a, a), (a, b), (a, c), (b, a), (b, b)\}$
  - $\xrightarrow{3} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$
  - $\xrightarrow{4} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{+} = \bigcup_{i>0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur au moins 1)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{+} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$ car

  - $\xrightarrow{1} = \{(a, a), (a, b), (b, a), (b, c)\} = \rightarrow$
  - $\xrightarrow{2} = \{(a, a), (a, b), (a, c), (b, a), (b, b)\}$
  - $\xrightarrow{3} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$
  - $\xrightarrow{4} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c)\}$
  - etc...

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive et réflexive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{*} = \bigcup_{i \geq 0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur même nulle)

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive et réflexive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{*} = \bigcup_{i \geq 0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur même nulle)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive et réflexive $\xrightarrow{+}$ d'une relation $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{*} = \bigcup_{i \geq 0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur même nulle)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{*} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, c)\}$

  car on a rajouté (c, c) à $\xrightarrow{+}$ car $(c, c) \in \xrightarrow{0}$

Grammaires formelles : Fermetures de relations binaires

Fermeture transitive et réflexive $\xrightarrow{+}$ **d'une relation** $\rightarrow$:

- C'est l'ensemble des couples $\xrightarrow{*} = \bigcup_{i \geq 0} \xrightarrow{i}$

  (on prend toutes les extrémités de "chemins" de longueur même nulle)

- Exemple : si $E = \{a, b, c\}$ et $\rightarrow = \{(a, a), (a, b), (b, a), (b, c)\}$

  $\xrightarrow{*} = \{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, c)\}$

  car on a rajouté (c, c) à $\xrightarrow{+}$ car $(c, c) \in \xrightarrow{0}$

on l'appelle parfois **fermeture réflexo-transitive** :

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture défini sur un alphabet Σ :

- C'est est une relation finie S incluse dans $\Sigma^* \times \Sigma^*$ (couples de mots)

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture défini sur un alphabet Σ :

- C'est est une relation finie S incluse dans $\Sigma^* \times \Sigma^*$ (couples de mots)
- Un système S induit une relation binaire (inifinie) $\underset{R}{\rightarrow}$ définie ainsi :

$$\underset{R}{\rightarrow} = \{ \ (m, m') \in \Sigma^* \times \Sigma^* \mid$$
$$\exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture défini sur un alphabet Σ :

- C'est est une relation finie S incluse dans $\Sigma^* \times \Sigma^*$ (couples de mots)
- Un système S induit une relation binaire (inifinie) $\underset{R}{\rightarrow}$ définie ainsi :

$$\underset{R}{\rightarrow} = \{\ (m, m') \in \Sigma^* \times \Sigma^* \mid \exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

- Les couples de $\underset{R}{\rightarrow}$ sont tous ceux qui s'obtiennent à partir de tout couple de S en ajoutant le même préfixe et le même suffixe au 1^{er} et au 2^{eme} élément d'un couple de S

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture défini sur un alphabet Σ :

- C'est est une relation finie S incluse dans $\Sigma^* \times \Sigma^*$ (couples de mots)
- Un système S induit une relation binaire (inifinie) $\underset{R}{\rightarrow}$ définie ainsi :

  $$\underset{R}{\rightarrow} = \{\ (m, m') \in \Sigma^* \times \Sigma^* \mid \exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

- Les couples de $\underset{R}{\rightarrow}$ sont tous ceux qui s'obtiennent à partir de tout couple de S en ajoutant le même préfixe et le même suffixe au 1^{er} et au 2^{eme} élément d'un couple de S
- Illustration : Si on a $(x, y) \in S$, alors un mot $m = uxv \in \Sigma^*$ se réécrit en un mot $m' = uyv \in \Sigma^*$ et on a donc $m = uxv \underset{R}{\rightarrow} m' = uyv$

Grammaires formelles : Systèmes de réécriture

Exemple : soit le système $S = \{(aa, bb)\}$ défini sur $\Sigma = \{a, b\}$:

- S est bien une relation binaire

Grammaires formelles : Systèmes de réécriture

Exemple : soit le système $S = \{(aa, bb)\}$ **défini sur** $\Sigma = \{a, b\}$ **:**

- S est bien une relation binaire
- Quelle est la relation binaire $\underset{R}{\rightarrow}$ induite par le système S ?

Grammaires formelles : Systèmes de réécriture

Exemple : soit le système $S = \{(aa, bb)\}$ **défini sur** $\Sigma = \{a, b\}$ **:**

- S est bien une relation binaire
- Quelle est la relation binaire $\underset{R}{\rightarrow}$ induite par le système S ?
- Par définition on a :

$$\underset{R}{\rightarrow} = \{\ (m, m') \in \Sigma^* \times \Sigma^* \mid$$

$$\exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

Grammaires formelles : Systèmes de réécriture

Exemple : soit le système $S = \{(aa, bb)\}$ défini sur $\Sigma = \{a, b\}$:

- S est bien une relation binaire
- Quelle est la relation binaire $\underset{R}{\rightarrow}$ induite par le système S ?
- Par définition on a :

  $$\underset{R}{\rightarrow} = \{\ (m, m') \in \Sigma^* \times \Sigma^* \mid$$
  $$\exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

- Prenons $m = u\mathbf{x}v$ avec $u = b, v = bab$

  Avec **x=aa** et **y=bb**, $m = b\mathbf{aa}bab$ se réécrit en $m' = b\mathbf{bb}bab$

  car on remplace $\mathbf{x} = \mathbf{aa}$ par $\mathbf{y} = \mathbf{bb}$

Grammaires formelles : Systèmes de réécriture

Exemple : soit le système $S = \{(aa, bb)\}$ **défini sur** $\Sigma = \{a, b\}$ **:**

- S est bien une relation binaire
- Quelle est la relation binaire $\underset{R}{\rightarrow}$ induite par le système S ?
- Par définition on a :

$$\underset{R}{\rightarrow} = \{\ (m, m') \in \Sigma^* \times \Sigma^* \mid \exists u, v \in \Sigma^* : m = uxv \text{ et } m' = uyv \text{ et } (x, y) \in S\}$$

- Prenons $m = u\mathbf{x}v$ avec $u = b, v = bab$

  Avec **x=aa** et **y=bb**, $m = b\mathbf{aa}bab$ se réécrit en $m' = b\mathbf{bb}bab$

  car on remplace $\mathbf{x} = \mathbf{aa}$ par $\mathbf{y} = \mathbf{bb}$

- Les couples de $\underset{R}{\rightarrow}$ sont donc :

$$\underset{R}{\rightarrow} = \{(\mathbf{aa}, \mathbf{bb}), (a\mathbf{aa}, a\mathbf{bb}), (\mathbf{aa}a, \mathbf{bb}a), (b\mathbf{aa}, b\mathbf{bb}), (\mathbf{aa}b, \mathbf{bb}b), (aa\mathbf{aa}, aa\mathbf{bb}), (\mathbf{aa}aa, \mathbf{bb}aa), (ab\mathbf{aa}, ab\mathbf{bb}), (\mathbf{aa}ab, \mathbf{bb}ab), (ba\mathbf{aa}, ba\mathbf{bb}), (\mathbf{aa}ba, \mathbf{bb}ba), \ldots\}$$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \overset{*}{\underset{R}{\rightarrow}} m'\}$$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
(disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \overset{*}{\underset{R}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \underset{R}{\overset{*}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

  Le langage engendré est $L = \{m' \in \Sigma^* \mid ab \underset{R}{\overset{*}{\rightarrow}} m'\} = \{a^n b^n \mid n \geq 1\}$:

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \overset{*}{\underset{R}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

  Le langage engendré est $L = \{m' \in \Sigma^* \mid ab \overset{*}{\underset{R}{\rightarrow}} m'\} = \{a^n b^n \mid n \geq 1\}$:

  - $ab \in L$ car $\quad ab \overset{0}{\underset{R}{\rightarrow}} ab$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \overset{*}{\underset{R}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

  Le langage engendré est $L = \{m' \in \Sigma^* \mid ab \overset{*}{\underset{R}{\rightarrow}} m'\} = \{a^n b^n \mid n \geq 1\}$:
  - $ab \in L$ car $\quad ab \overset{0}{\underset{R}{\rightarrow}} ab$
  - $aabb \in L$ car $\quad ab \overset{1}{\underset{R}{\rightarrow}} aabb$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \underset{R}{\overset{*}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

  Le langage engendré est $L = \{m' \in \Sigma^* \mid ab \underset{R}{\overset{*}{\rightarrow}} m'\} = \{a^n b^n \mid n \geq 1\}$:
  - $ab \in L$ car $ab \underset{R}{\overset{0}{\rightarrow}} ab$
  - $aabb \in L$ car $ab \underset{R}{\overset{1}{\rightarrow}} aabb$
  - $aaabbb \in L$ car $ab \underset{R}{\overset{1}{\rightarrow}} aabb \underset{R}{\overset{1}{\rightarrow}} aaabbb$ et donc $ab \underset{R}{\overset{2}{\rightarrow}} aaabbb$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour générer (ou engendrer) un langage

- Soit S un système de réécriture défini sur Σ et $D \subseteq \Sigma^*$
  (disons pour le moment que D est le langage d'origine)

  S permet de générer un langage L à partir du langage D
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $$L = \{m' \in \Sigma^* \mid \exists m \in D : m \underset{R}{\overset{*}{\rightarrow}} m'\}$$

- Exemple : avec $\Sigma = \{a, b\}, S = \{(ab, aabb)\}$ et $D = \{ab\}$

  Le langage engendré est $L = \{m' \in \Sigma^* \mid ab \underset{R}{\overset{*}{\rightarrow}} m'\} = \{a^n b^n \mid n \geq 1\}$:
  - $ab \in L$ car $ab \underset{R}{\overset{0}{\rightarrow}} ab$
  - $aabb \in L$ car $ab \underset{R}{\overset{1}{\rightarrow}} aabb$
  - $aaabbb \in L$ car $ab \underset{R}{\overset{1}{\rightarrow}} aabb \underset{R}{\overset{1}{\rightarrow}} aaabbb$ et donc $ab \underset{R}{\overset{2}{\rightarrow}} aaabbb$
  - etc.

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$

  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} =$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \underset{R}{\overset{*}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \underset{R}{\overset{*}{\rightarrow}} b\} = \{b\}$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\}$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \underset{R}{\overset{*}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \underset{R}{\overset{*}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\} \cup \{qa^{2n} \mid n \geq 0\}$:

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\} \cup \{qa^{2n} \mid n \geq 0\}$:

  - mot b reconnu car aucune application de la réécriture (cf. $F = \{b\}$)

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\} \cup \{qa^{2n} \mid n \geq 0\}$:

  - mot b reconnu car aucune application de la réécriture (cf. $F = \{b\}$)
  - mot $qa^{2\times 0} = qa^0 = q$ reconnu car $q \overset{1}{\underset{R}{\rightarrow}} b$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$
  (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F
  avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\} \cup \{qa^{2n} \mid n \geq 0\}$:

  - mot b reconnu car aucune application de la réécriture (cf. $F = \{b\}$)
  - mot $qa^{2\times 0} = qa^0 = q$ reconnu car $q \overset{1}{\underset{R}{\rightarrow}} b$
  - mot $pa^{2\times 0+1} = pa^1 = pa$ reconnu car $pa \overset{1}{\underset{R}{\rightarrow}} q \overset{1}{\underset{R}{\rightarrow}} b$

Grammaires formelles : Systèmes de réécriture

Systèmes de réécriture pour reconnaître un langage

- Soit S un système de réécriture défini sur Σ et $F \subseteq \Sigma^*$ (disons pour le moment que F est le langage d'arrivée)

  S permet de reconnaître un langage L à partir du langage F avec la relation binaire $\underset{R}{\rightarrow}$ induite par le système S :

  $L = \{m \in \Sigma^* \mid \exists m' \in F : m \overset{*}{\underset{R}{\rightarrow}} m'\}$

- Exemple : $\Sigma = \{a, b, p, q\}, S = \{(pa, q), (qa, p), (q, b)\}$ et $F = \{b\}$

  $L = \{m \in \Sigma^* \mid m \overset{*}{\underset{R}{\rightarrow}} b\} = \{b\} \cup \{pa^{2n+1} \mid n \geq 0\} \cup \{qa^{2n} \mid n \geq 0\}$:

  - mot b reconnu car aucune application de la réécriture (cf. $F = \{b\}$)
  - mot $qa^{2\times 0} = qa^0 = q$ reconnu car $q \overset{1}{\underset{R}{\rightarrow}} b$
  - mot $pa^{2\times 0+1} = pa^1 = pa$ reconnu car $pa \overset{1}{\underset{R}{\rightarrow}} q \overset{1}{\underset{R}{\rightarrow}} b$
  - mot $qa^{2\times 1} = qa^2 = qaa$ reconnu car $qaa \overset{1}{\underset{R}{\rightarrow}} pa \overset{1}{\underset{R}{\rightarrow}} q \overset{1}{\underset{R}{\rightarrow}} b$
  - etc.

