Calculabilité - Décidabilité (ICC)

Cours n°4

Stef Graillat

Sorbonne Université

Résumé du cours précédent

- Loi algèbrique sur les expressions régulières : permet de simplifier des expressions
- Le lemme de pompage : permet de prouver qu'un langage n'est pas régulier
- Propriété de fermeture des langages réguliers : union, intersection, complémentaire, différence, fermeture, concaténation, renversement
- Coûts des conversions entre les représentations : AFD, AFN, ε-AFN, ER
- Propriété de décision des langages réguliers : tester si un langage est vide, si un mot appartient à un langage, si deux langages sont égaux
- Minimisation d'un automate : un automate reconnaissant le même langage mais avec le moins d'états possibles (unicité de l'automate minimal)

Partie II : Automates à piles, grammaires hors-contexte et langages hors-contexte

Grammaires et langages hors-contexte

Définition d'un ensemble de langage contenant strictement les langages réguliers.

- Notation récursive naturelle (les grammaires)
- Rôle central dans les années 60 pour la compilation
- Compilation, Parsers, XML
- Modèle plus puissant, Automates à pile.

Langage défini par les palindromes $L_{\text{pal}} = \{w \in \Sigma^* : w = w^R\}$

Par exemple *otto* $\in L_{\text{pal}}$, *laval* $\in L_{\text{pal}}$

Posons $\Sigma = \{0, 1\}$

Application du lemme de pompage : L_{pal} n'est pas régulier !

Soit n donné par le lemme de pompage, regarder 0^n10^n.

Définition récursive du langage :

- **Base** : ε, 0 et 1 sont des palindromes
- **Induction** : Si w est un palindrome, $0w0$ et $1w1$ le sont aussi
- **Fermeture** : seuls les mots obtenus de cette façon sont des palindromes

Grammaires hors-contexte (suite)

Les grammaires hors-contexte sont un « mécanisme formel » pour des definitions telles que celle de L_{pal}

$$
\begin{array}{lcl}
P & \rightarrow & \varepsilon \\
P & \rightarrow & 0 \\
P & \rightarrow & 1 \\
P & \rightarrow & 0P0 \\
P & \rightarrow & 1P1
\end{array}
$$

0 et 1 sont des symboles terminaux

P est une variable (non terminale)

P est aussi le symbole de départ

Les $P \rightarrow \alpha$ sont des règles de production

Définition formelle

Une grammaire hors-contexte est un quadruplet

$$G = (V, T, P, S)$$

où

- V est un ensemble fini de variables (ou non terminaux). Chacune d'elle représente un langage.
- T est un ensemble fini de symboles formant les mots du langage (alphabet). Ce sont les symboles terminaux.
- P est un ensemble fini de règles de production (ou de réécriture ou de dérivation) de la forme $A \to \alpha$ où A est une variable et $\alpha \in (V \cup T)^*$
- S représente le langage à définir par la grammaire (symbole de départ ou axiome).

Exemples

Exemple 1 : $G_{\text{pal}} = (\{P\}, \{0,1\}, A, P)$ avec
$A = \{P \to \varepsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}$

Il arrive que l'on regroupe les règles de production qui commence par la même variable, $A = \{P \to \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1\}$

Exemple 2 : les expressions régulières sur $\{0,1\}$ peuvent être définies par la grammaire

$$G_{\text{expreg}} = (\{E\}, \{0,1\}, A, E)$$

avec $A = \{E \to 0, E \to 1, E \to E.E, E \to E + E, E \to E^*, E \to (E)\}$

Exemples (suite)

Exemple 3 : La grammaire G est définie à l'aide des variables $\{E, I\}$, définit un langage construit sur les symboles $\{+, *, (,), a, b, 0, 1\}$, la variable E représente le langage à définir ; les règles de production sont

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

Langages avec opérateurs $+$, $*$ et identifiants dans $(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b}+\mathbf{0}+\mathbf{1})^*$

Dérivations

Inférer que certains mots appartiennent au langage d'une variable

- Inférence récursive : On part des symboles (de l'alphabet) et on va jusqu'au mot.
- Dérivation : On étend le symbole de départ via une des productions et on le remplace par l'une de ses productions (jusqu'à dériver notre mot)

1.	E	$\rightarrow$	I
2.	E	$\rightarrow$	$E+E$
3.	E	$\rightarrow$	$E*E$
4.	E	$\rightarrow$	(E)
5.	I	$\rightarrow$	a
6.	I	$\rightarrow$	b
7.	I	$\rightarrow$	Ia
8.	I	$\rightarrow$	Ib
9.	I	$\rightarrow$	$I0$
10.	I	$\rightarrow$	$I1$

