Calculabilité - Décidabilité (ICC)

Cours n°6

Stef Graillat

Sorbonne Université

Résumé du cours précédent (1/2)

- Automate à pile : un automate à pile est un automate fini non-déterministe couplé à une pile qui peut stocker des mots de longueur arbitraire. Le pile ne peut être lue et modifiée qu'au somment
- Mouvement d'un automate à pile : un AP choisit ses mouvements en fonction de l'état courant, du symbole lu et du symbole en sommet de pile. On peut modifier le sommet de pile
- Acceptation par un automate à pile : Il y a 2 méthodes d'acceptation pour un AP : l'une en entrant dans un état final, l'autre quand la pile est vide. Ces 2 méthodes sont équivalentes dans le sens ou un langage accepté par l'une est aussi accepté (par un autre AP) par l'autre.
- Configuration : On décrit les états successifs d'un AP par une configuration. Il s'agit d'un triplet : état, entrée restante à lire et état de la pile
- Grammaire et automate à pile : les langages acceptés par un AP soit par état final soit par pile vide sont exactement les langages hors-contexte

Résumé du cours précédent (2/2)

- Automate à pile déterministe : un automate à pile est déterministe s'il n'a jamais le choix de mouvement étant donné un état, un symbole d'entrée et un symbole de pile
- Langage accepté par un APD : tous les langages réguliers sont acceptés (par état final) par un APD. Les langages acceptés par un APD sont hors-contexte et ont une grammaire non-ambigue. Les langages acceptés par un APD contiennent strictement les langages réguliers et sont inclus strictement dans les langages hors-contexte

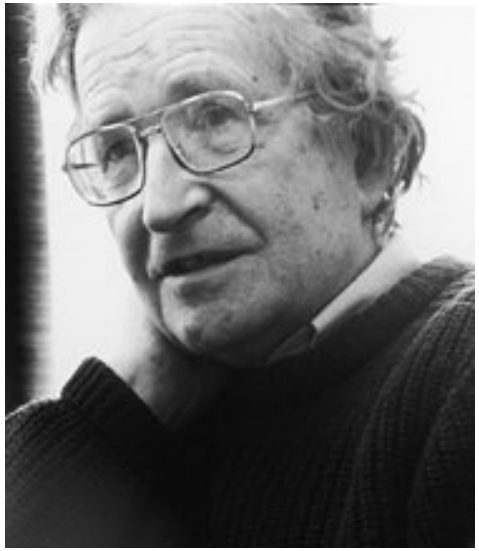
Propriétés des langages hors-contexte

- Simplifier les grammaires hors-contexte, forme spéciale
- Lemme de pompage pour les langages hors-contexte
- Propriétés de fermeture
- Propriétés de décision

Formes normales et grammaires hors-contexte

Forme normale de Chomsky

Objectif : Prouver que toute grammaire peut s'écrire de manière telle que chaque production s'écrit $A \to BC$ ou $A \to a$ où A, B et C sont des variables et a est un symbole terminal.



Noam Chomsky (1928–), professeur émérite de linguistique au MIT

Simplifications préliminaires :

- éliminer les symboles inutiles (n'apparaissant dans aucune dérivation d'un mot obtenu à partir du symbole de départ)
- éliminer les ε-productions ($A \to \varepsilon$)
- éliminer les productions unitaires $A \to B$, A et B étant des variables

Élimination de symboles inutiles

Définition 1

Soit $G = (V, T, P, S)$ une grammaire.

- *Un symbole X est utile s'il existe une dérivation de la forme $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ avec $w \in T^*$. Autrement, il est dit inutile*
- *Un symbole X est productif si $X \Rightarrow^* w$, avec $w \in T^*$*
- *Un symbole X est accessible s'il existe une dérivation $S \Rightarrow^* \alpha X \beta$ avec $\alpha, \beta \in (V \cup T)^*$*

Marche à suivre pour éliminer les symboles inutiles :

- D'abord éliminer les symbole improductifs.
- Puis éliminer les symboles inaccessibles

On obtient une grammaire définissant $L(G)$ sans symbole inutile

Élimination de symboles inutiles (exemple)

Soit la grammaire hors-contexte G définie par

$$S \to AB \mid a, \quad A \to b$$

Les symboles S et A sont productifs, mais B n'est pas productif. Si on élimine B, on doit éliminer la régle $S \to AB$. La grammaire devient

$$S \to a, \quad A \to b$$

Seul S est accessible. En éliminant A et b, la grammaire devient $S \to a$

Attention à l'ordre : si on s'intéresse d'abord à l'accessibilité, on trouve que tous les symboles sont accessibles. On élimine ensuite B qui n'est pas productif. La grammaire contient encore A et b qui sont pourtant inutiles

