Définitions

Grammaire

$G = (V, \Sigma, R, S)$
où V est l'ensemble de variables, Σ l'alphabet, R les règles de production, et S l'axiome ou variable de départ

Grammaire hors contexte

On limite les règles de la grammaires à $V \times (\Sigma \cup V)^*$

Grammaire linéaire

Règles de la forme $V \times (\Sigma^* V \Sigma^* \cup \Sigma^*)$
L'ensemble des langages engendré par des grammaires linéaires est noté $\mathrm{Lin}(\Sigma^*)$

Grammaire régulière

Grammaire linéaire droite (ie $R \subset V \times (\Sigma^* V \cup \Sigma^*)$)
L'ensemble des langages engendré par des grammaires régulières est noté $\mathrm{Reg}(\Sigma^*)$

Grammaire ambigue

Une grammaire est ambigue si un mot possède deux dérivations différentes

Langage inhéremment ambigu

Un langage est ambigu si toutes les grammaires qui l'engendrent sont ambigues

Langages hors contexte

L'ensemble des langages engendrés par des grammaires hors contexte. Il est noté $\mathrm{HC}(\Sigma^*)$

Remarque: $\mathrm{Reg}(\Sigma^*) \subset \mathrm{Lin}(\Sigma^*) \subset \mathrm{HC}(\Sigma^*)$

Automate à pile

$\mathcal{A} = (Q, \Sigma^*, Z, \Gamma, (q_0, z_0), K)$
où:
- Q est un ensemble fini d'états,
- Σ est l'alphabet d'entrée,
- Z est l'alphabet de pile,
- $(q_0, z_0) \in Q \times Z$ est la configuration initiale,
- $T \subset ((Q \times Z \times (\Sigma \cup \{\varepsilon\})) \times (Q \times Z^*))$ est la relation de transition,
- $K \subset Q \times Z^*$ est une condition d'acceptation.

Les modes d'acceptation sont:
- Par pile vide
- Par état final
- Par pile vide et état final

Remarque: dans tous les cas le mots doit avoir été lu jusqu'à la fin

Automate à pile déterministe

On a $\mathcal{A} = (Q, \Sigma^*, Z, \Gamma, (q_0, z_0), K)$
- Pas deux transition avec la même étiquette depuis un même état
- S'il y a une ε-transition provenant d'un état, aucune autre étiquette

L'ensemble des langages engendré par des Automate à pile déterministe est noté $\mathrm{DHC}(\Sigma^*)$

Théorème: Soit $L \in \mathrm{DHC}(\Sigma^*)$. Alors L peut être engendré par une grammaire non ambigüe

Remarque: on a $\mathrm{Reg}(\Sigma^*) \subsetneq \mathrm{DHC}(\Sigma^*) \subsetneq \mathrm{HC}(\Sigma^*)$

Symbole utile / productif

Un symbole $X \in V$ est utile s'il existe un mot $w \in \Sigma^*$ tel que
$\alpha X \beta \Rightarrow^* w$ avec $\alpha, \beta \in (\Sigma \cup V)^*$

Grammaire réduite

Une grammaire est dite réduite si elle ne comporte au symbole non utile

Symbole accessible

Un symbole $X \in V$ est accessible s'il existe une dérivation $S \Rightarrow^* \alpha X \beta$

Grammaire propre

Une grammaire est propre si elle ne contient pas de production unitaire ni d'ε-production

Méthodes

Stabilité des ensembles:

Langage reconnaissable

Si $L_1, L_2 \subset A^*$ sont reconnaissables, alors:
- $\overline{L_1}$ est reconnaissable
- $L_1 \cup L_2, L_1 \cap L_2$ sont reconnaissables
- $L_1 \cdot L_2$ est reconnaissable
- L_1^+, L_2^* sont reconnaissables

Langage HC

Si $L_1, L_2 \subset A^*$ sont HC et $K \subset A^*$ est reconnaissable, alors:
- $L_1 \cup L_2$ est HC
- $L_1 \cap K$ est reconnaissable
- L_1^* est HC
- si L_1 est deterministe, $\overline{L_1}$ est reconnaissable

Lemme fondamental

Soit $G = (V, \Sigma, R, S)$ une grammaire hors-contexte, soit $k \in \mathbb{N}$ et $\alpha_1, \alpha_2, \beta \in (\Sigma \cup V)^*$. Si $\alpha_1 \alpha_2 \Rightarrow^k \beta$, alors il existe $\beta_1, \beta_2 \in (\Sigma \cup V)^*$ et $k_1, k_2 \in \mathbb{N}$ tel que $\beta = \beta_1 \beta_2, \alpha_1 \Rightarrow_1^k \beta_1, \alpha_1 \Rightarrow_1^k \beta_2 \text{ et } k_1 + k_2 = k$

Transformation automate → grammaire

On note toutes les transitions sous forme $A \to [q_i Z q_j]$ ($[q_i Z q_j]$ sera par la suite remplacé par des variables)

Identifier règles qui pop et qui push
on met d'abord les pop, qui sont plus simples:
[etat_debut symbole_pile etat_arrive] → lettre_a_pop