	Chaîne	Langage	Règle	Chaîne utilisée
(i)	a	I	5	–
(ii)	b	I	6	–
(iii)	$b0$	I	9	(ii)
(iv)	$b00$	I	9	(iii)
(v)	a	E	1	(i)
(vi)	$b00$	E	1	(iv)
(vii)	$a+b00$	E	2	$(v),(vi)$
$(viii)$	$(a+b00)$	E	4	(vii)
(ix)	$a*(a+b00)$	E	3	$(v),(viii)$

Dérivations (suite)

Soit $G = (V, T, P, S)$ une grammaire hors-contexte, $A \in V$ une variable, $\alpha, \beta \in (V \cup T)^*$ et $A \to \gamma \in P$ une règle de production

Alors on écrit

$$\alpha A \beta \Rightarrow_G \alpha\gamma\beta$$

Extension : $\Rightarrow^*$

- **Base** : soit $\alpha \in (V \cup T)^*$, alors $\alpha \Rightarrow^* \alpha$
- **Induction** : Si $\alpha \Rightarrow^* \beta$ et $\beta \Rightarrow \gamma$ alors $\alpha \Rightarrow^* \gamma$.

Dérivations (suite)

Exemple : Dérivation de $a * (a + b00)$ à partir de E :

$$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E)$$
$$\Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$$

Remarque : à chaque étape, il se peut qu'il y ait plusieurs choix pour les règles à appliquer :
$I * E \Rightarrow a * E \Rightarrow a * (E)$ ou $I * E \Rightarrow I * (E) \Rightarrow a * (E)$

Dérivations à gauche, dérivations à droite

Dérivation à gauche $\Rightarrow_g$: on remplace toujours la variable la plus à gauche

Exemple :

$$\begin{aligned}
E \Rightarrow_g E * E \Rightarrow_g I * E \Rightarrow_g a * E \Rightarrow_g a * (E) \Rightarrow_g a * (E + E) \\
\Rightarrow_g a * (I + E) \Rightarrow_g a * (a + E) \Rightarrow_g a * (a + I) \Rightarrow_g a * (a + I0) \\
\Rightarrow_g a * (a + I00) \Rightarrow_g a * (a + b00)
\end{aligned}$$

Dérivation à droite $\Rightarrow_d$: on remplace toujours la variable la plus à droite

Exemple :

$$\begin{aligned}
E \Rightarrow_d E * E \Rightarrow_d E * (E) \Rightarrow_d E * (E + E) \Rightarrow_d E * (E + I) \Rightarrow_d E * (E + I0) \\
\Rightarrow_d E * (E + I00) \Rightarrow_d E * (E + b00) \Rightarrow_d E * (I + b00) \Rightarrow_d E * (a + b00) \\
\Rightarrow_d I * (a + b00) \Rightarrow_d a * (a + b00)
\end{aligned}$$

Langage d'une grammaire

Définition 1

Soit $G = (V, T, P, S)$ une grammaire hors-contexte. Le langage de G est défini par

$$L(G) = \{w \in T^\star \mid S \Rightarrow^\star w\}$$

On parle alors de langages hors-contexte (ou algébriques)

Autrement dit, $L(G)$ est l'ensemble des mots de T^* que l'on peut dériver à partir du symbole de départ S

Langage d'une grammaire (exemple)

Soit $G_{\text{pal}} = (\{P\}, \{0,1\}, A, P)$ avec
$A = \{P \to \varepsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}$

Théorème 1

$$L(G_{\text{pal}}) = \{w \in \{0,1\}^* : w = w^r\} = L_{\text{pal}}$$

Preuve : "$\supset$" Supposons $w = w^R$. Montrons par induction sur la longueur de w que $w \in L(G_{\text{pal}})$

Base : Si $|w| = 0$ ou $|w| = 1$ alors w est ε, 0 ou 1. Puisqu'on a les règles $P \to \varepsilon,\ P \to 0,\ P \to 1$, on en déduit que $P \Rightarrow^* w$.

Induction : Soit $n \geq 2$ et supposons que tout mot w vérifiant $w = w^R$ et $|w| \leq n$ implique que $w \in L(G_{\text{pal}})$.

Prenons w tel que $w = w^R$ et $|w| = n + 1$ et montrons que $w \in L(G_{\text{pal}})$.