Calcul des symboles productifs

Soit $G = (V, T, P, S)$ une grammaire. Les symboles productifs $g(G)$ peuvent se calculer de la manière suivante

- **Base :** $T \subset g(G)$
- **Induction :** Supposons qu'il existe une production $A \to \alpha$ et chaque symbole de α appartient à $g(G)$, alors $A \in g(G)$

Exemple : soit G défini par $S \to AB \mid a, A \to b$

D'abord $g(G) = \{a, b\}$. Puisque $S \to a$, S est dans $g(G)$ et comme comme $A \to b$, A est aussi dans $g(G)$

Au final, $g(G) = \{a, b, A, S\}$

Calcul des symboles accessibles

Soit $G = (V, T, P, S)$ une grammaire. Les symboles accessibles $r(G)$ peuvent se calculer de la manière suivante

- **Base :** $S \in r(G)$
- **Induction :** Supposons que $A \in r(G)$ et que l'on ait la règle $A \to \alpha$. Alors tout symbole de α est dans $r(G)$.

Exemple : soit G défini par $S \to AB \mid a, A \to b$

D'abord, $r(G) = \{S\}$

La première règle nous dit que $A, B, a \in r(G)$

La seconde règle nous dit que $b \in r(G)$

Au final, $r(G) = \{S, A, B, a, b\}$

Éliminer les ε-productions

Définition 2

Soit $G = (V, T, P, S)$ une grammaire. Une variable X est annulable si $X \Rightarrow^* \varepsilon$.

Calcul des variables annulables

- **Base :** Si $A \to \varepsilon$ alors A est annulable
- **Induction :** S'il existe une production, $B \to C_1 C_2 \cdots C_k$ où chaque C_i est annulable alors B est annulable

Exemple : soit la grammaire

$$\begin{array}{lcl} S & \to & AB \\ A & \to & aAA \mid \varepsilon \\ B & \to & bBB \mid \varepsilon \end{array}$$

A et B sont annulables. S est aussi annulable car $S \to AB$

Éliminer les ε-productions (suite)

Soit $G = (V, T, P, S)$ une grammaire hors-contexte. Pour éliminer les ε-productions, il faut

1. déterminer les symboles annulables de G
2. Construire une grammaire $G_1 = (V, T, P_1, S)$ dont les règles de production sont construites de la manière suivante :
   Pour chaque règle $A \to X_1 X_2 \cdots X_k$ de P avec $k \geq 1$ supposons que m des k symboles soient annulables. La nouvelle grammaire G_1 aura 2^m versions de cette règle ou chaque X_i annulable est présent ou absent. Si $m = k$ tous les symboles sont annulables et on inclut pas le cas ou tous les X_i sont absents. Si on a une règle de la forme $A \to \varepsilon$, on ne met pas cette règle dans P_1.

Théorème 1

Si G_1 est la grammaire construite précédemment alors $L(G_1) = L(G) \setminus \{\varepsilon\}$

Éliminer les ε-productions (suite)

Exemple : soit la grammaire

$$\begin{array}{lcl} S & \to & AB \\ A & \to & aAA \mid \varepsilon \\ B & \to & bBB \mid \varepsilon \end{array}$$

Les symboles annulables sont A, B, S.

1. $S \to AB$ devient $S \to AB \mid A \mid B$
2. $A \to aAA$ devient $A \to aAA \mid aA \mid aA \mid a$
3. $B \to bBB$ devient $B \to bBB \mid bB \mid bB \mid b$

La grammaire G_1 est donc

$$\begin{array}{lcl} S & \to & AB \mid A \mid B \\ A & \to & aAA \mid aA \mid a \\ B & \to & bBB \mid bB \mid b \end{array}$$

Éliminer les productions unitaires

Définition 3

Une production unitaire est une production de la forme $A \to B$ où A et B sont des variables.

On peut supprimer les productions unitaires.
Exemple : soit la grammaire

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

On peut étendre la règle $E \to T$ et obtenir

$$E \to F, \quad E \to T * F$$

On peut étendre $E \to F$ et obtenir $E \to I \mid (E) \mid T * F$
On peut finalement étendre $E \to I$ et on obtient

$$E \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T * F$$

Éliminer les productions unitaires (suite)

Définition 4

(A, B) est une paire unitaire si $A \Rightarrow^* B$ en utilisant que des productions unitaires.

- **Base :** (A, A) une paire unitaire (provenant de $A \Rightarrow^* A$)
- **Induction :** Si (A, B) est une paire unitaire et $B \to C$ une règle de production alors (A, C) est une paire unitaire.