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)
- P : ensemble fini de règles de production de la forme $\alpha \to \beta$

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)
- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ et

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)
- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ et
  - $\beta \in (N \cup T)^*$

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)
- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ et
  - $\beta \in (N \cup T)^*$

  NB. 1. on a au moins un non terminal en partie gauche de chaque règle
  NB. 2. règles de production : ce sont des règles de réécriture

Grammaires formelles : Définition

Grammaires formelles

Une grammaire G est définie par un quadruplet $G = (N, T, P, S)$ où :

- N : ensemble fini de symboles dits non terminaux
  (par convention, on notera ces symboles par des lettres majuscules)
- T : ensemble fini de symboles dits terminaux ($T \cap N = \emptyset$)
  (par convention, on notera ces symboles par des lettres minuscules)
- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ et
  - $\beta \in (N \cup T)^*$

  NB. 1. on a au moins un non terminal en partie gauche de chaque règle
  NB. 2. règles de production : ce sont des règles de réécriture
- $S \in N$: symbole non terminal appelé axiome
  (c'est le point de départ, i.e. d'origine des dérivations de la grammaire)

Grammaires formelles : Définition

Exemple de grammaire formelle : $G = (N, T, P, S)$ où :

Grammaires formelles : Définition

Exemple de grammaire formelle : $G = (N, T, P, S)$ où :

- $N = \{S, A, B\}$
- $T = \{a, b\}$
- $P = \{S \rightarrow A, S \rightarrow B, A \rightarrow aA, A \rightarrow a, B \rightarrow bB, B \rightarrow b\}$

  La notation de P peut aussi être fournie en notation compacte par

  $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$
- $S \in N$ est donc l'axiome

Grammaires formelles : Illustration

Exemple de règles de production du "monde réel" :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2eme édition, page 241)

instruction-de-sélection :
  `if` (*expression*) *instruction*
  `if` (*expression*) *instruction* `else` *instruction*
  `switch` (*expression*) *instruction*

instruction-d'itération :
  `while` (*expression*) *instruction*
  `do` *instruction* `while` (*expression*) ;
  `for` ($expression_{opt}$; $expression_{opt}$; $expression_{opt}$) *instruction*

Grammaires formelles : Illustration

Exemple de règles de production du "monde réel" :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2^{eme} édition, page 241)

instruction-de-sélection :
  `if` (*expression*) *instruction*
  `if` (*expression*) *instruction* `else` *instruction*
  `switch` (*expression*) *instruction*

instruction-d'itération :
  `while` (*expression*) *instruction*
  `do` *instruction* `while` (*expression*) ;
  `for` ($expression_{opt}$; $expression_{opt}$; $expression_{opt}$) *instruction*

- N contient : *instruction-de-sélection*, *expression*, *instruction*, *instruction-d'itération* et $expression_{opt}$

Grammaires formelles : Illustration

Exemple de règles de production du "monde réel" :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2eme édition, page 241)

instruction-de-sélection :
    `if` (*expression*) *instruction*
    `if` (*expression*) *instruction* `else` *instruction*
    `switch` (*expression*) *instruction*

instruction-d'itération :
    `while` (*expression*) *instruction*
    `do` *instruction* `while` (*expression*) ;
    `for` ($expression_{opt}$; $expression_{opt}$; $expression_{opt}$) *instruction*

- N contient : *instruction-de-sélection*, *expression*, *instruction*, *instruction-d'itération* et $expression_{opt}$
- T contient `if`, `(`, `)`, `else`, `switch`, `while`, `do`, `;`, et `for`

Grammaires formelles : Illustration

Exemple de règles de production du "monde réel" :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2^{eme} édition, page 241)

instruction-de-sélection :
    `if` `(` *expression* `)` *instruction*
    `if` `(` *expression* `)` *instruction* `else` *instruction*
    `switch` `(` *expression* `)` *instruction*

instruction-d'itération :
    `while` `(` *expression* `)` *instruction*
    `do` *instruction* `while` `(` *expression* `)` `;`
    `for` `(` $expression_{opt}$ `;` $expression_{opt}$ `;` $expression_{opt}$ `)` *instruction*

- N contient : *instruction-de-sélection*, *expression*, *instruction*, *instruction-d'itération* et $expression_{opt}$
- T contient `if`, `(`, `)`, `else`, `switch`, `while`, `do`, `;`, et `for`
- P : les flèches ($\rightarrow$) des règles de production sont remplacées par des :

  et les | de la notation compacte par des retours à la ligne

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot (une dérivation réécrit donc un mot en un autre mot)

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot
  (une dérivation réécrit donc un mot en un autre mot)

- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot (une dérivation réécrit donc un mot en un autre mot)
- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :
  - α possède un facteur X dans $(N \cup T)^*.N.(N \cup T)^*$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot (une dérivation réécrit donc un mot en un autre mot)
- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :
  - α possède un facteur X dans $(N \cup T)^*.N.(N \cup T)^*$
  - β possède un facteur Y dans $(N \cup T)^*$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot
  (une dérivation réécrit donc un mot en un autre mot)
- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :
  - α possède un facteur X dans $(N \cup T)^*.N.(N \cup T)^*$
  - β possède un facteur Y dans $(N \cup T)^*$
  - il existe d'une règle $r \in P$ de la forme $X \rightarrow Y$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot (une dérivation réécrit donc un mot en un autre mot)

- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :
  - α possède un facteur X dans $(N \cup T)^*.N.(N \cup T)^*$
  - β possède un facteur Y dans $(N \cup T)^*$
  - il existe d'une règle $r \in P$ de la forme $X \to Y$

  et cette dérivation est remplacé X dans α par Y pour former β

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- Une dérivation est l'application d'une règle de production à un mot (une dérivation réécrit donc un mot en un autre mot)

- Une dérivation notée $\alpha \underset{(r)}{\Rightarrow} \beta$ est possible si :
  - α possède un facteur X dans $(N \cup T)^*.N.(N \cup T)^*$
  - β possède un facteur Y dans $(N \cup T)^*$
  - il existe d'une règle $r \in P$ de la forme $X \rightarrow Y$

  et cette dérivation est remplacé X dans α par Y pour former β

- On note $\underset{(G)}{\overset{*}{\Rightarrow}}$ la fermeture transitive de $\Rightarrow$ sur P

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  On a :

  $$S \underset{(S \to A)}{\Rightarrow} A$$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a :

  $$S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA$$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

On a :

$$S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA$$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a :

$$S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA \underset{(A \to aA)}{\Rightarrow} aaA \underset{(A \to aA)}{\Rightarrow} aaaA$$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

On a :

$$S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(A \rightarrow aA)}{\Rightarrow} aaaA \underset{(A \rightarrow aA)}{\Rightarrow} aaaa$$

Grammaires formelles : Dérivation

Dérivation : comment produire un mot avec une grammaire

- **Exemple** avec la grammaire $G = (N, T, P, S)$ vue plus haut où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a :

  $$S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(A \rightarrow aA)}{\Rightarrow} aaaA \underset{(A \rightarrow aA)}{\Rightarrow} aaaa$$

  et donc on peut écrire :

  $$S \underset{(G)}{\overset{*}{\Rightarrow}} aaaa$$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa \ldots$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \rightarrow A)}{\Rightarrow} A$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA \underset{(A \to aA)}{\Rightarrow} aaA$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

- $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(A \rightarrow a)}{\Rightarrow} aaa$

ou bien

- $S \underset{(G)}{\overset{*}{\Rightarrow}} bbb\ldots$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(A \rightarrow a)}{\Rightarrow} aaa$

  ou bien

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} bbb\ldots$ car $S \underset{(S \rightarrow B)}{\Rightarrow} B$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S\to A)}{\Rightarrow} A \underset{(A\to aA)}{\Rightarrow} aA \underset{(A\to aA)}{\Rightarrow} aaA \underset{(A\to a)}{\Rightarrow} aaa$

  ou bien

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} bbb\ldots$ car $S \underset{(S\to B)}{\Rightarrow} B \underset{(B\to bB)}{\Rightarrow} bB$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(A \rightarrow a)}{\Rightarrow} aaa$

  ou bien

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} bbb\ldots$ car $S \underset{(S \rightarrow B)}{\Rightarrow} B \underset{(B \rightarrow bB)}{\Rightarrow} bB \underset{(B \rightarrow bB)}{\Rightarrow} bbB$

Grammaires formelles : langage engendré

Langage engendré par une grammaire :

- **Définition :** Le langage $L(G)$ engendré par une grammaire

  $G = (N, T, P, S)$ est défini par $L(G) = \{m \in T^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  On a $L(G) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ car :

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} aaa\ldots$ car $S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA \underset{(A \to aA)}{\Rightarrow} aaA \underset{(A \to a)}{\Rightarrow} aaa$

  ou bien

  - $S \underset{(G)}{\overset{*}{\Rightarrow}} bbb\ldots$ car $S \underset{(S \to B)}{\Rightarrow} B \underset{(B \to bB)}{\Rightarrow} bB \underset{(B \to bB)}{\Rightarrow} bbB \underset{(B \to b)}{\Rightarrow} bbb$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  $$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
  $$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \overset{*}{\underset{(G)}{\Rightarrow}} m\}$

  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  $$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
  $$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

  - $S \underset{(S \to A)}{\Rightarrow} A$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  $\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$
  $\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\}$ car

  - $S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

$$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
$$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

  - $S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA \underset{(A \to aA)}{\Rightarrow} aaA$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  $$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
  $$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

  - $S \underset{(S \to A)}{\Rightarrow} A \underset{(A \to aA)}{\Rightarrow} aA \underset{(A \to aA)}{\Rightarrow} aaA \underset{(G)}{\overset{*}{\Rightarrow}} aa \ldots aaA$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$

  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow A|B, A \rightarrow aA|a, B \rightarrow bB|b\}$

  $$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
  $$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

  - $S \underset{(S \rightarrow A)}{\Rightarrow} A \underset{(A \rightarrow aA)}{\Rightarrow} aA \underset{(A \rightarrow aA)}{\Rightarrow} aaA \underset{(G)}{\overset{*}{\Rightarrow}} aa \ldots aaA \underset{(A \rightarrow a)}{\Rightarrow} aa \ldots aaa$

Grammaires formelles : langage élargi engendré

Langage élargi (ou étendu) engendré par une grammaire :

- **Définition :** Le langage élargi $\hat{L}(G)$ engendré par une grammaire $G = (N, T, P, S)$ est défini par $\hat{L}(G) = \{m \in (N \cup T)^* \mid S \underset{(G)}{\overset{*}{\Rightarrow}} m\}$
  (on considère aussi les mots dits *proto-mots* faits avec des non-terminaux)
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S, A, B\}$
  - $T = \{a, b\}$
  - $P = \{S \to A|B, A \to aA|a, B \to bB|b\}$

  $$\hat{L}(G) = \{S\} \cup \{A\} \cup \{a^n A^p \mid 1 \leq n, 0 \leq p \leq 1\}$$
  $$\cup \{B\} \cup \{b^n B^p \mid 1 \leq n, 0 \leq p \leq 1\} \text{ car}$$

  - $S \underset{(S\to A)}{\Rightarrow} A \underset{(A\to aA)}{\Rightarrow} aA \underset{(A\to aA)}{\Rightarrow} aaA \underset{(G)}{\overset{*}{\Rightarrow}} aa\ldots aaA \underset{(A\to a)}{\Rightarrow} aa\ldots aaa$

  ou bien
  - $S \underset{(S\to B)}{\Rightarrow} B \underset{(B\to bB)}{\Rightarrow} bB \underset{(B\to bB)}{\Rightarrow} bbB \underset{(G)}{\overset{*}{\Rightarrow}} bb\ldots bbB \underset{(B\to b)}{\Rightarrow} bb\ldots bbb$

Grammaires formelles : dérivations et arborescences

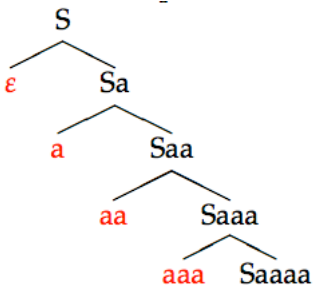
On peut représenter les dérivations par des arborescences :

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a\}$
  - $P = \{S \rightarrow Sa|\varepsilon\}$

Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a\}$
  - $P = \{S \rightarrow Sa|\varepsilon\}$
- Arborescence des dérivations (les feuilles sont des proto-mots) :



avec $L(G) = \{a^n \mid n \geq 0\}$ et $\hat{L}(G) = \{a^n \mid n \geq 0\} \cup \{Sa^n \mid n \geq 0\}$

Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

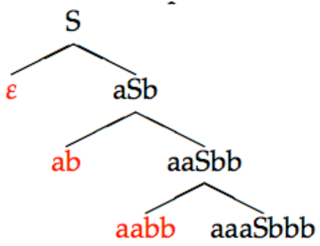
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSb|\varepsilon\}$

Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSb|\varepsilon\}$

- Arborescence des dérivations (les feuilles sont des proto-mots) :



avec $L(G) = \{a^n b^n \mid n \geq 0\}$ et $\hat{L}(G) = \{a^n b^n \mid n \geq 0\} \cup \{a^n S b^n \mid n \geq 0\}$

Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

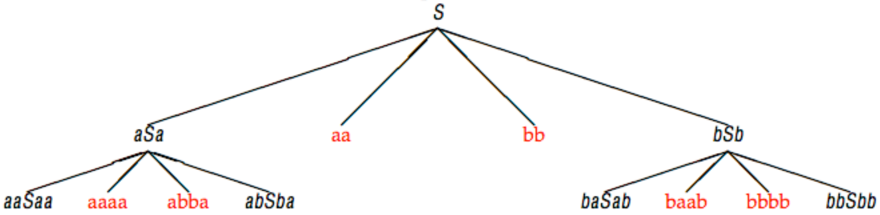
- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSa|bSb|aa|bb\}$

Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSa|bSb|aa|bb\}$

- Arborescence des dérivations (les feuilles sont des proto-mots) :

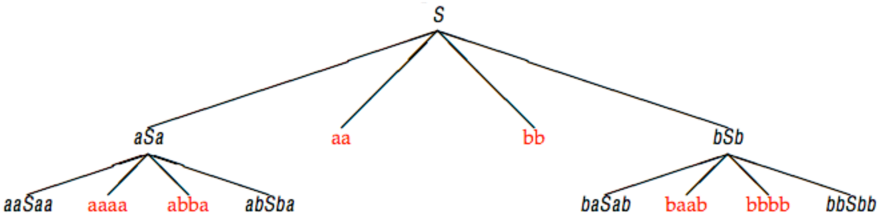


Grammaires formelles : dérivations et arborescences

On peut représenter les dérivations par des arborescences :

- **Exemple** avec la grammaire $G = (N, T, P, S)$ où :
  - $N = \{S\}$
  - $T = \{a, b\}$
  - $P = \{S \rightarrow aSa|bSb|aa|bb\}$

- Arborescence des dérivations (les feuilles sont des proto-mots) :



NB. Cette arborescence (inifinie) représente toutes les dérivations possibles

Grammaires formelles : notion d'arbre de dérivation

Définition : arbres de dérivation (ou "parse trees")

Étant donnée une grammaire $G = (N, T, P, S)$, les arbres de dérivation pour G sont des arborescences (donc enracinées) planaires (on dit aussi ordonnées car lus de la gauche vers la droite) telles que :

Grammaires formelles : notion d'arbre de dérivation

Définition : arbres de dérivation (ou "parse trees")

Étant donnée une grammaire $G = (N, T, P, S)$, les arbres de dérivation pour G sont des arborescences (donc enracinées) planaires (on dit aussi ordonnées car lus de la gauche vers la droite) telles que :

- chaque nœud intérieur est étiqueté par un élément de N

Grammaires formelles : notion d'arbre de dérivation

Définition : arbres de dérivation (ou "parse trees")

Étant donnée une grammaire $G = (N, T, P, S)$, les arbres de dérivation pour G sont des arborescences (donc enracinées) planaires (on dit aussi ordonnées car lus de la gauche vers la droite) telles que :

- chaque nœud intérieur est étiqueté par un élément de N
- chaque feuille est étiquetée par un élément de $N \cup T \cup \{\varepsilon\}$

Grammaires formelles : notion d'arbre de dérivation

Définition : arbres de dérivation (ou "parse trees")

Étant donnée une grammaire $G = (N, T, P, S)$, les arbres de dérivation pour G sont des arborescences (donc enracinées) planaires (on dit aussi ordonnées car lus de la gauche vers la droite) telles que :

- chaque nœud intérieur est étiqueté par un élément de N
- chaque feuille est étiquetée par un élément de $N \cup T \cup \{\varepsilon\}$
- si un nœud intérieur est étiqueté par A et ses enfants sont étiquetés de la gauche vers la droite par $B_1, B_2, \ldots B_k$, alors il existe une règle de P de la forme $A \to B_1 B_2 \ldots B_k$

Grammaires formelles : notion d'arbre de dérivation

Définition : arbres de dérivation (ou "parse trees")

Étant donnée une grammaire $G = (N, T, P, S)$, les arbres de dérivation pour G sont des arborescences (donc enracinées) planaires (on dit aussi ordonnées car lus de la gauche vers la droite) telles que :

- chaque nœud intérieur est étiqueté par un élément de N
- chaque feuille est étiquetée par un élément de $N \cup T \cup \{\varepsilon\}$
- si un nœud intérieur est étiqueté par A et ses enfants sont étiquetés de la gauche vers la droite par $B_1, B_2, \ldots B_k$, alors il existe une règle de P de la forme $A \rightarrow B_1 B_2 \ldots B_k$

Exemple avec la grammaire $G = (N, T, P, E)$ où :

- $N = \{E, T, F\}$
- $T = \{+, *, a\}$
- $P = \{E \rightarrow T + E | T, T \rightarrow F * T | F, F \rightarrow a\}$
- E est l'axiome !