Langage d'une grammaire (exemple)

Comme $w = w^R$, on a $w = 0x0$ ou $w = 1x1$ avec $x = x^R$ et $|x| \leq n$.
Si $w = 0x0$ alors par hypothèse on sait que $P \Rightarrow^* x$. Mais alors

$$P \Rightarrow 0P0 \Rightarrow^* 0x0 = w$$

Par conséquent $w \in L(G_{\text{pal}})$. Le cas de $w = 1x1$ est similaire.

"$\subset$" Supposons que $w \in L(G_{\text{pal}})$. Montrons par induction sur la longueur de dérivation que $w = w^R$.

Base : Si la dérivation $P \Rightarrow^* w$ est de longueur 1 alors w est égale à ε, 0, ou 1 qui sont des palindromes.

Langage d'une grammaire (exemple)

Induction : Soit $n \geq 1$ et supposons que pour tout dérivation $P \Rightarrow^* w$ de longueur n on ait $w = w^R$. Soit alors la dérivation $P \Rightarrow^* w$ de longueur $n+1$. On doit donc avoir

$$\begin{aligned} P &\Rightarrow 0P0 &\Rightarrow^* 0x0 = w \\ P &\Rightarrow 1P1 &\Rightarrow^* 1x1 = w \end{aligned}$$

où la deuxième dérivation est faite en n étapes. Alors x est un palindrome ($x = x^R$) et par conséquent w qui vaut $0x0$ ou $1x1$ est encore un palindrome.
■

Propriété fondamentale des grammaires hors-contexte

Théorème 2

soit $G = (V, T, P, S)$ une grammaire hors-contexte. Soit une dérivation $X_1X_2 \cdots X_n \Rightarrow^* w$ avec $X_i, w \in (V \cup T)^*$. Alors il existe, pour tout $i = 1, \ldots, n$, $w_i \in (V \cup T)^*$ tel que $X_i \Rightarrow^* w_i$ et $w = w_1 w_2 \cdots w_n$.

Preuve : On raisonne par induction sur la longueur de dérivation p.
Base : Si $p = 0$ alors $w = X_1X_2 \cdots X_n$ et donc le théorème est vrai.
Induction : Suppons le théorème vrai pour toutes les dérivations de longueur p. Puisque les parties gauche des règles sont de longueur 1, première règle est appliquée à un X_i. Il existe donc X_i et T_i tels que

$$X_1 \cdots X_{i-1} X_i X_{i+1} \cdots X_n \Rightarrow X_1 \cdots X_{i-1} T_i X_{i+1} \cdots X_n \Rightarrow^* w$$

Par hypothèse, il existe $w_1, \ldots, w_n$ tels que

- $T_i \Rightarrow^* w_i$
- $X_k \Rightarrow w_k$ pour $k = 1, \ldots, i-1$ et $i+1, \ldots, n$
- $w = w_1 \ldots w_n$

Par conséquent $X_k \Rightarrow w_k$ pour $k = 1, \ldots, n$ ∎

Forme syntaxique

Définition 2

Soit $G = (V, T, P, S)$ *une grammaire hors-contexte et* $\alpha \in (V \cup T)^*$. *Si*

$$S \Rightarrow^* \alpha$$

on dit que α *est une forme syntaxique.*

Si $S \Rightarrow_g w$, on parle de forme syntaxique gauche et si $S \Rightarrow_d w$, on parle de forme syntaxique droite

Remarque : $L(G)$ est donc l'ensemble des formes syntaxiques qui sont dans T^*

Arbres de dérivation

Représentation des dérivations par un arbre.

Structure de données pour représenter le source d'un programme.

Facilite la traduction du source vers un exécutable.

Relations entre arbres de dérivations, dérivations et inférences.

Construction des arbres de dérivations

Définition 3

Soit $G = (V, T, P, S)$ une grammaire hors-contexte. Un arbre est un arbre de dérivation pour G si

- *Chaque nœud représente une variable de V*
- *Chaque feuille représente soit un symbole terminal soit ε soit une variable. S'il s'agit de ε il doit être le seul et unique fils de son père.*
- *Si un nœud interne représente A et ses enfants $X_1, \ldots, X_k$ alors $A \to X_1 \cdots X_k$ est une production de la grammaire.*

Productions d'un arbre de dérivation tel que :

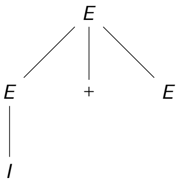
- Chaque feuille représente un symbole terminal ou ε
- La racine est le symbole de départ.