Exemple : soit la grammaire

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

Base : (E, E), (T, T) et (F, F)

Éliminer les productions unitaires (suite)

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

Base : (E, E), (T, T) et (F, F)
Induction :

- (E, E) et la règle $E \rightarrow T$ donne (E, T)
- (E, T) et la règle $T \rightarrow F$ donne (E, F)
- (E, F) et la règle $F \rightarrow I$ donne (E, I)
- (T, T) et la règle $T \rightarrow F$ donne (T, F)
- (T, F) et la règle $F \rightarrow I$ donne (T, I)
- (F, F) et la règle $F \rightarrow I$ donne (F, I)

Éliminer les productions unitaires (suite)

Soit $G = (V, T, P, S)$ une grammaire. On construit une grammaire $G_1 = (V, T, P_1, S)$ de la façon suivante.

Élimination des productions unitaires :

- Trouver toutes les paires unitaires de G
- Pour chaque paire unitaire (A, B), ajouter dans P_1 les productions $A \to \alpha$ où α est tel que $B \to \alpha$ est une production non unitaire de P.

Théorème 2

La grammaire G_1 ne contient pas de production unitaire et $L(G_1) = L(G)$.

Exemple : soit la grammaire

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

Éliminer les productions unitaires (suite)

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

Paire	Règle de production
(E, E)	$E \rightarrow E + T$
(E, T)	$E \rightarrow T * F$
(E, F)	$E \rightarrow (E)$
(E, I)	$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(T, T)	$T \rightarrow T * F$
(T, F)	$T \rightarrow (E)$
(T, I)	$T \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(F, F)	$F \rightarrow (E)$
(F, I)	$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(I, I)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Simplification d'une grammaire

En résumé, pour simplifier une grammaire, on peut dans l'ordre

1. éliminer les ε-productions
2. éliminer les productions unitaires
3. éliminer les symboles inutiles

Forme normale de Chomsky

Théorème 3

Tout langage hors-contexte ne contenant pas ε peut être défini par une grammaire dont les productions sont de la forme suivante :

- $A \to BC$ *où* A, B *et* C *sont des variables,*
- $A \to a$ *où* A *est une variable et* a *un terminal.*

Algorithme 1 (Mise sous forme normale)

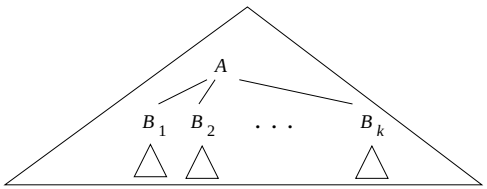
Partir d'une grammaire sans symbole inutile, sans ε-production et sans production unitaire.

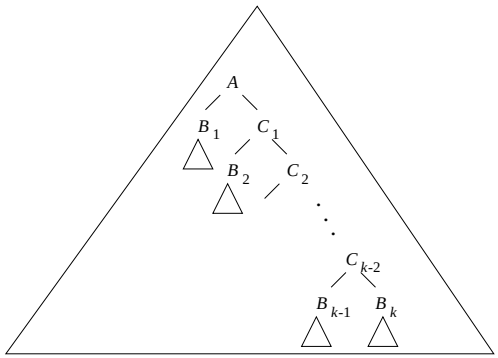
- *(a) Modifier les corps des productions pour que ceux de longueur 2 ou plus ne contiennent que des variables.*
- *(b) Scinder les corps de longueur 3 ou plus en une cascade de productions dont les corps ne contiennent que deux variables.*

Forme normale de Chomsky (suite)

- (a) Modifier les corps des productions pour que ceux de longueur 2 ou plus ne contiennent que des variables.
  Si un terminal a apparait dans le corps d'une règle de longueur 2 ou plus, on crée une nouvelle variable, par exemple A et on remplace a par A dans toutes les règles. On ajoute ensuite la règle $A \to a$.
- (b) Scinder les corps de longueur 3 ou plus en une cascade de productions dont les corps ne contiennent que deux variables.
  Pour chaque règle de la forme $A \to B_1 B_2 \cdots B_k$, $k \geq 3$, on introduit des nouvelles variables $C_1, C_2, \ldots, C_{k-2}$ et on remplace la règle par

$$
\begin{array}{lcl}
A & \to & B_1 C_1 \\
C_1 & \to & B_2 C_2 \\
 & \cdots & \\
C_{k-3} & \to & B_{k-2} C_{k-2} \\
C_{k-2} & \to & B_{k-1} B_k
\end{array}
$$





Forme normale de Chomsky : exemple

Exemple : grammaire déjà simplifiée

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

À l'étape (a), on a besoin des règles $A \to a$, $B \to b$, $Z \to 0$, $O \to 1$, $P \to +$, $M \to *$, $L \to ($, $R \to)$ et en remplaçant dans la grammaire, on obtient

$$\begin{array}{lcl} E & \to & EPF \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ T & \to & TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ F & \to & LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ I & \to & a \mid b \mid IA \mid IB \mid IZ \mid IO \\ A & \to & a, B \to b, Z \to 0, O \to 1, P \to +, M \to *, L \to (, R \to) \end{array}$$