Puis les push:
[etat_debut symbole_pile etat_arrive] →
lettre_a_empiler [circuit à prendre pour dépiler règles de droite]

Transformation grammaire → automate

$(q_0, Z_0) \xrightarrow{\varepsilon} (q_l, SZ_0)$

$(q_0, X) \xrightarrow{\varepsilon} (q_l, w), \forall X \to w \in R$

$(q_0, a) \xrightarrow{a} (q_l, \varepsilon), \forall a \in \Sigma$

$(q_l, Z_0) \xrightarrow{\varepsilon} (q_f, \varepsilon)$

Déterminer si un langage est reconnaissable

Faire hypothèse:
- Reconnaissable? → contruction d'automate / regex
- Non reconnaissable? → raisonnement par l'absurde avec: règles de cloture et lemme de l'étoile

Lemme de l'étoile

Soit L un langage reconnaissable. Soit $w = xyz \in L$. Alors il existe $N \in \mathbb{N}$, $|w| \geq N$ et:
- $|y| > 0$
- $|xy| \leq N$
- $xy^*z \in L$

Démo: $L = \{a^p \mid p \text{ premier}\}$ **non reconnaissable**
Supposons qu'il l'est.
On dispose de N tq $\forall \omega \in L$ si $|\omega| \geq N$ alors on respecte les hypo du lemme de l'étoile
Soit P premier tq $P \geq N, \omega = a^P$ On décompose P en trois nombres: $P = p + q + r$. Alors $\omega = a^p a^q a^r = xyz$

On pose $m = p + 2q + r + 2$ D'après le lemme de l'étoile, $xy^m z = a^{p+r+q(p+2q+r+2)} = a^{(q+1)(p+2q+r)} \in L$ or $(q+1)(p+2q+r)$ n'est pas premier, contradiction

Déterminer si un langage est hors contexte

Faire hypothèse:
- hors contexte? → contruction d'automate / regex
- Non hors contexte? → raisonnement par l'absurde avec le lemme d'itération algébrique

Lemme d'itération algébrique

Soit $L \in \mathrm{HC}(\Sigma^*)$. Alors il existe $N \in \mathbb{N}$ tel que: $\forall \omega \in L$ si $|\omega| > N$ alors $w = uvxyz$, avec:
- $|vy| > 0$
- $|vxy| \leq N$
- $uv^i xy^i z \in L, \forall i \in \mathbb{N}$

Démo: $L = \{a^n b^m c^n d^m\}$ **n'est pas HC**
Supposons L HC. On considère $\omega = a^N b^N c^N d^N$ $|\omega| \geq N$

On définit 4 blocs consécutifs de longueur N: $A = a^n$, …

Comme $|vxy| \leq N$, vxy est:
- soit dans un seul bloc
- soit en chevauche

1. Si $vxy \in A$ D'après le lemme de l'étoile, $uv^0xy^0z \in L$ or le nombre de c est constant, alors que le nombre de a diminue. On a $n = |w|_a < |w|_c = n$. Absurde
2. Si vxy chevauche AB Donc v et y contiennent soit des a soit des b S'il contient au moins un a: cas 1 S'il contient au moins un b: cas 1 en miroir

Réduction de grammaires

$G' = \mathcal{R}(\mathcal{P}(G))$

Elimination des symboles non-productifs

On calcule $\mathbb{P}(G)$ l'ensemble des symboles accessibles. Construction récursive de $\mathbb{P}(G)$:

$$
\begin{aligned}
&P_0(G) = \{X \mid X \to w \in R \ \text{ avec } \ w \in \Sigma^*\} \\
&P_{n+1}(G) = P_n(G) \cup \{X \mid W \to u \in R \ \text{ et } \ u \in (\Sigma \cup P_n(G))^*\} \\
&\mathbb{P}(G) = \bigcup P_n(G)
\end{aligned}
$$

On calcule alors $\mathcal{P}(G)$ en retirant à G toutes les productions ne contenant ni à gauche ni à droite un élément de $\mathbb{P}(G)$

Elimination des symboles non-accessibles

On calcule $\mathbb{R}(G)$ l'ensemble des symboles accessibles. Construction récursive de $\mathbb{R}(G)$:

$$
\begin{aligned}
&R_0(G) = \{S\} \\
&R_{n+1}(G) = R_n(G) \cup \{X \mid Y \to u \in R \ \text{ et } \ Y \in R_n(G) \ \text{ et } \ X \in u\} \\
&\mathbb{R}(G) = \bigcup R_n(G)
\end{aligned}
$$

On calcule alors $\mathcal{R}(G)$ en retirant à G toutes les productions ne contenant ni à gauche ni à droite un élément de $\mathbb{R}(G)$

Rendre une grammaire propre

~~Il faut la laver~~

Trouver toutes ε-production, les supprimer, créer des nouvelles règles exprimant toutes les possibilité de l'absence de ce symbole.

Par la suite, supprimer les mono-production avec en remplaçant leurs expressions par l'expension de la variable complète