Remarque : ici, on peut avoir 2 non-terminaux à droite

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

- $N = \{E, T, F\}$
- $T = \{+, *, a\}$
- $P = \{E \rightarrow T + E | T, T \rightarrow F * T | F, F \rightarrow a\}$
- E est l'axiome (et non S !)

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

- $N = \{E, T, F\}$
- $T = \{+, *, a\}$
- $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$
- E est l'axiome (et non S !)

Avec la dérivation :

$$E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(T \rightarrow F)}{\Rightarrow} F + E \underset{(F \rightarrow a)}{\Rightarrow} a + E \underset{(E \rightarrow T)}{\Rightarrow} a + T \underset{(T \rightarrow F*T)}{\Rightarrow} a + F * T$$

Grammaires formelles : notion d'arbre de dérivation

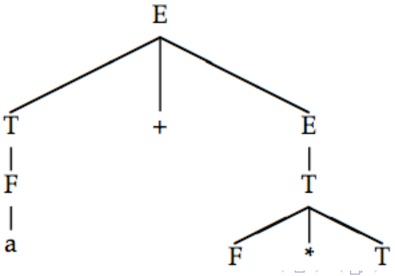
Exemple avec la grammaire $G = (N, T, P, E)$ où :

- $N = \{E, T, F\}$
- $T = \{+, *, a\}$
- $P = \{E \to T + E | T, T \to F * T | F, F \to a\}$
- E est l'axiome (et non S !)

Avec la dérivation :

$$E \underset{(E \to T+E)}{\Rightarrow} T + E \underset{(T \to F)}{\Rightarrow} F + E \underset{(F \to a)}{\Rightarrow} a + E \underset{(E \to T)}{\Rightarrow} a + T \underset{(T \to F*T)}{\Rightarrow} a + F * T$$

On obtient l'arbre de dérivation :



Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \to T + E | T, T \to F * T | F, F \to a\}$

Plusieurs non-terminaux en partie droite des productions

$\Rightarrow$ plusieurs dérivations conduisent au même proto-mot

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \to T + E | T, T \to F * T | F, F \to a\}$

Plusieurs non-terminaux en partie droite des productions

$\Rightarrow$ plusieurs dérivations conduisent au même proto-mot

- $E \underset{(E \to T+E)}{\Rightarrow} T + E \underset{(T \to F)}{\Rightarrow} F + E \underset{(F \to a)}{\Rightarrow} a + E \underset{(E \to T)}{\Rightarrow} a + T \underset{(T \to F*T)}{\Rightarrow} a + F * T$
  $\underset{(F \to a)}{\Rightarrow} a + a * T \underset{(T \to F)}{\Rightarrow} a + a * F \underset{(F \to a)}{\Rightarrow} a + a * a$

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$

Plusieurs non-terminaux en partie droite des productions

$\Rightarrow$ plusieurs dérivations conduisent au même proto-mot

- $E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(T \rightarrow F)}{\Rightarrow} F + E \underset{(F \rightarrow a)}{\Rightarrow} a + E \underset{(E \rightarrow T)}{\Rightarrow} a + T \underset{(T \rightarrow F*T)}{\Rightarrow} a + F * T$
  $\underset{(F \rightarrow a)}{\Rightarrow} a + a * T \underset{(T \rightarrow F)}{\Rightarrow} a + a * F \underset{(F \rightarrow a)}{\Rightarrow} a + a * a$
- **Dérivation droite :** on réécrit le non terminal le plus à droite
  $E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(E \rightarrow T)}{\Rightarrow} T + T \underset{(T \rightarrow F*T)}{\Rightarrow} T + F * T \underset{(T \rightarrow F)}{\Rightarrow} T + F * F$
  $\underset{(F \rightarrow a)}{\Rightarrow} T + F * a \underset{(F \rightarrow a)}{\Rightarrow} T + a * a \underset{(T \rightarrow F)}{\Rightarrow} F + a * a \underset{(F \rightarrow a)}{\Rightarrow} a + a * a$

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$

Plusieurs non-terminaux en partie droite des productions

$\Rightarrow$ plusieurs dérivations conduisent au même proto-mot

- $E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(T \rightarrow F)}{\Rightarrow} F + E \underset{(F \rightarrow a)}{\Rightarrow} a + E \underset{(E \rightarrow T)}{\Rightarrow} a + T \underset{(T \rightarrow F*T)}{\Rightarrow} a + F * T$
  $\underset{(F \rightarrow a)}{\Rightarrow} a + a * T \underset{(T \rightarrow F)}{\Rightarrow} a + a * F \underset{(F \rightarrow a)}{\Rightarrow} a + a * a$
- **Dérivation droite :** on réécrit le non terminal le plus à droite
  $E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(E \rightarrow T)}{\Rightarrow} T + T \underset{(T \rightarrow F*T)}{\Rightarrow} T + F * T \underset{(T \rightarrow F)}{\Rightarrow} T + F * F$
  $\underset{(F \rightarrow a)}{\Rightarrow} T + F * a \underset{(F \rightarrow a)}{\Rightarrow} T + a * a \underset{(T \rightarrow F)}{\Rightarrow} F + a * a \underset{(F \rightarrow a)}{\Rightarrow} a + a * a$
- **Dérivation gauche :** on réécrit le non terminal le plus à droite
  $E \underset{(E \rightarrow T+E)}{\Rightarrow} T + E \underset{(T \rightarrow F)}{\Rightarrow} F + E \underset{(F \rightarrow a)}{\Rightarrow} a + E \underset{(E \rightarrow T)}{\Rightarrow} a + T$
  $\underset{(T \rightarrow F*T)}{\Rightarrow} a + F * T \underset{(F \rightarrow a)}{\Rightarrow} a + a * T \underset{(T \rightarrow F)}{\Rightarrow} a + a * F \underset{(F \rightarrow a)}{\Rightarrow} a + a * a$

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

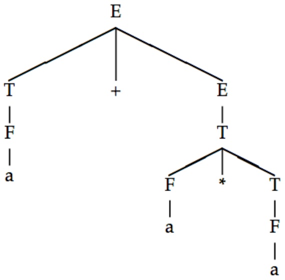
$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E | T, T \rightarrow F * T | F, F \rightarrow a\}$

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$

Arbre de dérivation obtenu avec différentes dérivations complètes :



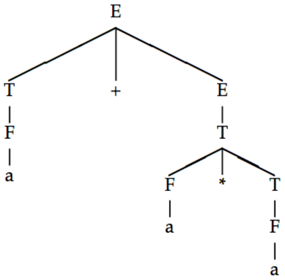
Mais un arbre de dérivation:

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \to T + E|T, T \to F * T|F, F \to a\}$

Arbre de dérivation obtenu avec différentes dérivations complètes :



Mais un arbre de dérivation:

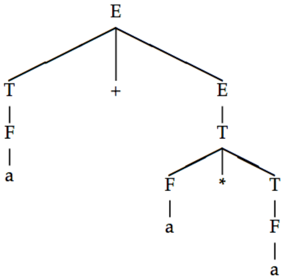
- indique les règles utilisées dans une dérivation

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \rightarrow T + E|T, T \rightarrow F * T|F, F \rightarrow a\}$

Arbre de dérivation obtenu avec différentes dérivations complètes :



Mais un arbre de dérivation:

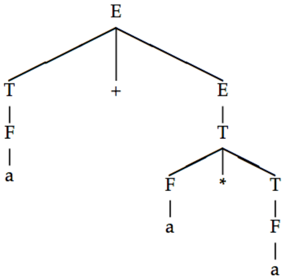
- indique les règles utilisées dans une dérivation
- n'indique pas l'ordre dans lequel elles ont été appliquées

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \to T + E | T, T \to F * T | F, F \to a\}$

Arbre de dérivation obtenu avec différentes dérivations complètes :



Mais un arbre de dérivation:

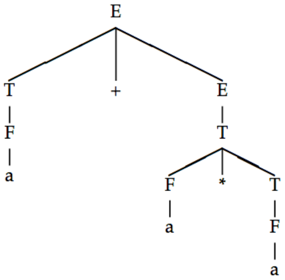
- indique les règles utilisées dans une dérivation
- n'indique pas l'ordre dans lequel elles ont été appliquées
- correspond à une seule dérivation droite

Grammaires formelles : notion d'arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$N = \{E, T, F\}$, $T = \{+, *, a\}$ et $P = \{E \to T + E | T, T \to F * T | F, F \to a\}$

Arbre de dérivation obtenu avec différentes dérivations complètes :



Mais un arbre de dérivation:

- indique les règles utilisées dans une dérivation
- n'indique pas l'ordre dans lequel elles ont été appliquées
- correspond à une seule dérivation droite
- et à une seule dérivation gauche

Grammaires formelles : grammaires ambiguës

Définition : Un grammaire G est dite ambiguë s'il existe au moins un mot $m \in L(G)$ associé à plus d'un arbre de dérivation

Grammaires formelles : grammaires ambiguës

Définition : Un grammaire G est dite ambiguë s'il existe au moins un mot $m \in L(G)$ associé à plus d'un arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$$N = \{E\},\ T = \{+, *, a\} \text{ et } P = \{E \rightarrow E * E | E + E | a\}$$

Grammaires formelles : grammaires ambiguës

Définition : Un grammaire G est dite ambiguë s'il existe au moins un mot $m \in L(G)$ associé à plus d'un arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$$N = \{E\},\ T = \{+, *, a\} \text{ et } P = \{E \to E * E | E + E | a\}$$

Avec cette grammaire, on a $E \underset{(G)}{\overset{*}{\Rightarrow}} a + a * a$

Grammaires formelles : grammaires ambiguës

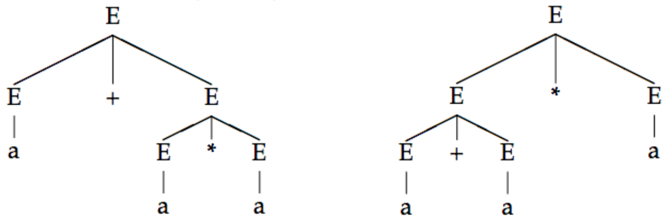
Définition : Un grammaire G est dite ambiguë s'il existe au moins un mot $m \in L(G)$ associé à plus d'un arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$$N = \{E\},\ T = \{+, *, a\} \text{ et } P = \{E \to E * E | E + E | a\}$$

Avec cette grammaire, on a $E \underset{(G)}{\overset{*}{\Rightarrow}} a + a * a$

et deux arbres de dérivations possibles (la grammaire est ambiguë) :



Grammaires formelles : grammaires ambiguës

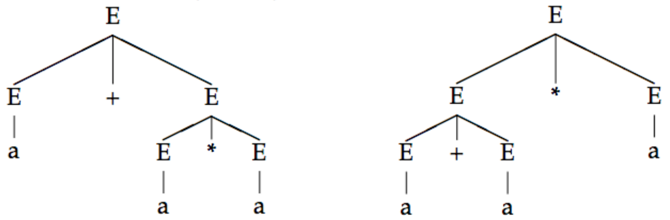
Définition : Un grammaire G est dite ambiguë s'il existe au moins un mot $m \in L(G)$ associé à plus d'un arbre de dérivation

Exemple avec la grammaire $G = (N, T, P, E)$ où :

$$N = \{E\},\ T = \{+, *, a\} \text{ et } P = \{E \rightarrow E * E | E + E | a\}$$

Avec cette grammaire, on a $E \underset{(G)}{\overset{*}{\Rightarrow}} a + a * a$

et deux arbres de dérivations possibles (la grammaire est ambiguë) :



NB. Dans certains cas, l'ambiguïté est liée au langage engendré

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ (on a au moins 1 non-terminal)

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \rightarrow \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ (on a au moins 1 non-terminal)
  - $\beta \in (N \cup T)^*$

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \to \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ (on a au moins 1 non-terminal)
  - $\beta \in (N \cup T)^*$

Et on a vu des grammaires avec plus ou moins de liberté dans les règles :

- $G_1 = (N_1, T_1, P_1, S)$ et $G_2 = (N_2, T_2, P_2, S)$ avec
  - $P_1 = \{S \to A \mid B,\ A \to aA \mid a,\ B \to bB \mid b\}$
    qui engendre $L(G_1) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ un langage régulier

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \to \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ (on a au moins 1 non-terminal)
  - $\beta \in (N \cup T)^*$

Et on a vu des grammaires avec plus ou moins de liberté dans les règles :

- $G_1 = (N_1, T_1, P_1, S)$ et $G_2 = (N_2, T_2, P_2, S)$ avec
  - $P_1 = \{S \to A \mid B,\ A \to aA \mid a,\ B \to bB \mid b\}$
    qui engendre $L(G_1) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ un langage régulier
  - $P_2 = \{S \to aSb \mid \varepsilon\}$
    qui engendre $L(G_1) = \{a^n b^n \mid n \geq 0\}$ un langage qui n'est pas régulier