Forme normale de Chomsky : exemple

À l'étape (b), on remplace

$$E \to EPT \text{ par } E \to EC_1, C_1 \to PT$$

$$E \to TMF, T \to TMF \text{ par } E \to TC_2, T \to TC_2, C_2 \to MF$$

$$E \to LER, T \to LER, F \to LER \text{ par } E \to LC_3, T \to LC_3, F \to LC_3, C_3 \to ER$$

Une forme normale de Chomsky de la grammaire est donc

$$\begin{array}{lcl} E & \to & EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ T & \to & TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ F & \to & LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid I0 \\ I & \to & a \mid b \mid IA \mid IB \mid IZ \mid IO \\ A & \to & a, B \to b, Z \to 0, O \to 1, P \to +, M \to *, L \to (, R \to) \\ C_1 & \to & PT, C_2 \to MF, C_3 \to ER \end{array}$$

Le lemme de pompage pour les langages hors-contexte

Un outil pour prouver que certains langages ne sont pas hors-contexte

Pour tout mot assez long d'un langage hors-contexte, on peut trouver deux sous-mots qu'on peut répéter un nombre arbitraire de fois en tandem : le mot obtenu est encore dans le langage hors-contexte

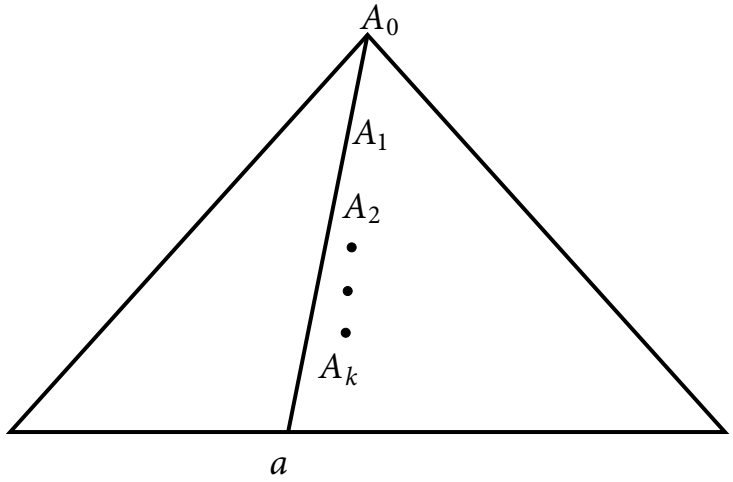
Lemme 1 (Lemme de pompage)

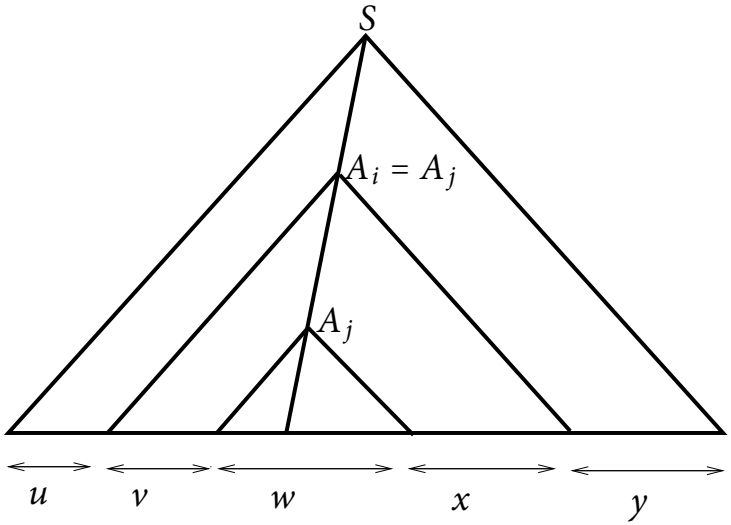
Soit L un langage hors-contexte. Il existe un entier n tel que si $z \in L$ et $|z| \geq n$, alors on peut écrire $z = uvwxy$ avec :

- $|vwx| \leq n$
- $vx \neq \varepsilon$
- *pour tout* $i \geq 0$, $uv^i wx^i y \in L$