Construction des arbres de dérivations (exemple)

Dans la grammaire

$$\begin{array}{llll} 1. & E & \rightarrow & I \\ 2. & E & \rightarrow & E + E \\ 3. & E & \rightarrow & E * E \\ 4. & E & \rightarrow & (E) \end{array}$$

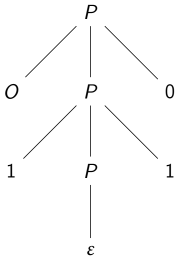
l'exemple suivant est un arbre de dérivations



L'arbre de dérivations montre la dérivation $E \Rightarrow^* I + E$

Construction des arbres de dérivations (exemple)

$$
\begin{array}{llll}
1. & P & \rightarrow & \varepsilon \\
2. & P & \rightarrow & 0 \\
3. & P & \rightarrow & 1 \\
4. & P & \rightarrow & 0P0 \\
5. & P & \rightarrow & 1P1
\end{array}
$$



Cet arbre montre la dérivation $P \Rightarrow^* 0110$

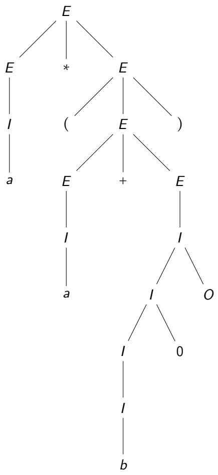
Frontière d'un arbre de dérivation

Définition 4

La frontière d'un arbre de dérivation est le mot obtenu par concaténation des étiquettes des feuilles de gauche à droite

Construction des arbres de dérivations (exemple)

Un autre exemple d'arbre de dérivation



La frontière de l'arbre est $a * (a + b00)$

Soit $G = (V, T, P, S)$ une grammaire hors-contexte et $A \in V$.

On a les équivalences suivantes :

- La procédure d'inférence récursive détermine si un mot w est dans le langage associé à une variable A.
- $A \Rightarrow^\star w$
- $A \Rightarrow_g^\star w$
- $A \Rightarrow_d^\star w$
- il existe un arbre de dérivation de racine A et de frontière w

De l'inférence aux arbres : récurrence sur le nombre de pas dans l'inférence
Des arbres aux dérivations : récurrence sur la hauteur de l'arbre
Des dérivations aux inférences : récurrence sur la longueur de la dérivation

Ambiguité dans les grammaires

Dans la grammaire

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E)
\end{array}
$$

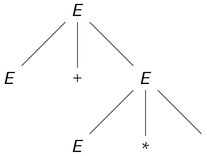
la forme syntaxique $E + E * E$ admet 2 dérivations :

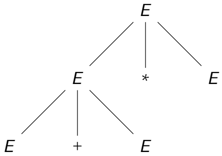
$$E \Rightarrow E + E \Rightarrow E + E * E$$

et

$$E \Rightarrow E * E \Rightarrow E + E * E$$

correspondant au 2 arbres de dérivation





Ambiguité dans les grammaires

Ce n'est pas l'existence de plusieurs dérivations qui est dangereux mais l'existence de plusieurs arbres de dérivation !

Exemple : avec la grammaire

$$\begin{array}{lcl} I & \rightarrow & a \\ I & \rightarrow & b \\ I & \rightarrow & Ia \\ I & \rightarrow & Ib \\ I & \rightarrow & I0 \\ I & \rightarrow & I1 \end{array}$$

la chaîne $a + b$ admet plusieurs dérivations, par exemple,

$$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$

et

$$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$$

Pourtant les arbres de dérivation sont identiques

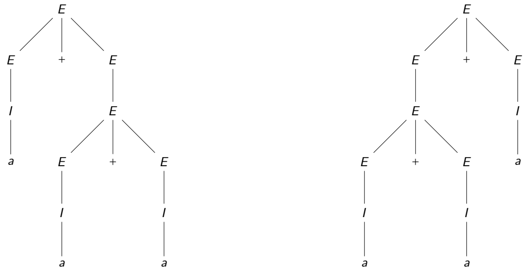
Ambiguité dans les grammaires

Définition 5

Soit $G = (V, T, P, S)$ une grammaire hors-contexte. On dit que G est ambigue s'il existe un mot $w \in L(G)$ admettant deux arbres de dérivation différents

Si tout mot $w \in L(G)$ admet un seul arbre de dérivation, on dit alors que la grammaire est non ambigue.

Exemple : le mot $a + a * a$ a 2 arbres de dérivation



Suppression de l'ambiguité dans les grammaires

Bonne nouvelle : on peut des fois supprimer l'ambiguité « à la main »

Mauvaise nouvelle : il n'existe pas d'algorithme décidant si une grammaire est ambigue !