Grammaires formelles : une classification via les règles

Rappel : une grammaire G est définie par $G = (N, T, P, S)$ où :

- P : ensemble fini de règles de production de la forme $\alpha \to \beta$ où :
  - $\alpha \in (N \cup T)^*.N.(N \cup T)^*$ (on a au moins 1 non-terminal)
  - $\beta \in (N \cup T)^*$

Et on a vu des grammaires avec plus ou moins de liberté dans les règles :

- $G_1 = (N_1, T_1, P_1, S)$ et $G_2 = (N_2, T_2, P_2, S)$ avec
  - $P_1 = \{S \to A \mid B,\ A \to aA \mid a,\ B \to bB \mid b\}$
    qui engendre $L(G_1) = \{a^n \mid n \geq 1\} \cup \{b^n \mid n \geq 1\}$ un langage régulier
  - $P_2 = \{S \to aSb \mid \varepsilon\}$
    qui engendre $L(G_1) = \{a^n b^n \mid n \geq 0\}$ un langage qui n'est pas régulier

Nature des langages engendrables : fonction de restrictions sur les règles

On parle de **typologie des grammaires** via des types de règles

Grammaires formelles : typologie des grammaires

Quatre types de règles :

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 3) Règles régulières dites aussi linéaires :**

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 3) Règles régulières dites aussi linéaires :**
  - **à gauche :** de la forme $A \rightarrow Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 3) Règles régulières dites aussi linéaires :**
  - **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
  - **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 3) Règles régulières dites aussi linéaires :**
  - **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
  - **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

- **(Type 2) Règles hors contexte ou non contextuelles** : de la forme

  $A \to \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

  ("hors contexte" car elle ne considèrent pas de contexte en partie gauche)

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 3) Règles régulières dites aussi linéaires :**
  - **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
  - **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

- **(Type 2) Règles hors contexte ou non contextuelles** : de la forme

  $A \to \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

  ("hors contexte" car elle ne considèrent pas de contexte en partie gauche)

  Par exemple : $S \to aSb \mid \varepsilon$

Grammaires formelles : typologie des grammaires

Quatre types de règles :

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec
  - $A \in N$

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec
  - $A \in N$
  - $g, d, m \in (N \cup T)^*$

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \to g.m.d$ avec
  - $A \in N$
  - $g, d, m \in (N \cup T)^*$

("contextuelles" car la réécriture de A impose le contexte $g.A.d$)

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec
  - $A \in N$
  - $g, d, m \in (N \cup T)^*$

("contextuelles" car la réécriture de A impose le contexte $g.A.d$)

Par exemple : $aaAb \rightarrow aaBabBb$ qui a permis de réécrire A en $BabB$ à condition que A soit "entouré" (cf. contexte) de aa et b.

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec
  - $A \in N$
  - $g, d, m \in (N \cup T)^*$

  ("contextuelles" car la réécriture de A impose le contexte $g.A.d$)

  Par exemple : $aaAb \rightarrow aaBabBb$ qui a permis de réécrire A en $BabB$ à condition que A soit "entouré" (cf. contexte) de aa et b.

- **(Type 0) Sans restriction :** de la forme $\alpha \rightarrow \beta$ avec $|\alpha| \geq 1$

  ($|\alpha| \geq 1$ *a minima* car on a $\alpha \in (N \cup T)^*.N.(N \cup T)^*$)

Grammaires formelles : typologie des grammaires

Quatre types de règles :

- **(Type 1) Règles contextuelles :** de la forme $g.A.d \rightarrow g.m.d$ avec
  - $A \in N$
  - $g, d, m \in (N \cup T)^*$

  ("contextuelles" car la réécriture de A impose le contexte $g.A.d$)

  Par exemple : $aaAb \rightarrow aaBabBb$ qui a permis de réécrire A en $BabB$ à condition que A soit "entouré" (cf. contexte) de aa et b.

- **(Type 0) Sans restriction :** de la forme $\alpha \rightarrow \beta$ avec $|\alpha| \geq 1$

  ($|\alpha| \geq 1$ *a minima* car on a $\alpha \in (N \cup T)^*.N.(N \cup T)^*$)

  Par exemple : $aAbBc \rightarrow abcCcba$... comme quoi, tout est permis !

Grammaires formelles : typologie des grammaires

De quatre types de règles à quatre types de grammaires

Grammaires formelles : typologie des grammaires

De quatre types de règles à quatre types de grammaires

Un grammaire est dite

- **de type 3 ou régulière** si toutes ses règles sont linéaires soit à gauche, soit à droite

Grammaires formelles : typologie des grammaires

De quatre types de règles à quatre types de grammaires

Un grammaire est dite

- **de type 3 ou régulière** si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0

Grammaires formelles : typologie des grammaires

De quatre types de règles à quatre types de grammaires

Un grammaire est dite

- **de type 3 ou régulière** si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0
- **de type 1 ou contextuelles** si elle possède des règles qui sont contextuelles mais aucune de type 0

Grammaires formelles : typologie des grammaires

De quatre types de règles à quatre types de grammaires

Un grammaire est dite

- **de type 3 ou régulière** si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0
- **de type 1 ou contextuelles** si elle possède des règles qui sont contextuelles mais aucune de type 0
- **de type 0** si elle possède des règles qui sont type 0

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Types de grammaires	Classes de langages	Types de machines
Type 3	réguliers, rationnels	Automates finis

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Types de grammaires	Classes de langages	Types de machines
Type 3	réguliers, rationnels	Automates finis
Type 2	algébriques	Automates à pile

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Types de grammaires	Classes de langages	Types de machines
Type 3	réguliers, rationnels	Automates finis
Type 2	algébriques	Automates à pile
Type 1	contextuels	Automates linéairement bornés

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Types de grammaires	Classes de langages	Types de machines
Type 3	réguliers, rationnels	Automates finis
Type 2	algébriques	Automates à pile
Type 1	contextuels	Automates linéairement bornés
Type 0	récursivement énumérables	Machine de Turing

Grammaires formelles : typologie des grammaires

Quatre types de grammaires, de langages et de machines

À un type de grammaire, on peut associer une classe de langages et un type de machine pour leur reconnaissance :

Types de grammaires	Classes de langages	Types de machines
Type 3	réguliers, rationnels	Automates finis
Type 2	algébriques	Automates à pile
Type 1	contextuels	Automates linéairement bornés
Type 0	récursivement énumérables	Machine de Turing

Hiérarchie de Chomsky : inclusions strictes entre puissances des grammaires et types de langages engendrés :

$$\textbf{Type 3} \subsetneq \textbf{Type 2} \subsetneq \textbf{Type 1} \subsetneq \textbf{Type 0}$$

Plan

Plan

Grammaires régulières : équivalence avec les automates

Objectif : démontrer l'équivalence entre grammaires régulières et

- Automates d'états finis
- Expressions régulières

Grammaires régulières : équivalence avec les automates

Objectif : démontrer l'équivalence entre grammaires régulières et

- Automates d'états finis
- Expressions régulières

Cela revient à démontrer le théorème suivant :

Théorème : Un langage L est rationnel (ou régulier ou reconnaissable)

si et seulement si

Il existe un grammaire régulière G telle que $L = L(G)$

Grammaires régulières : équivalence avec les automates

Objectif : démontrer l'équivalence entre grammaires régulières et

- Automates d'états finis
- Expressions régulières

Cela revient à démontrer le théorème suivant :

Théorème : Un langage L est rationnel (ou régulier ou reconnaissable)

si et seulement si

Il existe un grammaire régulière G telle que $L = L(G)$

Pour rappel :

- **(Règles de type 3) Règles régulières dites aussi linéraires :**
  - **à gauche :** de la forme $A \rightarrow Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
  - **à droite :** de la forme $A \rightarrow aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Grammaires régulières : équivalence avec les automates

Preuve de :

Théorème : Un langage L est rationnel (ou régulier ou reconnaissable)

si et seulement si

Il existe un grammaire régulière G telle que $L = L(G)$

Grammaires régulières : équivalence avec les automates

Preuve de :

Théorème : Un langage L est rationnel (ou régulier ou reconnaissable)
si et seulement si
Il existe un grammaire régulière G telle que $L = L(G)$

Constructive et en deux étapes :

- À partir d'une grammaire régulière G telle que $L = L(G)$,
  on construit d'un automate A tel que $L(A) = L$
- À partir d'un automate A tel que $L = L(A)$,
  on construit un grammaire régulière G telle que $L(G) = L$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome
- δ définie par des transitions liées aux règles linéraires à droite :

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome
- δ définie par des transitions liées aux règles linéraires à droite :
    - $\delta(q_i, a) = q_j$ pour une production $A \to aB$ avec $q_i = A$ et $q_j = B$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome
- δ définie par des transitions liées aux règles linéraires à droite :
  - $\delta(q_i, a) = q_j$ pour une production $A \to aB$ avec $q_i = A$ et $q_j = B$
  - $\delta(q_i, a) = X$ pour une production $A \to a$ avec $q_i = A$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome
- δ définie par des transitions liées aux règles linéraires à droite :
  - $\delta(q_i, a) = q_j$ pour une production $A \rightarrow aB$ avec $q_i = A$ et $q_j = B$
  - $\delta(q_i, a) = X$ pour une production $A \rightarrow a$ avec $q_i = A$
- $F = \{X\} \cup \{q_i = A : \text{ il existe une production } A \rightarrow \varepsilon\}$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve (constructive) :

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à droite,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- $Q = N \cup \{X\}$: les non-terminaux vont correspondre aux états et on rajoute un X
- $\Sigma = T$: l'alphabet correspond aux symboles terminaux
- $q_0 = S$: l'état initial correspond à l'axiome
- δ définie par des transitions liées aux règles linéraires à droite :
  - $\delta(q_i, a) = q_j$ pour une production $A \to aB$ avec $q_i = A$ et $q_j = B$
  - $\delta(q_i, a) = X$ pour une production $A \to a$ avec $q_i = A$
- $F = \{X\} \cup \{q_i = A :$ il existe une production $A \to \varepsilon\}$

La preuve rigoureuse se fait en montrant par induction sur la longueur d'un chemin dans l'automate, que $\forall m \in T^* : q \in \widehat{\delta}(S, m) \Leftrightarrow S \overset{*}{\Rightarrow} m.q$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA|a, A \rightarrow bB, B \rightarrow aA|a\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA|a, A \rightarrow bB, B \rightarrow aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA|a, A \rightarrow bB, B \rightarrow aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA|a, A \to bB, B \to aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA|a, A \rightarrow bB, B \rightarrow aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \rightarrow \varepsilon$, alors $F = \{X\} = \{q_X\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA|a, A \to bB, B \to aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \to \varepsilon$, alors $F = \{X\} = \{q_X\}$
- δ définie par des transitions liées aux règles linéraires à droite :

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA|a, A \rightarrow bB, B \rightarrow aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \rightarrow \varepsilon$, alors $F = \{X\} = \{q_X\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \rightarrow aA|a$ donne $\delta(q_S, a) = q_A$ et $\delta(q_S, a) = q_X$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA|a, A \to bB, B \to aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \to \varepsilon$, alors $F = \{X\} = \{q_X\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA|a$ donne $\delta(q_S, a) = q_A$ et $\delta(q_S, a) = q_X$
  - $A \to bB$ donne $\delta(q_A, b) = q_B$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA|a, A \to bB, B \to aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \to \varepsilon$, alors $F = \{X\} = \{q_X\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA|a$ donne $\delta(q_S, a) = q_A$ et $\delta(q_S, a) = q_X$
  - $A \to bB$ donne $\delta(q_A, b) = q_B$
  - $B \to aA|a$ donne $\delta(q_B, a) = q_A$ et $\delta(q_B, a) = q_X$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA|a, A \to bB, B \to aA|a\}$

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme aucune production n'est de la forme $C \to \varepsilon$, alors $F = \{X\} = \{q_X\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA|a$ donne $\delta(q_S, a) = q_A$ et $\delta(q_S, a) = q_X$
  - $A \to bB$ donne $\delta(q_A, b) = q_B$
  - $B \to aA|a$ donne $\delta(q_B, a) = q_A$ et $\delta(q_B, a) = q_X$

On peut facilement vérifier que $L(G) = \{a(ba)^n \mid n \geq 0\} = L(A)$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA, A \rightarrow bB|\varepsilon, B \rightarrow aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA, A \rightarrow bB|\varepsilon, B \rightarrow aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA, A \rightarrow bB|\varepsilon, B \rightarrow aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \rightarrow \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \to \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$
- δ définie par des transitions liées aux règles linéraires à droite :

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \rightarrow aA, A \rightarrow bB|\varepsilon, B \rightarrow aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \rightarrow \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \rightarrow aA$ donne $\delta(q_S, a) = q_A$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \to \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA$ donne $\delta(q_S, a) = q_A$
  - $A \to bB|\varepsilon$ donne $\delta(q_A, b) = q_B$ et c'est tout

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \to \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA$ donne $\delta(q_S, a) = q_A$
  - $A \to bB|\varepsilon$ donne $\delta(q_A, b) = q_B$ et c'est tout
  - $B \to aA|a$ donne $\delta(q_B, a) = q_A$ et $\delta(q_B, a) = q_X$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de $G = (N, T, P, S)$ linéaire à droite :

- $N = \{S, A, B\}$
- $T = \{a, b\}$: les symboles terminaux correspondent à l'alphabet
- $P = \{S \to aA, A \to bB|\varepsilon, B \to aA|a\}$ avec une règle qui produit ε

on construit l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = N \cup \{X\} = \{S, A, B, X\}$ que l'on notera $Q = \{q_S, q_A, q_B, q_X\}$
- $\Sigma = T = \{a, b\}$
- $q_0 = S = q_S$
- comme on a la production $A \to \varepsilon$, alors $F = \{X\} \cup \{A\} = \{q_X, q_A\}$
- δ définie par des transitions liées aux règles linéraires à droite :
  - $S \to aA$ donne $\delta(q_S, a) = q_A$
  - $A \to bB|\varepsilon$ donne $\delta(q_A, b) = q_B$ et c'est tout
  - $B \to aA|a$ donne $\delta(q_B, a) = q_A$ et $\delta(q_B, a) = q_X$

On peut aussi facilement vérifier que $L(G) = \{a(ba)^n \mid n \geq 0\} = L(A)$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière à gauche $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière à gauche $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve : (pour les grammaires linéaires à gauche)

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à gauche,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière à gauche $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve : (pour les grammaires linéaires à gauche)

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à gauche,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

Il suffit pour cela :

- **(1)** d'inverser l'ordre des symboles dans les parties droites des règles afin d'obtenir une grammaire régulière qui engendre le même langage mais avec tous les mots inversés

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière à gauche $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve : (pour les grammaires linéaires à gauche)

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à gauche,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

Il suffit pour cela :

- **(1)** d'inverser l'ordre des symboles dans les parties droites des règles afin d'obtenir une grammaire régulière qui engendre le même langage mais avec tous les mots inversés
- **(2)** de construire l'automate correspondant selon la méthode décrite ci-dessus

Grammaires régulières : équivalence avec les automates

Preuve de :

G grammaire régulière à gauche $\Rightarrow \exists A$ automate tel que $L(A) = L(G)$

Idée de la preuve : (pour les grammaires linéaires à gauche)

étant donnée une grammaire $G = (N, T, P, S)$ linéaire à gauche,

on construit un automate avec ϵ-transition $A = (Q, \Sigma, \delta, q_0, F)$ avec :

Il suffit pour cela :

- **(1)** d'inverser l'ordre des symboles dans les parties droites des règles afin d'obtenir une grammaire régulière qui engendre le même langage mais avec tous les mots inversés
- **(2)** de construire l'automate correspondant selon la méthode décrite ci-dessus
- **(3)** d'inverser le sens des arcs de l'automate et d'échanger les rôles de l'état initial et de l'état final, afin d'obtenir un automate qui reconnaît le langage originel.