Élément de preuve du lemme de pompage

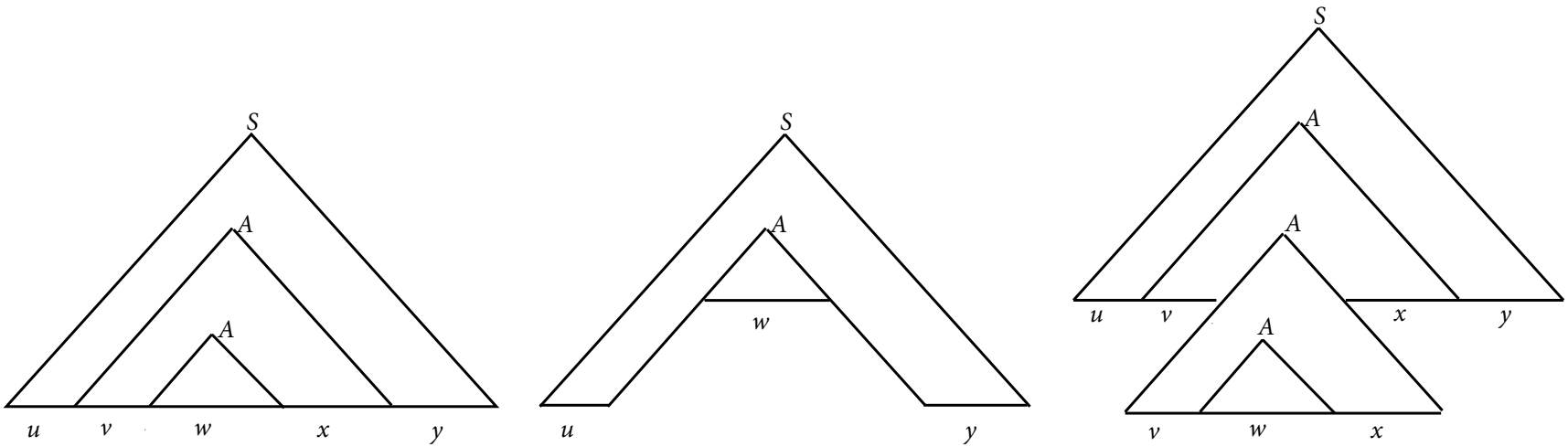
Pour un mot z suffisament long, un arbre de dérivation pour z doit avoir une variable qui est présente au moins 2 fois dans un chemin menant de la racine à une feuille





Supposons donc que $A_i = A_j$ et telle que la forme syntaxique soit $z = uvwxy$ où w est la forme syntaxique géneré par le sous-arbre A_j et vwx est la forme syntaxique géneré par le sous-arbre A_i

Élément de preuve du lemme de pompage



On peut alors remplacer l'arbre A_i par l'arbre A_j ce qui donne la forme syntaxique uwy (correspondant au cas $i = 0$), qui appartient à L.

On peut aussi remplacer l'arbre A_j par l'arbre A_i ce qui donne la forme syntaxique uv^2wx^2y (correspondant au cas $i = 2$), qui appartient à L.

etc.

Exemples d'applications du lemme de pompage

Exemple 1 : $L = \{0^n 1^n 2^n \mid n \in \mathbb{N}\}$

Supposons L hors-contexte. Soit n donné par le lemme de pompage et $z = 0^n 1^n 2^n$. On peut alors découper z en $z = uvwxy$ avec $|vwx| \leq n$ et $vx \neq \varepsilon$.

On remarque que vwx ne peut contenir à la fois des 0 et des 2 puisqu'entre le dernier 0 et le premier 2 il y a $n+1$ positions.

Deux cas sont possibles :

- vwx n'a pas de 2. Alors vx n'a que des 0 et des 1. Par conséquent, uwy qui doit être dans L a n 2 mais moins de n 0 et 1
- vwx n'a pas de 0. Alors vx n'a que des 1 et des 2. Par conséquent ; uwy qui doit être dans L a n 0 mais moins de n 1 et 2

Par conséquent L n'est pas hors-contexte

Exemples d'applications du lemme de pompage (suite)

Exemple 2 : $L = \{0^i 1^j 2^i 3^j \mid (i,j) \in \mathbb{N}^2\}$

Supposons L hors-contexte. Soit n donné par le lemme de pompage et $z = 0^n 1^n 2^n 3^n$. On peut alors découper z en $z = uvwxy$ avec $|vwx| \leq n$ et $vx \neq \varepsilon$.

On remarque que vwx contient un ou deux symboles différents

- si vwx n'a qu'un symbole alors uwy a n occurrences de trois symboles différents et strictement moins de n occurrences du quatrième. Donc uwy ne peut appartenir à L
- si vwx contient deux symboles, par exemple 1 et 2 alors uwy manque soit de 1 soit de 2 soit des deux

Par conséquent L n'est pas hors-contexte

Propriétés de fermeture des langages hors-contexte

Quelques nuances avec les propriétés de fermeture des langages réguliers

Substitutions

Soit Σ un alphabet et à tout symbole $a \in \Sigma$, on associe un langage L_a,

$$s(a) = L_a$$

Pour $w \in \Sigma^*$ avec $w = a_1 \cdots a_n$, on définit $s(w) = s(a_1) \cdots s(a_n)$ et pour $L \subset \Sigma^*$, on définit

$$s(L) = \bigcup_{w \in L} s(w)$$

L'application s est appelée une substitution

Exemple : $\Sigma = \{0, 1\}$ et $s(0) = \{a^n b^n : n \geq 1\}$, $s(1) = \{aa, bb\}$