De plus, il existe des langages hors-contexte qui ne peuvent être définis que par des grammaires ambigues !

Pour la grammaire

$$\begin{array}{rcl} E & \to & I \mid E + E \mid E * E \mid (E) \\ I & \to & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \end{array}$$

l'ambiguité vient de :

- priorité entre $+$ et $*$: il n'y en a pas
- groupement des opérateurs : $E + E + E$ signifie $E + (E + E)$ ou $(E + E) + E$

Suppression de l'ambiguité dans les grammaires (suite)

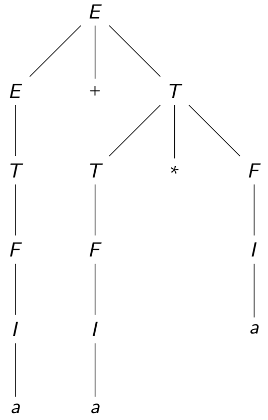
Solution : on introduit de nouvelles variables :

1. les facteurs F : ce sont les identifieurs et les expressions parenthésées
2. les termes T : ceux qu'on ne peut pas « séparer » avec un +
3. les expressions E : on peut « séparer » avec un + ou un $*$

$$
\begin{array}{lcl}
I & \to & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \to & I \mid (E) \\
T & \to & F \mid T * F \\
E & \to & T \mid E + T
\end{array}
$$

Suppression de l'ambiguité dans les grammaires (suite)

Exemple : $a + a * a$ n'a qu'un seul arbre de dérivation

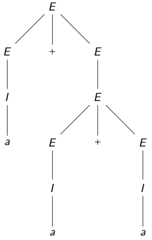


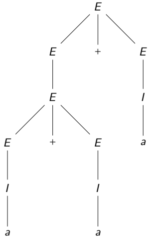
Dérivations à gauche (droite) et ambiguité

Même si une grammaire est non ambigue, les dérivations possibles d'un mot ne sont pas uniques. Mais les dérivations à gauche (droite) le sont !

Théorème 3

Pour toute grammaire, un mot a deux arbres de dérivations distincts ssi il admet deux dérivations à gauche (droite) distinctes.





$$E \Rightarrow_g E + E \Rightarrow_g I + E \Rightarrow_g a + E$$
$$\Rightarrow_g a + E * E \Rightarrow_g a + I * E \Rightarrow_g a + a * E$$
$$\Rightarrow_g a + a * I \Rightarrow_g a + a * a$$

et

$$E \Rightarrow_g E * E \Rightarrow_g E + E * E \Rightarrow_g I + E * E$$
$$\Rightarrow_g a + E * E \Rightarrow_g a + I * E \Rightarrow_g a + a * E$$
$$\Rightarrow_g a + a * I \Rightarrow_g a + a * a$$

Définition 6

Un langage hors-contexte L est inhéremment ambigu si toutes les grammaires hors-contexte pour L sont ambigues.

Exemple : considérons le langage

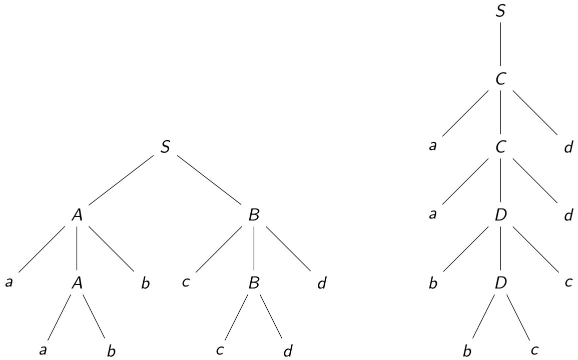
$$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

Une grammaire pour L est

$$\begin{array}{lcl} S & \rightarrow & AB \mid C \\ A & \rightarrow & aAb \mid ab \\ B & \rightarrow & cBd \mid cd \\ C & \rightarrow & aCd \mid aDd \\ D & \rightarrow & bDc \mid bc \end{array}$$

Ambiguité inhérente (suite)

Deux arbres de dérivation pour *aabbccdd*



Ambiguité inhérente (suite)

On voit alors deux dérivations à gauche différentes :

$$S \Rightarrow_g AB \Rightarrow_g aAbB \Rightarrow_g aabbB \Rightarrow_g aabbcBd \Rightarrow_g aabbccdd$$

et

$$S \Rightarrow_g C \Rightarrow_g aCd \Rightarrow_g aaDdd \Rightarrow_g aabDcdd \Rightarrow_g aabbccdd$$

On peut montrer que toute grammaire pour L se comporte de la même manière. Le langage L est inheremment ambigu