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet
- $S = q_0$: l'axiome correspond à l'état initial

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet
- $S = q_0$: l'axiome correspond à l'état initial
- Les règles de production sont construites à partir de δ :

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet
- $S = q_0$: l'axiome correspond à l'état initial
- Les règles de production sont construites à partir de δ :
  - $A \rightarrow aB$ pour une transition $\delta(q_i, a) = q_j$ avec $A = q_i$ et $B = q_j$

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet
- $S = q_0$: l'axiome correspond à l'état initial
- Les règles de production sont construites à partir de δ :
  - $A \rightarrow aB$ pour une transition $\delta(q_i, a) = q_j$ avec $A = q_i$ et $B = q_j$
  - $A \rightarrow a$ pour une transition $\delta(q_i, a) \in F$ avec $A = q_i$

Grammaires régulières : équivalence avec les automates

Preuve de :

A automate $\Rightarrow \exists G$ grammaire régulière telle que $L(G) = L(A)$

Idée de la preuve (constructive) :

étant donné un automate $A = (Q, \Sigma, \delta, q_0, F)$

on construit une grammaire $G = (N, T, P, S)$ linéaire à droite, avec :

- $N = Q$: les états vont correspondre aux non-terminaux
- $T = \Sigma$: les symboles terminaux correspondent à l'alphabet
- $S = q_0$: l'axiome correspond à l'état initial
- Les règles de production sont construites à partir de δ :
  - $A \rightarrow aB$ pour une transition $\delta(q_i, a) = q_j$ avec $A = q_i$ et $B = q_j$
  - $A \rightarrow a$ pour une transition $\delta(q_i, a) \in F$ avec $A = q_i$
  - $A \rightarrow \varepsilon$ pour tout état $q_i \in F$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$
- et les règles de production linéaires à droite à partir des transitions :

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$
- et les règles de production linéaires à droite à partir des transitions :
  - $S \to aQ_1$ car on a $\delta(q_0, a) = q_1$ et $S \to a$ car $Q_1 \in F$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$
- et les règles de production linéaires à droite à partir des transitions :
    - $S \rightarrow aQ_1$ car on a $\delta(q_0, a) = q_1$ et $S \rightarrow a$ car $Q_1 \in F$
    - $Q_1 \rightarrow bQ_2$ car $\delta(q_1, b) = q_2$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$
- et les règles de production linéaires à droite à partir des transitions :
  - $S \to aQ_1$ car on a $\delta(q_0, a) = q_1$ et $S \to a$ car $Q_1 \in F$
  - $Q_1 \to bQ_2$ car $\delta(q_1, b) = q_2$
  - $Q_2 \to aQ_1$ car on a $\delta(q_2, a) = q_1$ et $Q_2 \to a$ car $Q_1 \in F$

Grammaires régulières : équivalence avec les automates

Un exemple : à partir de l'automate $A = (Q, \Sigma, \delta, q_0, F)$:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $F = \{q_1\}$
- δ définie par $\delta(q_0, a) = q_1$, $\delta(q_1, b) = q_2$ et $\delta(q_2, a) = q_1$

on construit la grammaire $G = (N, T, P, S)$ linéaire à droite :

- $N = Q = \{q_0, q_1, q_2\}$ que l'on notera $N = \{S, Q_1, Q_2\}$
- $T = \Sigma = \{a, b\}$
- $q_0 = S$
- et les règles de production linéaires à droite à partir des transitions :
  - $S \to aQ_1$ car on a $\delta(q_0, a) = q_1$ et $S \to a$ car $Q_1 \in F$
  - $Q_1 \to bQ_2$ car $\delta(q_1, b) = q_2$
  - $Q_2 \to aQ_1$ car on a $\delta(q_2, a) = q_1$ et $Q_2 \to a$ car $Q_1 \in F$

On peut aussi facilement vérifier que $L(A) = \{a(ba)^n \mid n \geq 0\} = L(G)$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \ldots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \rightarrow Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \rightarrow aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \rightarrow mB|m|\varepsilon$ ou de la forme $A \rightarrow Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \rightarrow mB$ où $m = s_1 \dots s_k$ peut être remplacée par k règles :

- $A \rightarrow s_1 B_1$
- $B_1 \rightarrow s_2 B_2$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \ldots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$
- $B_1 \to s_2 B_2 \ldots$
- $B_{k-2} \to s_{k-1} B_{k-1}$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \ldots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$
- $B_1 \to s_2 B_2 \ldots$
- $B_{k-2} \to s_{k-1} B_{k-1}$
- $B_{k-1} \to s_k B$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \dots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$
- $B_1 \to s_2 B_2 \dots$
- $B_{k-2} \to s_{k-1} B_{k-1}$
- $B_{k-1} \to s_k B$

car on peut réaliser exactement les mêmes dérivations :

$$A \underset{(A \to s_1 B_1)}{\Rightarrow} s_1 B_1$$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \rightarrow Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \rightarrow aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \rightarrow mB|m|\varepsilon$ ou de la forme $A \rightarrow Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \rightarrow mB$ où $m = s_1 \dots s_k$ peut être remplacée par k règles :

- $A \rightarrow s_1 B_1$
- $B_1 \rightarrow s_2 B_2 \dots$
- $B_{k-2} \rightarrow s_{k-1} B_{k-1}$
- $B_{k-1} \rightarrow s_k B$

car on peut réaliser exactement les mêmes dérivations :

$$A \underset{(A \rightarrow s_1 B_1)}{\Rightarrow} s_1 B_1 \underset{(B_1 \rightarrow s_2 B_2)}{\Rightarrow} s_1 s_2 B_2$$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \ldots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$
- $B_1 \to s_2 B_2 \ldots$
- $B_{k-2} \to s_{k-1} B_{k-1}$
- $B_{k-1} \to s_k B$

car on peut réaliser exactement les mêmes dérivations :

$$A \underset{(A \to s_1 B_1)}{\Rightarrow} s_1 B_1 \underset{(B_1 \to s_2 B_2)}{\Rightarrow} s_1 s_2 B_2 \underset{(B_2 \to s_3 B_3)}{\Rightarrow} \ldots \; s_1 s_2 \ldots s_{k-2} B_{k-2}$$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \to Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \to aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \to mB|m|\varepsilon$ ou de la forme $A \to Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \to mB$ où $m = s_1 \dots s_k$ peut être remplacée par k règles :

- $A \to s_1 B_1$
- $B_1 \to s_2 B_2 \dots$
- $B_{k-2} \to s_{k-1} B_{k-1}$
- $B_{k-1} \to s_k B$

car on peut réaliser exactement les mêmes dérivations :

$$A \underset{(A \to s_1 B_1)}{\Rightarrow} s_1 B_1 \underset{(B_1 \to s_2 B_2)}{\Rightarrow} s_1 s_2 B_2 \underset{(B_2 \to s_3 B_3)}{\Rightarrow} \dots \; s_1 s_2 \dots s_{k-2} B_{k-2}$$

$$\underset{(B_{k-2} \to s_{k-1} B_{k-1})}{\Rightarrow} s_1 s_2 \dots s_{k-1} B_{k-1}$$

Grammaires régulières : extension des règles

On peut travailler sur des règles moins restrictives :

Par définition on a :

- **à gauche :** de la forme $A \rightarrow Ba \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$
- **à droite :** de la forme $A \rightarrow aB \mid a \mid \varepsilon$ avec $A, B \in N$ et $a \in T$

Mais : avec des règles de la forme $A \rightarrow mB|m|\varepsilon$ ou de la forme $A \rightarrow Bm|m|\varepsilon$ avec $A, B \in N$ et $m \in T^*$ (le terminal a remplacé par un mot m) on dispose de grammaires de même puissance.

En effet, une règle $A \rightarrow mB$ où $m = s_1 \dots s_k$ peut être remplacée par k règles :

- $A \rightarrow s_1 B_1$
- $B_1 \rightarrow s_2 B_2 \dots$
- $B_{k-2} \rightarrow s_{k-1} B_{k-1}$
- $B_{k-1} \rightarrow s_k B$

car on peut réaliser exactement les mêmes dérivations :

$$A \underset{(A \rightarrow s_1 B_1)}{\Rightarrow} s_1 B_1 \underset{(B_1 \rightarrow s_2 B_2)}{\Rightarrow} s_1 s_2 B_2 \underset{(B_2 \rightarrow s_3 B_3)}{\Rightarrow} \dots s_1 s_2 \dots s_{k-2} B_{k-2}$$

$$\underset{(B_{k-2} \rightarrow s_{k-1} B_{k-1})}{\Rightarrow} s_1 s_2 \dots s_{k-1} B_{k-1} \underset{(B_{k-1} \rightarrow s_k B)}{\Rightarrow} s_1 s_2 \dots s_{k-1} s_k B = mB$$

Plan

Plan

Grammaires algébriques

Rappel : De quatre types de règles à quatre types de grammaires

- de type 3 ou régulière si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0
- de type 1 ou contextuelles si elle possède des règles qui sont contextuelles mais aucune de type 0
- de type 0 si elle possède des règles qui sont type 0

Grammaires algébriques

Rappel : De quatre types de règles à quatre types de grammaires

- de type 3 ou régulière si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0
- de type 1 ou contextuelles si elle possède des règles qui sont contextuelles mais aucune de type 0
- de type 0 si elle possède des règles qui sont type 0

On s'intéresse aux grammaire **algébriques** :

(Type 2) Règles hors contexte ou non contextuelles : de la forme

$A \rightarrow \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

("hors contexte" car elle ne considèrent pas de contexte en partie gauche)

Grammaires algébriques

Rappel : De quatre types de règles à quatre types de grammaires

- de type 3 ou régulière si toutes ses règles sont linéaires soit à gauche, soit à droite
- **de type 2 ou hors contexte ou non contextuelles ou algébrique** si elle possède des règles qui sont hors contexte mais aucune de type 1 ou 0
- de type 1 ou contextuelles si elle possède des règles qui sont contextuelles mais aucune de type 0
- de type 0 si elle possède des règles qui sont type 0

On s'intéresse aux grammaire **algébriques** :

(Type 2) Règles hors contexte ou non contextuelles : de la forme

$A \rightarrow \beta$ avec $A \in N$ et $\beta \in (N \cup T)^*$

("hors contexte" car elle ne considèrent pas de contexte en partie gauche)

Par exemple : $S \rightarrow aSb \mid \varepsilon$

Grammaires algébriques

Pourquoi les grammaires algébrique ?

Grammaires algébriques

Pourquoi les grammaires algébrique ?

- Plus puissantes que les grammaires régulières

  (pour rappel, les grammaires régulières servent à l'analyse lexicale)

Grammaires algébriques

Pourquoi les grammaires algébrique ?

- Plus puissantes que les grammaires régulières

  (pour rappel, les grammaires régulières servent à l'analyse lexicale)

- Suffisamment puissantes pour la description de la syntaxe des langages de programmation : elles servent à l'**analyse syntaxique**

- Le problème de la reconnaissance de langage, i.e. est-ce qu'un mot appartient à un langage, est décidable (avec l'**algorithme CYK** par exemple)

Grammaires algébriques

Pourquoi les grammaires algébrique ?

- Plus puissantes que les grammaires régulières

  (pour rappel, les grammaires régulières servent à l'analyse lexicale)

- Suffisamment puissantes pour la description de la syntaxe des langages de programmation : elles servent à l'**analyse syntaxique**

- Le problème de la reconnaissance de langage, i.e. est-ce qu'un mot appartient à un langage, est décidable (avec l'**algorithme CYK** par exemple)

- Automatisation de la reconnaissance : aussi par les **automates à pile**

Grammaires algébriques

Pourquoi les grammaires algébrique ?

- Plus puissantes que les grammaires régulières

  (pour rappel, les grammaires régulières servent à l'analyse lexicale)

- Suffisamment puissantes pour la description de la syntaxe des langages de programmation : elles servent à l'**analyse syntaxique**

- Le problème de la reconnaissance de langage, i.e. est-ce qu'un mot appartient à un langage, est décidable (avec l'**algorithme CYK** par exemple)

- Automatisation de la reconnaissance : aussi par les **automates à pile**

et quand on les aura vues... le programme de l'UE sera terminé !