Pour $w = 01$, on a $s(w) = s(0).s(1) = \{a^n b^n aa : n \geq 1\} \cup \{a^n b^{n+2} : n \geq 1\}$

Pour $L = \{0\}^*$ alors $s(L) = (s(0))^* = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \cdots a^{n_k} b^{n_k} : k \geq 0, n_i \geq 1\}$

Substitutions (suite)

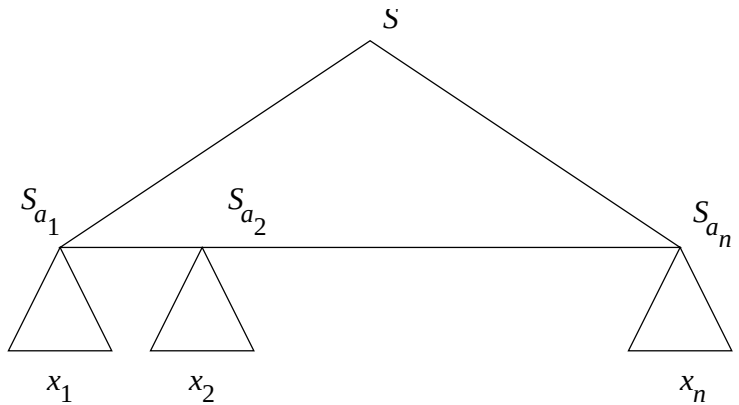
Théorème 4

Si L est un langage hors-contexte construit sur Σ, et s une substitution telle que pour tout $a \in \Sigma$, $s(a)$ est hors-contexte, alors $s(L)$ est hors-contexte.

Idée de la preuve : soit $G = (V, \Sigma, P, S)$ une grammaire pour L et $G_a = (V_a, T_a, P_a, S_a)$ des grammaires pour $s(a)$. Construisons la grammaire $G' = (V', T', P', S)$ avec

- $V' = (\bigcup_{a \in \Sigma} V_a) \cup V$
- $T' = \bigcup_{a \in \Sigma} T_a$
- $P' = \bigcup_{a \in \Sigma} P_a$ plus les productions de P dans lesquelles chaque a est remplacé par le symbole S_a

$$L(G') = s(L)$$



Applications des substitutions

Théorème 5

Soit L_1 et L_2 des langages hors-contexte. Alors les langages suivant sont encore hors-contexte :

1. *union :* $L_1 \cup L_2$
2. *concaténation :* $L_1.L_2$
3. *clôture et clôture positive :* L_1^* *et* L_1^+

Preuve :

1. soit $L = \{1, 2\}$ et $s(1) = L_1$, $s(2) = L_2$ alors $L_1 \cup L_2 = s(L)$
2. soit $L = \{12\}$ et $s(1) = L_1$, $s(2) = L_2$ alors $L_1.L_2 = s(L)$
3. soit $L = \{1\}^*$ et $s(1) = L_1$ alors $L_1^* = s(L)$
   soit $L = \{1\}^+$ et $s(1) = L_1$ alors $L_1^+ = s(L)$

Renversement

Théorème 6

Soit L un langage hors-contexte. Alors L^R est encore un langage hors-contexte.

Preuve : Comme L est hors-contexte, L est généré par une grammaire $G = (V, T, P, S)$. Construisons $G^R = (V, T, P^R, S)$ avec

$$P^R = \{A \to \alpha^R : A \to \alpha \in P\}$$

On peut alors montrer par induction sur la longueur de dérivation que $L(G)^R = L(G^R)$. ■

Intersection avec un langage régulier

Contrairement aux langages réguliers, les langages hors-contexte ne sont pas clos par intersection.
Soit $L_1 = \{0^n 1^n 2^i : n \geq 1, i \geq 1\}$. Le langage L_1 est hors-contexte car reconnu par la grammaire

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & 0A1 \mid 01 \\ B & \rightarrow & 2B \mid 2 \end{array}$$

Soit $L_2 = \{0^i 1^n 2^n : n \geq 1, i \geq 1\}$. Le langage L_2 est hors-contexte car reconnu par la grammaire

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & 0A \mid 0 \\ B & \rightarrow & 1B2 \mid 12 \end{array}$$

Mais $L_1 \cap L_2 = \{0^n 1^n 2^n \mid n \in \mathbb{N}\}$ dont on a vu qu'il n'était pas hors-contexte.

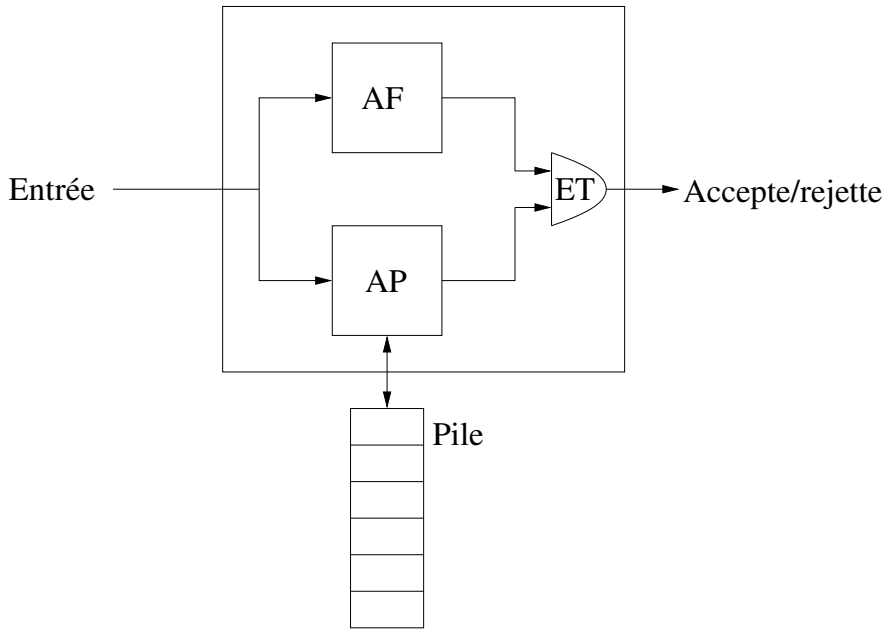
Intersection avec un langage régulier

Théorème 7

Si L est un langage hors-contexte et R un langage régulier, alors $L \cap R$ est un langage hors-contexte.

Preuve : soit L accepté par l'AP $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$ par état final et soit R accepté par l'AF $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$.

On construit un AP pour $L \cap R$ de la façon suivante :



Intersection avec un langage régulier (suite)

Formellement on définit

$$P' = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_A)$$

avec

$$\delta((q,p),a,X) = \{((r, \widehat{\delta}_A(p,a)),\gamma) : (r,\gamma) \in \delta_P(q,a,X)\}$$

On montre alors que

$$(q_P, w, Z_0) \vdash^* (q, \varepsilon, \gamma) \text{ dans } P$$

si et seulement si

$$((q_P, Q_A), w, Z_0) \vdash^* ((q, \widehat{\delta}(p_A, w)), \varepsilon, \gamma) \text{ dans } P'$$

Intersection avec un langage régulier (suite)

Théorème 8

Soit L, L_1 et L_2 des langages hors-contexte, R un langage régulier. Alors

1. *$L \setminus R$ est un langage hors-contexte.*
2. *$\overline{L}$ n'est pas nécessairement hors-contexte.*
3. *$L_1 \setminus L_2$ n'est pas nécessairement hors-contexte.*

Preuve :

1. $\overline{R}$ est régulier et donc $L \setminus R = L \cap \overline{R}$ est hors-contexte
2. si $\overline{L}$ était toujours hors-contexte alors

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

   serait toujours hors-contexte
3. Σ^* est hors-contexte donc $L_1 \setminus L_2$ était toujours hors-contexte alors $\Sigma^* \setminus L = \overline{L}$ serait toujours hors-contexte

Propriétés de décision sur les grammaires hors-contexte

Tests classiques :

- Un langage hors-contexte est-il vide ?
- Un mot donné appartient-il à un langage hors-contexte ?

Problèmes sans solution algorithmique $\rightsquigarrow$ indécidabilité

Complexité de la conversion entre AP et grammaires

Conversions linéaires en la taille de l'entrée :

- Convertir une grammaire en un AP
- Convertir un AP acceptant par état final en un AP acceptant par pile vide
- Convertir un AP acceptant par pile vide en un AP acceptant par état final

Complexité de la conversion AP → Grammaires

Soit n la longueur de l'entrée $\rightsquigarrow n^3$ variables $[pXq]$

Si on y prend garde la conversion peut être exponentielle en n. Le nombre total de productions peut être de l'ordre de n^n !

Si $\delta(q, a, X)$ contient $(r_0, Y_1 \cdots Y_k)$ on introduit des états $p_2, p_3, \ldots, p_{k-1}$ et

- On substitue $(r_0, Y_1 \cdots Y_k)$ dans $\delta(q, a, X)$ par $\{p_{k-1}, Y_{k-1}Y_k\}$
- On introduit les nouvelles transitions :

$$\delta(p_{k-1}, Y_{k-1}) = \{p_{k-2}, Y_{k-2}Y_{k-1}\}$$

pour i allant de 1 à $k-3$ et $\delta(p_2, Y_2) = \{r_0, Y_1Y_2\}$

Pas plus de deux symboles de pile dans chaque transition
On a ajouté au plus n états
Longueur des règles de transition en $\mathcal{O}(n)$
$\mathcal{O}(n)$ transitions engendrant chacune $\mathcal{O}(n^2)$ productions.

$$\mathcal{O}(n^3) \text{ opérations}$$

Conversion en une forme normale de Chomsky

Soit n la longueur de la grammaire donnée en entrée.