Grammaires algébriques

Exemples de grammaires algébrique :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2^{eme} édition, page 241)

Règle donnée sous **forme normale de Backus** (Backus Normal Form) :

instruction-de-sélection :
  `if` (*expression*) *instruction*
  `if` (*expression*) *instruction* `else` *instruction*
  `switch` (*expression*) *instruction*

Grammaires algébriques

Exemples de grammaires algébrique :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2eme édition, page 241)

Règle donnée sous **forme normale de Backus** (Backus Normal Form) :

instruction-de-sélection :
`if` (*expression*) *instruction*
`if` (*expression*) *instruction* `else` *instruction*
`switch` (*expression*) *instruction*

- N contient A, B et C pour :

  $A = \texttt{instruction-de-sélection}$, $B = \texttt{expression}$ et $C = \texttt{instruction}$

Grammaires algébriques

Exemples de grammaires algébrique :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2^{eme} édition, page 241)

Règle donnée sous **forme normale de Backus** (Backus Normal Form) :

instruction-de-sélection :
`if` (*expression*) *instruction*
`if` (*expression*) *instruction* `else` *instruction*
`switch` (*expression*) *instruction*

- N contient A, B et C pour :

  $A = \mathtt{instruction\text{-}de\text{-}sélection}$, $B = \mathtt{expression}$ et $C = \mathtt{instruction}$

- T contient a, b, c, d et e pour :

  $a = \mathtt{if}$, $b = \mathtt{(}$, $c = \mathtt{)}$ $d = \mathtt{else}$ et $e = \mathtt{switch}$

Grammaires algébriques

Exemples de grammaires algébrique :

(emprunté au "Kernighan et Ritchie" (Le Langage C, 2^{eme} édition, page 241)

Règle donnée sous **forme normale de Backus** (Backus Normal Form) :

instruction-de-sélection :
  `if` (*expression*) *instruction*
  `if` (*expression*) *instruction* `else` *instruction*
  `switch` (*expression*) *instruction*

- N contient A, B et C pour :

  $A = \mathtt{instruction\text{-}de\text{-}sélection}$, $B = \mathtt{expression}$ et $C = \mathtt{instruction}$

- T contient a, b, c, d et e pour :

  $a = \mathtt{if}$, $b = \mathtt{(}$, $c = \mathtt{)}$ $d = \mathtt{else}$ et $e = \mathtt{switch}$

- La règle de production est : $A \to abBcC \mid abBcCdC \mid ebBcC$

  qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$S \underset{(S \to S+S)}{\Rightarrow} S + S$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \dots x_k\}$
- $P = \{S \rightarrow x_1 \mid x_2 \mid \dots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$\color{red}{S} \underset{(S \rightarrow S+S)}{\Rightarrow} \color{red}{S} \color{black}{+ S} \underset{(S \rightarrow S*S)}{\Rightarrow} \color{red}{S} \color{black}{* S + S}$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \rightarrow x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$S \underset{(S \rightarrow S+S)}{\Rightarrow} S + S \underset{(S \rightarrow S*S)}{\Rightarrow} S * S + S \underset{(S \rightarrow x_1)}{\Rightarrow} x_1 * S + S$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$\color{red}{S} \underset{(S \to S+S)}{\Rightarrow} \color{red}{S} \color{black}{+ S} \underset{(S \to S*S)}{\Rightarrow} \color{red}{S} \color{black}{* S + S} \underset{(S \to x_1)}{\Rightarrow} x_1 * \color{red}{S} \color{black}{+ S} \underset{(S \to (S))}{\Rightarrow} x_1 * (\color{red}{S}\color{black}{) + S}$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S \to S+S)}{\Rightarrow} S + S \underset{(S \to S*S)}{\Rightarrow} S * S + S \underset{(S \to x_1)}{\Rightarrow} x_1 * S + S \underset{(S \to (S))}{\Rightarrow} x_1 * (S) + S$$

$$\underset{(S \to S-S)}{\Rightarrow} x_1 * (S - S) + S$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S+S \mid S-S \mid S*S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S \to S+S)}{\Rightarrow} S+S \underset{(S \to S*S)}{\Rightarrow} S*S+S \underset{(S \to x_1)}{\Rightarrow} x_1*S+S \underset{(S \to (S))}{\Rightarrow} x_1*(S)+S$$

$$\underset{(S \to S-S)}{\Rightarrow} x_1*(S-S)+S \underset{(S \to x_2)}{\Rightarrow} x_1*(x_2-S)+S$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \rightarrow x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S \rightarrow S+S)}{\Rightarrow} S + S \underset{(S \rightarrow S*S)}{\Rightarrow} S * S + S \underset{(S \rightarrow x_1)}{\Rightarrow} x_1 * S + S \underset{(S \rightarrow (S))}{\Rightarrow} x_1 * (S) + S$$

$$\underset{(S \rightarrow S-S)}{\Rightarrow} x_1 * (S - S) + S \underset{(S \rightarrow x_2)}{\Rightarrow} x_1 * (x_2 - S) + S \underset{(S \rightarrow x_3)}{\Rightarrow} x_1 * (x_2 - x_3) + S$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S \to S+S)}{\Rightarrow} S + S \underset{(S \to S*S)}{\Rightarrow} S * S + S \underset{(S \to x_1)}{\Rightarrow} x_1 * S + S \underset{(S \to (S))}{\Rightarrow} x_1 * (S) + S$$

$$\underset{(S \to S-S)}{\Rightarrow} x_1 * (S - S) + S \underset{(S \to x_2)}{\Rightarrow} x_1 * (x_2 - S) + S \underset{(S \to x_3)}{\Rightarrow} x_1 * (x_2 - x_3) + S$$

$$\underset{(S \to S/S)}{\Rightarrow} x_1 * (x_2 - x_3) + S/S$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \rightarrow x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S \rightarrow S+S)}{\Rightarrow} S + S \underset{(S \rightarrow S*S)}{\Rightarrow} S * S + S \underset{(S \rightarrow x_1)}{\Rightarrow} x_1 * S + S \underset{(S \rightarrow (S))}{\Rightarrow} x_1 * (S) + S$$

$$\underset{(S \rightarrow S-S)}{\Rightarrow} x_1 * (S - S) + S \underset{(S \rightarrow x_2)}{\Rightarrow} x_1 * (x_2 - S) + S \underset{(S \rightarrow x_3)}{\Rightarrow} x_1 * (x_2 - x_3) + S$$

$$\underset{(S \rightarrow S/S)}{\Rightarrow} x_1 * (x_2 - x_3) + S/S \underset{(S \rightarrow x_2)}{\Rightarrow} x_1 * (x_2 - x_3) + x_2/S$$

Grammaires algébriques

Exemple de grammaire algébrique :

Grammaire engendrant les expressions arithmétiques bien parenthésés (avec k variables)

- $N = \{S\}$
- $T = \{+, -, *, /, (,), x_1, x_2, \ldots x_k\}$
- $P = \{S \to x_1 \mid x_2 \mid \ldots \mid x_k \mid (S) \mid S + S \mid S - S \mid S * S \mid S/S\}$

qui est bien une règle de type 2 car non contextuelle (et pas régulière)

Exemple de dérivation (gauche) :

$$S \underset{(S\to S+S)}{\Rightarrow} S + S \underset{(S\to S*S)}{\Rightarrow} S * S + S \underset{(S\to x_1)}{\Rightarrow} x_1 * S + S \underset{(S\to (S))}{\Rightarrow} x_1 * (S) + S$$

$$\underset{(S\to S-S)}{\Rightarrow} x_1 * (S - S) + S \underset{(S\to x_2)}{\Rightarrow} x_1 * (x_2 - S) + S \underset{(S\to x_3)}{\Rightarrow} x_1 * (x_2 - x_3) + S$$

$$\underset{(S\to S/S)}{\Rightarrow} x_1 * (x_2 - x_3) + S/S \underset{(S\to x_2)}{\Rightarrow} x_1 * (x_2 - x_3) + x_2/S \underset{(S\to x_1)}{\Rightarrow} x_1 * (x_2 - x_3) + x_2/x_1$$

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

- Les langages algébriques sont **clos pour l'union**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est algébrique

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

- Les langages algébriques sont **clos pour l'union**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est algébrique

- Les langages algébriques sont **clos pour la concaténation**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1.L_2$ est algébrique

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

- Les langages algébriques sont **clos pour l'union**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est algébrique

- Les langages algébriques sont **clos pour la concaténation**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1.L_2$ est algébrique

- Les langages algébriques sont **clos pour l'étoile**

  Cela signifie que si L est algébrique, alors L^* est algébrique

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

- Les langages algébriques sont **clos pour l'union**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est algébrique
- Les langages algébriques sont **clos pour la concaténation**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1.L_2$ est algébrique
- Les langages algébriques sont **clos pour l'étoile**

  Cela signifie que si L est algébrique, alors L^* est algébrique

Mais :

- Les langages algébriques ne sont **pas clos pour l'intersection**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cap L_2$ n'est pas nécessairement algébrique

Grammaires algébriques : quelques résultats importants

Clôture des langages algébriques :

- Les langages algébriques sont **clos pour l'union**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est algébrique

- Les langages algébriques sont **clos pour la concaténation**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1.L_2$ est algébrique

- Les langages algébriques sont **clos pour l'étoile**

  Cela signifie que si L est algébrique, alors L^* est algébrique

Mais :

- Les langages algébriques ne sont **pas clos pour l'intersection**

  Cela signifie que si L_1 et L_2 sont algébriques, alors $L_1 \cap L_2$ n'est pas nécessairement algébrique

- Les langages algébriques ne sont pas **clos pour la complémentation**

  Cela signifie que si L défini sur Σ est algébrique, alors $\Sigma^* - L$ n'est pas nécessairement algébrique

Grammaires algébriques : quelques résultats importants

Généralisation du Lemme de l'Étoile aux langages algébriques

Aussi appelé "Lemme d'Ogden"

Grammaires algébriques : quelques résultats importants

Généralisation du Lemme de l'Étoile aux langages algébriques

Aussi appelé "Lemme d'Ogden"

Lemme :

Pour tout langage algébrique L, il existe une constante $\alpha \in \mathbb{N}^*$ telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

Grammaires algébriques : quelques résultats importants

Généralisation du Lemme de l'Étoile aux langages algébriques

Aussi appelé "Lemme d'Ogden"

Lemme :

Pour tout langage algébrique L, il existe une constante $\alpha \in \mathbb{N}^*$ telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v, w, x et y tels que $m = u.v.w.x.y$,

Grammaires algébriques : quelques résultats importants

Généralisation du Lemme de l'Étoile aux langages algébriques

Aussi appelé "Lemme d'Ogden"

Lemme :

Pour tout langage algébrique L, il existe une constante $\alpha \in \mathbb{N}^*$ telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v, w, x et y tels que $m = u.v.w.x.y$, et
- (2) $1 \leq |v.x| \leq \alpha$ et $1 \leq |v.w.x| \leq \alpha$

Grammaires algébriques : quelques résultats importants

Généralisation du Lemme de l'Étoile aux langages algébriques

Aussi appelé "Lemme d'Ogden"

Lemme :

Pour tout langage algébrique L, il existe une constante $\alpha \in \mathbb{N}^*$ telle que pour tout mot $m \in L$ avec $|m| \geq \alpha$:

- (1) il existe des facteurs u, v, w, x et y tels que $m = u.v.w.x.y$, et
- (2) $1 \leq |v.x| \leq \alpha$ et $1 \leq |v.w.x| \leq \alpha$ et
- (3) $\forall i \geq 0$, on a $u.v^i.w.x^i.y \in L$

Ce lemme permet de prouver que $\{a^n b^n c^n \mid n > 0\}$ n'est pas algébrique

Grammaires algébriques : reconnaissance de langage

Le problème :

- Données : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
- Question : est-ce que $m \in L(G)$?

Grammaires algébriques : reconnaissance de langage

Le problème :

- Données : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
- Question : est-ce que $m \in L(G)$?

Un algorithme : CYK pour Cocke-Younger-Kasami

Grammaires algébriques : reconnaissance de langage

Le problème :

- Données : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
- Question : est-ce que $m \in L(G)$?

Un algorithme : CYK pour Cocke-Younger-Kasami

- Hypothèse de travail: la grammaire G en entrée est sous forme normale de Chomsky.

Grammaires algébriques : reconnaissance de langage

Le problème :

- Données : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
- Question : est-ce que $m \in L(G)$?

Un algorithme : CYK pour Cocke-Younger-Kasami

- Hypothèse de travail: la grammaire G en entrée est sous forme normale de Chomsky.
- En sortie : si le mot $m \in L(G)$, alors l'algorithme fournit aussi en résultat un arbre syntaxique

Grammaires algébriques : reconnaissance de langage

Le problème :

- Données : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
- Question : est-ce que $m \in L(G)$?

Un algorithme : CYK pour Cocke-Younger-Kasami

- Hypothèse de travail: la grammaire G en entrée est sous forme normale de Chomsky.
- En sortie : si le mot $m \in L(G)$, alors l'algorithme fournit aussi en résultat un arbre syntaxique
- Complexité : $O(n^3.g)$ avec $n = |m|$ et g est la taille de la grammaire. (donc en $O(n^3)$ si g est une constante)

Nous étudions donc les formes normales pour les grammaires algébriques

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées
- **Deux formes normales** classiques pour les grammaires algébriques

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées
- **Deux formes normales** classiques pour les grammaires algébriques
  - **Forme normale de Greibach** : les règles sont de la forme
    - $A \rightarrow aB_1B_2 \ldots B_k$ ou
    - $A \rightarrow a$ ou encore
    - $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées
- **Deux formes normales** classiques pour les grammaires algébriques
  - **Forme normale de Greibach** : les règles sont de la forme
    - $A \rightarrow aB_1B_2 \dots B_k$ ou
    - $A \rightarrow a$ ou encore
    - $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$

    Théorème : toute grammaire algébrique admet une grammaire algébrique équivalente sous forme normale de Greibach

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées
- **Deux formes normales** classiques pour les grammaires algébriques
  - **Forme normale de Greibach** : les règles sont de la forme
    - $A \rightarrow aB_1B_2 \ldots B_k$ ou
    - $A \rightarrow a$ ou encore
    - $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$

    Théorème : toute grammaire algébrique admet une grammaire algébrique équivalente sous forme normale de Greibach
  - **Forme normale de Chomsky** : les règles sont de la forme
    - $A \rightarrow BC$ avec $B \neq S$ et $C \neq S$, ou
    - $A \rightarrow a$, ou encore
    - $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

Grammaires algébriques : formes normales

Formes normales :

- **Motivations :** faciliter les traitements automatique en "normalisant" les entrées
- **Deux formes normales** classiques pour les grammaires algébriques
  - **Forme normale de Greibach** : les règles sont de la forme
    - $A \to aB_1B_2 \ldots B_k$ ou
    - $A \to a$ ou encore
    - $[S \to \varepsilon$ si $\varepsilon \in L(G)]$

    Théorème : toute grammaire algébrique admet une grammaire algébrique équivalente sous forme normale de Greibach
  - **Forme normale de Chomsky** : les règles sont de la forme
    - $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
    - $A \to a$, ou encore
    - $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

    Théorème : toute grammaire algébrique admet une grammaire algébrique équivalente sous forme normale de Chomsky

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \to aBCdE$ par :

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \to aBCdE$ par :

1. Pour chaque terminal :

   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle A pour a avec $A \to a$

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \to aBCdE$ par :

1. Pour chaque terminal :

   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle
   A pour a avec $A \to a$ et D pour d avec $D \to d$

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \to aBCdE$ par :

1. Pour chaque terminal :

   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle
   A pour a avec $A \to a$ et D pour d avec $D \to d$

   (1.2) Modification de la règle originelle par remplacement des terminaux :
   On obtient alors : $X \to ABCDE$, $A \to a$ et $D \to d$

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \to BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \to a$ ou encore
- $[S \to \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \to aBCdE$ par :

1. Pour chaque terminal :
   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle
   A pour a avec $A \to a$ et D pour d avec $D \to d$

   (1.2) Modification de la règle originelle par remplacement des terminaux :
   On obtient alors : $X \to ABCDE$, $A \to a$ et $D \to d$
2. On limite à 2 non-terminaux les parties droites des règles :

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \rightarrow BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \rightarrow a$ ou encore
- $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \rightarrow aBCdE$ par :

1. Pour chaque terminal :

   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle
   A pour a avec $A \rightarrow a$ et D pour d avec $D \rightarrow d$

   (1.2) Modification de la règle originelle par remplacement des terminaux :
   On obtient alors : $X \rightarrow ABCDE$, $A \rightarrow a$ et $D \rightarrow d$

2. On limite à 2 non-terminaux les parties droites des règles :

   $X \rightarrow ABCDE$ devient :

   $X \rightarrow AX_1$ et $X_1 \rightarrow BX_2$ et $X_2 \rightarrow CX_3$ et $X_3 \rightarrow DE$

Grammaires algébriques : formes normales

Formes normales de Chomsky : exemple de normalisation (pour l'idée)

On doit obtenir des règles sont de la forme

- $A \rightarrow BC$ avec $B \neq S$ et $C \neq S$, ou
- $A \rightarrow a$ ou encore
- $[S \rightarrow \varepsilon$ si $\varepsilon \in L(G)]$ et de plus, on n'a pas de symbole inutile

On normalise une règle du type $X \rightarrow aBCdE$ par :

1. Pour chaque terminal :

   (1.1) Ajout d'un nouveau non-terminal et d'une nouvelle règle
   A pour a avec $A \rightarrow a$ et D pour d avec $D \rightarrow d$

   (1.2) Modification de la règle originelle par remplacement des terminaux :
   On obtient alors : $X \rightarrow ABCDE$, $A \rightarrow a$ et $D \rightarrow d$

2. On limite à 2 non-terminaux les parties droites des règles :

   $X \rightarrow ABCDE$ devient :

   $X \rightarrow AX_1$ et $X_1 \rightarrow BX_2$ et $X_2 \rightarrow CX_3$ et $X_3 \rightarrow DE$

Complexité : en $O(g^2)$ si g est la taille de la grammaire en entrée

Grammaires algébriques : algorithme CYK

Algorithme CYK (pour Cocke-Younger-Kasami)

- Entrées : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
  (G est sous forme normale de Chomsky)
- Sortie : vrai si $m = a_1 a_2 \dots a_n \in L(G)$

Grammaires algébriques : algorithme CYK

Algorithme CYK (pour Cocke-Younger-Kasami)

- Entrées : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
  (G est sous forme normale de Chomsky)
- Sortie : vrai si $m = a_1 a_2 \dots a_n \in L(G)$

Idée : approche par "programmation dynamique"

Grammaires algébriques : algorithme CYK

Algorithme CYK (pour Cocke-Younger-Kasami)

- Entrées : une grammaire $G = (N, T, P, S)$ et un mot $m \in T^*$
  (G est sous forme normale de Chomsky)
- Sortie : vrai si $m = a_1 a_2 \dots a_n \in L(G)$

Idée : approche par "programmation dynamique"

Construction d'une table "triangulaire" de X_{ij}

- axe horizontal indicé par $a_1,\ a_2,\ \dots a_i,\ a_{i+1}\ \dots a_n$
- X_{ij} contient les non-terminaux A tels que $A \overset{*}{\Rightarrow} a_i a_{i+1} \dots a_j$
- la table est construite de proche en proche pour arriver à vérifier si $S \in X_{1n}$ c'est-à-dire si $S \overset{*}{\Rightarrow} a_1 a_1 \dots a_n$
- la première ligne de la table est construite en initilisant, pour $1 \leq i \leq n$, les X_{ii} avec les règles de la forme $A \rightarrow a_i$

Grammaires algébriques : algorithme CYK

- axe horizontal indicé par a_1, a_2, $\ldots a_i$, $a_{i+1} \ldots a_n$
- la première ligne de la table est construite en initilisant, pour $1 \leq i \leq n$, les X_{ii} avec les règles de la forme $A \rightarrow a_i$

La table "triangulaire" de X_{ij} (pour un mot de longueur $n = 5$) :

a_1	a_2	a_3	a_4	a_5
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
X_{12}	X_{23}	X_{34}	X_{45}	
X_{13}	X_{24}	X_{35}		
X_{14}	X_{25}			
X_{15}				

Grammaires algébriques : algorithme CYK

- axe horizontal indicé par $a_1,\ a_2,\ \ldots a_i,\ a_{i+1}\ \ldots a_n$
- la première ligne de la table est construite en initilisant, pour $1 \leq i \leq n$, les X_{ii} avec les règles de la forme $A \rightarrow a_i$

La table "triangulaire" de X_{ij} (pour un mot de longueur $n = 5$) :

a_1	a_2	a_3	a_4	a_5
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
X_{12}	X_{23}	X_{34}	X_{45}	
X_{13}	X_{24}	X_{35}		
X_{14}	X_{25}			
X_{15}				

- Calcul de X_{ij} :
  - X_{ij} contient les non-terminaux A tels que $A \overset{*}{\Rightarrow} a_i a_{i+1} \ldots a_j$

Grammaires algébriques : algorithme CYK

- axe horizontal indicé par $a_1, a_2, \ldots a_i, a_{i+1} \ldots a_n$
- la première ligne de la table est construite en initilisant, pour $1 \leq i \leq n$, les X_{ii} avec les règles de la forme $A \rightarrow a_i$

La table "triangulaire" de X_{ij} (pour un mot de longueur $n = 5$) :

a_1	a_2	a_3	a_4	a_5
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
X_{12}	X_{23}	X_{34}	X_{45}	
X_{13}	X_{24}	X_{35}		
X_{14}	X_{25}			
X_{15}				

- Calcul de X_{ij} :
  - X_{ij} contient les non-terminaux A tels que $A \stackrel{*}{\Rightarrow} a_i a_{i+1} \ldots a_j$
  - donc on a $A \underset{(A \rightarrow BC)}{\Rightarrow} BC \stackrel{*}{\Rightarrow} a_i a_{i+1} \ldots a_j = a_i a_{i+1} \ldots a_k a_{k+1} \ldots a_j$
    avec $B \stackrel{*}{\Rightarrow} a_i a_{i+1} \ldots a_k$ et $C \stackrel{*}{\Rightarrow} a_{k+1} a_{k+2} \ldots a_j$
    et donc $B \in X_{ik}$ et $C \in X_{k+1,j}$, et cela pour tout k tel que $i \leq k < j$

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	

avec X_{11} et X_{22}

$\{B\}\{A, C\}=$

$\{BA, BC\}$

$S \to BC$, $A \to BA$

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
avec X_{11} et X_{22} $\{B\}\{A, C\}=$ $\{BA, BC\}$ $S \to BC$, $A \to BA$	avec X_{22} et X_{33} $\{A, C\}\{A, C\}=$ $\{AA, AC, CA, CC\}$ $B \to CC$			

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
avec X_{11} et X_{22} $\{B\}\{A, C\}=$ $\{BA, BC\}$ $S \to BC$, $A \to BA$	avec X_{22} et X_{33} $\{A, C\}\{A, C\}=$ $\{AA, AC, CA, CC\}$ $B \to CC$	avec X_{33} et X_{44} $\{A, C\}\{B\}=$ $\{AB, CB\}$ $S \to AB$, $C \to AB$		

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
avec X_{11} et X_{22} $\{B\}\{A, C\}=$ $\{BA, BC\}$ $S \to BC$, $A \to BA$	avec X_{22} et X_{33} $\{A, C\}\{A, C\}=$ $\{AA, AC, CA, CC\}$ $B \to CC$	avec X_{33} et X_{44} $\{A, C\}\{B\}=$ $\{AB, CB\}$ $S \to AB$, $C \to AB$	avec X_{44} et X_{55} $\{B\}\{A, C\}=$ $\{BA, BC\}$ $S \to BC$, $A \to BA$	

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \rightarrow AB|BC$, $A \rightarrow BA|a$, $B \rightarrow CC|b$, $C \rightarrow AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
avec X_{11} et X_{23} $\{B\}\{B\}=\{BB\}$ aucune règle	avec X_{22} et X_{34} $\{A, C\}\{S, C\}=$ $\{AS, AC, CS, CC\}$ $B \rightarrow CC$	avec X_{33} et X_{45} $\{A, C\}\{S, A\}=$ $\{AS, AA, CS, CA\}$ aucune règle		
avec X_{12} et X_{33} $\{S, A\}\{A, C\}=$ $\{SA, SC, AA, AC\}$ aucune règle	avec X_{23} et X_{44} $\{B\}\{B\}=\{BB\}$ aucune règle	avec X_{34} et X_{55} $\{S, C\}\{A, C\}=$ $\{SA, SC, CA, CC\}$ $B \rightarrow CC$		

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \rightarrow AB|BC$, $A \rightarrow BA|a$, $B \rightarrow CC|b$, $C \rightarrow AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
avec $\mathbf{X_{11}}$ et $\mathbf{X_{24}}$ $\{B\}\{B\}=\{BB\}$ aucune règle	avec $\mathbf{X_{22}}$ et $\mathbf{X_{35}}$ $\{A, C\}\{B\}=\{AB, CB\}$ $S \rightarrow AB$, $C \rightarrow AB$			
avec $\mathbf{X_{12}}$ et $\mathbf{X_{34}}$ $\{S, A\}\{S, C\}=$ $\{SS, SC, AS, AC\}$ aucune règle	avec $\mathbf{X_{23}}$ et $\mathbf{X_{45}}$ $\{B\}\{S, A\}=\{BS, BA\}$ $A \rightarrow BA$			
avec $\mathbf{X_{13}}$ et $\mathbf{X_{44}}$ $\emptyset\{B\}=\emptyset$ aucune règle	avec $\mathbf{X_{24}}$ et $\mathbf{X_{55}}$ $\{B\}\{A, C\}=\{BA, BC\}$ $S \rightarrow BC$, $A \rightarrow BA$			

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				
avec X_{11} et X_{25} $\{B\}\{S, A, C\}=$ $\{BS, BA, BC\}$ $S \to BC$, $A \to BA$	avec X_{12} et X_{35} $\{S, A\}\{B\}=\{SA, AB\}$ $S \to AB$, $C \to AB$			
avec X_{13} et X_{45} $\emptyset\{S, A\}=\emptyset$ aucune règle	avec X_{14} et X_{55} $\emptyset\{A, C\}=\emptyset$ aucune règle			

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Exemple : pour $m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Comme $S \in X_{15} = \{S, A, C\}$ on est assuré d'avoir $S \xRightarrow{*} baaba$

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :**

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$
  (et en supposant la taille de la grammaire constante)

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$
  (et en supposant la taille de la grammaire constante)
- **En temps :** $O(n^3)$

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$ (et en supposant la taille de la grammaire constante)
- **En temps :** $O(n^3)$
  - toutes les cases de la table sont à traiter : $n^2/2$ cases

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$

  (et en supposant la taille de la grammaire constante)

- **En temps :** $O(n^3)$
  - toutes les cases de la table sont à traiter : $n^2/2$ cases
  - le temps de traitement d'une cases X_{ij} est lié au nombre de "coupures" pour son calcul basé sur X_{ik} et X_{k+1j}

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$

  (et en supposant la taille de la grammaire constante)

- **En temps :** $O(n^3)$
  - toutes les cases de la table sont à traiter : $n^2/2$ cases
  - le temps de traitement d'une cases X_{ij} est lié au nombre de "coupures" pour son calcul basé sur X_{ik} et X_{k+1j}
    et comme $1 \leq i \leq k < j \leq n$, on a au pire $n - 1$ possibilités,

Grammaires algébriques : algorithme CYK

Complexité de l'algorithme (avant implémentation détaillée) :

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
$X_{15} = \{S, A, C\}$				

- **En espace :** taille de la table soit $O(n^2/2) = O(n^2)$ avec $n = |m|$
  (et en supposant la taille de la grammaire constante)
- **En temps :** $O(n^3)$
  - toutes les cases de la table sont à traiter : $n^2/2$ cases
  - le temps de traitement d'une cases X_{ij} est lié au nombre de "coupures" pour son calcul basé sur X_{ik} et X_{k+1j}
    et comme $1 \leq i \leq k < j \leq n$, on a au pire $n - 1$ possibilités,
    on arrive à un facteur mutiplicatif majoré par n

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \rightarrow AB|BC$, $A \rightarrow BA|a$, $B \rightarrow CC|b$, $C \rightarrow AB|a$

- Soit par $S \rightarrow AB$:

$$S \underset{(S \rightarrow AB)}{\Rightarrow} AB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S \to AB)}{\Rightarrow} AB \underset{(A \to BA)}{\Rightarrow} BAB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S \to AB)}{\Rightarrow} AB \underset{(A \to BA)}{\Rightarrow} BAB \underset{(B \to b)}{\Rightarrow} bAB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S \to AB)}{\Rightarrow} AB \underset{(A \to BA)}{\Rightarrow} BAB \underset{(B \to b)}{\Rightarrow} bAB \underset{(A \to a)}{\Rightarrow} baB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$

$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S \to AB)}{\Rightarrow} AB \underset{(A \to BA)}{\Rightarrow} BAB \underset{(B \to b)}{\Rightarrow} bAB \underset{(A \to a)}{\Rightarrow} baB \underset{(B \to CC)}{\Rightarrow} baCC$$

$$\underset{(C \to AB)}{\Rightarrow} baABC \underset{(A \to a)}{\Rightarrow} baaBC \underset{(B \to b)}{\Rightarrow} baabC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \rightarrow AB|BC$, $A \rightarrow BA|a$, $B \rightarrow CC|b$, $C \rightarrow AB|a$