- Détection des symboles accessibles et productifs en temps $\mathcal{O}(n)$. élimination des symboles inutiles en temps $\mathcal{O}(n)$ et la grammaire obtenue n'est pas de longueur supérieure à n.
- Construction des paires unitaires et élimination des productions unitaires en temps $\mathcal{O}(n^2)$. La grammaire obtenue est de longueur $\mathcal{O}(n^2)$.
- La substitution des terminaux dans le corps des productions par des variables se fait en temps $\mathcal{O}(n)$ et on obtient une grammaire de longueur $\mathcal{O}(n)$.
- Le scindage des productions se fait en temps $\mathcal{O}(n)$ et il en résulte une grammaire de longueur $\mathcal{O}(n)$.

Conversion en une forme normale de Chomsky (suite)

La procédure éliminant les ε-productions est exponentielle en n

Idée : borner la longueur des productions en les scindant en des productions de longueur 2 au plus.

$\rightarrow$ on élimine ainsi les ε-productions en temps $\mathcal{O}(n)$

Coût total de la mise sous forme normale de Chomsky : $\mathcal{O}(n^2)$

Test du vide

Tester le vide d'un langage hors-contexte : tester si le symbole de départ S est productif

Algorithme naïf : $\mathcal{O}(n^2)$

Objectif : $\mathcal{O}(n)$

Structure de données :

- Pour chaque variable on considère une chaîne de toutes les positions où la variable considérée apparaît.
- Pour chaque production, on a un compteur donnant le nombre de variables y apparaissant dont on ne sait pas si elles sont productives.
- Si un compteur vaut 0, la variable en tête de la production est productive.

Test du vide (suite)

- Création et initialisation du tableau : $\mathcal{O}(n)$ (pas plus de n variables dans une grammaire de taille n).
- Initialisation des liens et des compteurs en $\mathcal{O}(n)$ (pas plus de n productions et leur longueur totale est $\leq n$).
- Quand une production a un compteur valant 0 :
  - Découvrir que le compteur vaut 0 récupérer la variable en-tête et vérifier si elle est déjà connue comme étant productive. Temps $\mathcal{O}(n)$ au total pour toutes les productions.
  - Travail à effectuer dans les productions contenant une variable (proportionnel au nombre de positions de A). Cout total proportionnel à la somme des longueurs des productions : $\mathcal{O}(n)$.

Cout total : $\mathcal{O}(n)$.

Test d'appartenance

Algorithme naïf : exponentiel en la longueur n du mot.

Passer par une forme normale de Chomsky $\rightsquigarrow$ les arbres de dérivation sont binaires à $2n-1$ nœuds.
On peut les lister $\Rightarrow$ Complexité exponentielle en n.

Programmation dynamique $\rightsquigarrow \mathcal{O}(n^3)$

Construction d'une table triangulaire

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
a_1	a_2	a_3	a_4	a_5

Les $X_{i,j}$ sont les ensembles de variables A telles que

$$A \Rightarrow^* a_i a_{i+1} \cdots a_j$$

Découvrir si $S \in X_{1,5}$
On travaille rangée par rangée.

Test d'appartenance (suite)

Algorithme CYK (Cocke-Younger-Kasami)

On suppose la grammaire sous forme normale de Chomsky

Base : Traitement de la première rangée. $X_{i,i}$ contient les variables A telles que $A \to a_i$.
Induction : On veut calculer $X_{i,j}$ qui se trouve dans la rangée $j - i + 1$. La rangée $j - i$ a déjà été calculée. $A \in X_{i,j}$ ssi il existe B, C et k tels que

1. $i \leq k < j$
2. B est dans $X_{i,k}$
3. C est dans $X_{k+1,j}$
4. $A \to BC$ est une production de G.

Complexité : $\mathcal{O}(n^3)$

Test d'appartenance (exemple)

Exemple : Soit G la grammaire

$$\begin{array}{lcl} S & \to & AB \mid BC \\ A & \to & BA \mid a \\ B & \to & CC \mid b \\ C & \to & AB \mid a \end{array}$$

$\{S, A, C\}$				
$-$	$\{S, A, C\}$			
$-$	$\{B\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

Problèmes indécidables

- Une grammaire donnée est-elle ambigue ?
- Une grammaire donnée est-elle intrinsèquement ambigue ?
- L'intersection de deux langages hors-contexte est-elle vide ?
- Deux langages hors-contexte sont-ils égaux ?
- Un langage hors-contexte donné est-il égale à $\Sigma^\star$?