- Soit par $S \rightarrow AB$:

$$S \underset{(S\rightarrow AB)}{\Rightarrow} AB \underset{(A\rightarrow BA)}{\Rightarrow} BAB \underset{(B\rightarrow b)}{\Rightarrow} bAB \underset{(A\rightarrow a)}{\Rightarrow} baB \underset{(B\rightarrow CC)}{\Rightarrow} baCC$$

$$\underset{(C\rightarrow AB)}{\Rightarrow} baABC \underset{(A\rightarrow a)}{\Rightarrow} baaBC \underset{(B\rightarrow b)}{\Rightarrow} baabC \underset{(C\rightarrow a)}{\Rightarrow} baaba$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC,\ A \to BA|a,\ B \to CC|b,\ C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \rightarrow AB|BC$, $A \rightarrow BA|a$, $B \rightarrow CC|b$, $C \rightarrow AB|a$

- Soit par $S \rightarrow AB$:

$$S \underset{(S\rightarrow AB)}{\Rightarrow} AB \underset{(A\rightarrow BA)}{\Rightarrow} BAB \underset{(B\rightarrow b)}{\Rightarrow} bAB \underset{(A\rightarrow a)}{\Rightarrow} baB \underset{(B\rightarrow CC)}{\Rightarrow} baCC$$
$$\underset{(C\rightarrow AB)}{\Rightarrow} baABC \underset{(A\rightarrow a)}{\Rightarrow} baaBC \underset{(B\rightarrow b)}{\Rightarrow} baabC \underset{(C\rightarrow a)}{\Rightarrow} baaba$$

- Soit par $S \rightarrow BC$:

$$S \underset{(S\rightarrow BC)}{\Rightarrow} BC \underset{(B\rightarrow b)}{\Rightarrow} bC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC,\ A \to BA|a,\ B \to CC|b,\ C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC,\ A \to BA|a,\ B \to CC|b,\ C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC,\ A \to BA|a,\ B \to CC|b,\ C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S \to AB)}{\Rightarrow} AB \underset{(A \to BA)}{\Rightarrow} BAB \underset{(B \to b)}{\Rightarrow} bAB \underset{(A \to a)}{\Rightarrow} baB \underset{(B \to CC)}{\Rightarrow} baCC$$
$$\underset{(C \to AB)}{\Rightarrow} baABC \underset{(A \to a)}{\Rightarrow} baaBC \underset{(B \to b)}{\Rightarrow} baabC \underset{(C \to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S \to BC)}{\Rightarrow} BC \underset{(B \to b)}{\Rightarrow} bC \underset{(C \to AB)}{\Rightarrow} bAB \underset{(A \to a)}{\Rightarrow} baB \underset{(B \to CC)}{\Rightarrow} baCC$$
$$\underset{(C \to AB)}{\Rightarrow} baABC \underset{(A \to a)}{\Rightarrow} baaBC \underset{(B \to b)}{\Rightarrow} baabC$$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation ?

Deux dérivations gauches possibles pour

$m = baaba$ avec $S \to AB|BC$, $A \to BA|a$, $B \to CC|b$, $C \to AB|a$

- Soit par $S \to AB$:

$$S \underset{(S\to AB)}{\Rightarrow} AB \underset{(A\to BA)}{\Rightarrow} BAB \underset{(B\to b)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

- Soit par $S \to BC$:

$$S \underset{(S\to BC)}{\Rightarrow} BC \underset{(B\to b)}{\Rightarrow} bC \underset{(C\to AB)}{\Rightarrow} bAB \underset{(A\to a)}{\Rightarrow} baB \underset{(B\to CC)}{\Rightarrow} baCC$$
$$\underset{(C\to AB)}{\Rightarrow} baABC \underset{(A\to a)}{\Rightarrow} baaBC \underset{(B\to b)}{\Rightarrow} baabC \underset{(C\to a)}{\Rightarrow} baaba$$

On illlustre avec $S \to AB$

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

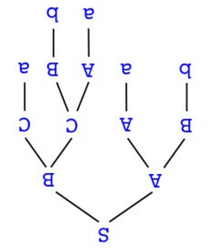
b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				

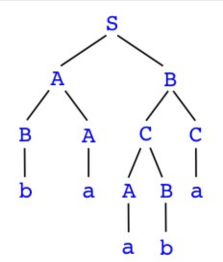


Grammaires algébriques : algorithme CYK

Calcul d'un arbre de dérivation :

- remonter dans la table X_{ij} via les règles de production
- en remontant : s'assurer que la règle fait passer d'un X_{ij} à X_{ik} et X_{k+1j}

b	a	a	b	a
$X_{11} = \{B\}$	$X_{22} = \{A, C\}$	$X_{33} = \{A, C\}$	$X_{44} = \{B\}$	$X_{55} = \{A, C\}$
$B \to b$	$A \to a$ et $C \to a$	$A \to a$ et $C \to a$	$B \to b$	$A \to a$ et $C \to a$
$X_{12} = \{S, A\}$	$X_{23} = \{B\}$	$X_{34} = \{S, C\}$	$X_{45} = \{S, A\}$	
$S \to BC$, $A \to BA$	$B \to CC$	$S \to AB$, $C \to AB$	$S \to BC$, $A \to BA$	
$X_{13} = \emptyset$	$X_{24} = \{B\}$	$X_{35} = \{B\}$		
aucune règle	$B \to CC$	$B \to CC$		
$X_{14} = \emptyset$	$X_{25} = \{S, A, C\}$			
aucune règle	$S \to AB$, $C \to AB$ $S \to BC$, $A \to BA$			
$X_{15} = \{S, A, C\}$				
$S \to BC$, $S \to AB$				



Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

- permettent la reconnaissance des langages algébriques

Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

- permettent la reconnaissance des langages algébriques
- strictement plus puissants que les AFD (donc AFND)
  (strictement plus puissants $\Leftrightarrow$ reconnaissent strictement plus de langages)

Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

- permettent la reconnaissance des langages algébriques
- strictement plus puissants que les AFD (donc AFND)
  (strictement plus puissants $\Leftrightarrow$ reconnaissent strictement plus de langages)
- sont dotés d'une pile : mémoire additionnelle de taille non bornée

Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

- permettent la reconnaissance des langages algébriques
- strictement plus puissants que les AFD (donc AFND)
  (strictement plus puissants $\Leftrightarrow$ reconnaissent strictement plus de langages)
- sont dotés d'une pile : mémoire additionnelle de taille non bornée
- acceptation soit par état final, soit par pile vide, soit les deux

Grammaires algébriques : automates à pile

Automates à pile (*pushdown automaton* en anglais) :

- permettent la reconnaissance des langages algébriques
- strictement plus puissants que les AFD (donc AFND)
  (strictement plus puissants $\Leftrightarrow$ reconnaissent strictement plus de langages)
- sont dotés d'une pile : mémoire additionnelle de taille non bornée
- acceptation soit par état final, soit par pile vide, soit les deux

Plusieurs types d'automates à pile :

- déterministes
- non déterministes : strictement plus puissants que les déterministes
- il existe d'autres variantes qui ne seront pas traitées ici

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre (comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre (comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre (comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre (comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z
  - on empile un mot $p \in \Gamma^*$ (z peut débuter le mot p)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre (comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z
  - on empile un mot $p \in \Gamma^*$ (z peut débuter le mot p)
  - on transite dans un état $q' \subseteq Q$

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z
  - on empile un mot $p \in \Gamma^*$ (z peut débuter le mot p)
  - on transite dans un état $q' \subseteq Q$

  (si $s = \varepsilon$, on parle d'ε-transition)

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z
  - on empile un mot $p \in \Gamma^*$ (z peut débuter le mot p)
  - on transite dans un état $q' \subseteq Q$

  (si $s = \varepsilon$, on parle d'ε-transition)

En sortie : m est accepté si q' est un état final, ou si la pile est vide...
ou les deux

Grammaires algébriques : automates à pile

Un automates à pile non-déterministe : comment ça marche ?

En entrée : un mot $m \in \Sigma^*$ qui va être lu dans l'ordre
(comme pour les AFD)

- $s \in \Sigma \cup \{\varepsilon\}$: symbole courant de m (comme les AFND avec ε-transitions)
- q : un état courant (comme pour les AFD)
- z : un symbole (d'un alphabet de pile noté Γ) situé en sommet de pile
- les transitions : pour une configuration donnée (q, s, z)
  - on dépile z
  - on empile un mot $p \in \Gamma^*$ (z peut débuter le mot p)
  - on transite dans un état $q' \subseteq Q$

  (si $s = \varepsilon$, on parle d'ε-transition)

En sortie : m est accepté si q' est un état final, ou si la pile est vide...
ou les deux

Non-déterminisme : lié aux possibilités de transitions multiples dans une même configuration

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile
- $F \subseteq Q$: ensemble des états terminaux d'acceptation

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile
- $F \subseteq Q$: ensemble des états terminaux d'acceptation
- δ est la fonction de transition : $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$

  où $\mathcal{P}(Q \times \Gamma^*)$ désigne l'ensemble des parties de $Q \times \Gamma^*$

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile
- $F \subseteq Q$: ensemble des états terminaux d'acceptation
- δ est la fonction de transition : $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$

  où $\mathcal{P}(Q \times \Gamma^*)$ désigne l'ensemble des parties de $Q \times \Gamma^*$

Langages reconnus par $\mathcal{A}$ **:** $L_F(\mathcal{A})$ ou bien $L_\emptyset(\mathcal{A})$

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile
- $F \subseteq Q$: ensemble des états terminaux d'acceptation
- δ est la fonction de transition : $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$

  où $\mathcal{P}(Q \times \Gamma^*)$ désigne l'ensemble des parties de $Q \times \Gamma^*$

Langages reconnus par $\mathcal{A}$: $L_F(\mathcal{A})$ ou bien $L_\emptyset(\mathcal{A})$

- Par état final : $L_F(\mathcal{A}) = \{m \in \Sigma^* \mid m \text{ fait transiter de } q_0 \text{ vers } q \in F\}$

Grammaires algébriques : automates à pile

Définition (formelle) : un automates à pile non-déterministe est un sextuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- Q : ensemble fini d'états
- Σ : alphabet (fini) des symboles d'entrée
- Γ : alphabet (fini) des symboles de pile (*a priori* $\Sigma \cap \Gamma = \emptyset$)
- $q_0 \in Q$: état initial
- $z_0 \in \Gamma$: symbole de bas de pile
- $F \subseteq Q$: ensemble des états terminaux d'acceptation
- δ est la fonction de transition : $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$

  où $\mathcal{P}(Q \times \Gamma^*)$ désigne l'ensemble des parties de $Q \times \Gamma^*$

Langages reconnus par $\mathcal{A}$: $L_F(\mathcal{A})$ ou bien $L_\emptyset(\mathcal{A})$

- Par état final : $L_F(\mathcal{A}) = \{m \in \Sigma^* \mid m \text{ fait transiter de } q_0 \text{ vers } q \in F\}$
- Par pile vide : $L_\emptyset(\mathcal{A}) = \{m \in \Sigma^* \mid m \text{ fait transiter jusqu'à la pile vide }\}$

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
  - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
  - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
  - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
    - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
    - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A
    - $\delta(q_0, A, a) = (q_0, AA)$: avec a et un sommet A on empile AA

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
  - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
  - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A
  - $\delta(q_0, A, a) = (q_0, AA)$: avec a et un sommet A on empile AA
  - $\delta(q_0, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
    - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
    - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A
    - $\delta(q_0, A, a) = (q_0, AA)$: avec a et un sommet A on empile AA
    - $\delta(q_0, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien
    - $\delta(q_1, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
  - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
  - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A
  - $\delta(q_0, A, a) = (q_0, AA)$: avec a et un sommet A on empile AA
  - $\delta(q_0, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien
  - $\delta(q_1, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien
  - $\delta(q_1, z_0, \varepsilon) = OUI$

Grammaires algébriques : automates à pile

Exemple bien connu : reconnaissance par pile vide de $\{a^n b^n \mid n \geq 0\}$

Avec l'automate $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ tel que :

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$ avec q_0 pour la lecture des a et q_1 celle des b
- $\Gamma = \{z_0, A\}$: un A sera empilé à chaque lecture d'un a en entrée, et un A sera dépilé à chaque lecture d'un b en entrée
- $q_0 \in Q$: état initial
- F : non significatif car reconnaissance par pile vide
- δ définie par :
  - $\delta(q_0, z_0, \varepsilon) = (q_0, \varepsilon)$: renvoie vers la pile vide et l'acceptation
  - $\delta(q_0, z_0, a) = (q_0, A)$: avec a et une pile vide, on empile A
  - $\delta(q_0, A, a) = (q_0, AA)$: avec a et un sommet A on empile AA
  - $\delta(q_0, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien
  - $\delta(q_1, A, b) = (q_1, \varepsilon)$: avec b et un sommet A on empile rien
  - $\delta(q_1, z_0, \varepsilon) = OUI$ et toutes les autres transitions refusent

Grammaires algébriques : automates à pile

Automates à pile : quelques propriétés

Grammaires algébriques : automates à pile

Automates à pile : quelques propriétés

- **Théorème :** [Pile vide VS État final] Un langage L est reconnaissable par un automate à pile par pile vide si et seulement si le langage L est reconnaissable par un automate à pile par état final
  (**Justification :** quand la pile est vidée, on provoque une transition vers un état final spécifique, et dans l'autre sens, on fait en sorte que la pile soit vidée quand un état final est atteint)

Grammaires algébriques : automates à pile

Automates à pile : quelques propriétés

- **Théorème :** [Pile vide VS État final] Un langage L est reconnaissable par un automate à pile par pile vide si et seulement si le langage L est reconnaissable par un automate à pile par état final

  (**Justification :** quand la pile est vidée, on provoque une transition vers un état final spécifique, et dans l'autre sens, on fait en sorte que la pile soit vidée quand un état final est atteint)

- **Théorème :** [Grammaires hors-contexte] Un langage L est algébrique (engendrable par une grammaire hors-contexte) si et seulement si L est reconnaissable par un automate à pile

Grammaires algébriques : automates à pile

Automates à pile : quelques propriétés

- **Théorème :** [Pile vide VS État final] Un langage L est reconnaissable par un automate à pile par pile vide si et seulement si le langage L est reconnaissable par un automate à pile par état final
  (**Justification :** quand la pile est vidée, on provoque une transition vers un état final spécifique, et dans l'autre sens, on fait en sorte que la pile soit vidée quand un état final est atteint)
- **Théorème :** [Grammaires hors-contexte] Un langage L est algébrique (engendrable par une grammaire hors-contexte) si et seulement si L est reconnaissable par un automate à pile
- **Théorème :** [Déterminisme VS Non-Dét] Il existe des langages algébriques non reconnaissables par des automate à pile déterministes
  (**Conséquence :** les automates à pile non déterministes sont strictement plus puissants que les automates à pile déterministes)