Chapitre 4

(Chapitre 2 de Sipser)

Les langages hors-contexte

Motivation

- description de langages plus intéressants que les langages réguliers
- préparation à l'étude de la compilation des langages de programmation
- ramifications en théorie de la complexité (classe LOGCFL, dépasse l'objet de ce cours)

Grammaire hors-contexte

Exemple 4.1.

$$
\begin{array}{rcl}
S & \to & \varepsilon \\
 & & \mid \langle\text{INCRÉMENTATION}\rangle S \\
 & & \mid \langle\text{AFFECTATION}\rangle S \\
 & & \mid \langle\text{RÉPÉTER}\rangle S \\
\langle\text{INCRÉMENTATION}\rangle & \to & \texttt{inc}(V) \\
\langle\text{AFFECTATION}\rangle & \to & V \leftarrow V \\
\langle\text{RÉPÉTER}\rangle & \to & \texttt{répéter}\ V\ \texttt{fois}\ [S] \\
V & \to & r_N \\
N & \to & C \mid CN \\
C & \to & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

▲

Exemple 4.2.

$$
\begin{array}{rcl}
\langle\text{PROGRAMME}\rangle & \to & \texttt{program}\langle\text{ID}\rangle; \\
 & & \langle\text{DECL}\rangle \\
 & & \langle\text{CORPS}\rangle \\
\langle\text{DECL}\rangle & \to & \texttt{var}\langle\text{LISTE VAR}\rangle \\
\langle\text{LISTE VAR}\rangle & \to & \langle\text{LISTE ID}\rangle{:}\langle\text{TYPE}\rangle; \\
\langle\text{LISTE ID}\rangle & \to & \langle\text{ID}\rangle\langle\text{AUTRES ID}\rangle \\
\langle\text{AUTRES ID}\rangle & \to & \varepsilon \mid ,\langle\text{ID}\rangle\langle\text{AUTRES ID}\rangle \\
 & \vdots &
\end{array}
$$

▲

Définition 4.3. (Sipser 2.2)
Une grammaire hors contexte (GHC) est un quadruplet (V, Σ, R, S) où :

- V est un ensemble fini de variables ;
- Σ est un alphabet fini, parfois appelé ensemble de terminaux, où $V \cap \Sigma = \emptyset$;
- R est un ensemble fini de règles de la forme $A \to z$ où $A \in V$ et $z \in (V \cup \Sigma)^*$;
- $S \in V$ est la variable initiale.

▲

Exemple 4.4.

$$G = (V, \Sigma, R, S)$$

où

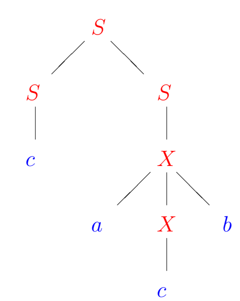
$$\begin{aligned} V &= \{S, X\} \\ \Sigma &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \\ R &= \{S \to SS \mid X \mid \mathtt{c}, \quad X \to \mathtt{a}X\mathtt{b} \mid \mathtt{c}\}. \end{aligned}$$

$$\begin{aligned} V &= \{S, X\} \\ \Sigma &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \\ R &= \{S \to SS \mid X \mid \mathtt{c}, \quad X \to \mathtt{a}X\mathtt{b} \mid \mathtt{c}\} \end{aligned}$$

Dérivation du mot cacb :

$$\begin{aligned} S &\Rightarrow SS \\ &\Rightarrow \mathtt{c}S \\ &\Rightarrow \mathtt{c}X \\ &\Rightarrow \mathtt{ca}X\mathtt{b} \\ &\Rightarrow \mathtt{cacb} \end{aligned}$$

Arbre de dérivation de cacb



▲

Soit une GHC $G = (V, \Sigma, R, S)$.

Rappel : un mot est un élément de Σ^*.

Définition 4.5. Une forme est un élément de $(V \cup \Sigma)^*$. ▲

Remarque 4.6. “hors-contexte” car les règles régissent le remplacement d’une variable A dans uAv indépendamment du contexte uv. ▲

Soit $G = (V, \Sigma, R, S)$ une GHC et soient u, v, w des formes.

Définition 4.7. On dit que uAv donne uwv, noté

$$uAv \Rightarrow uwv,$$

si $A \to w$ est une règle de R.

On note $u \overset{n}{\Rightarrow} v$ si u donne v en n étapes :

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_n = v.$$

Aussi, $u \overset{*}{\Rightarrow} v$ s’il existe un $n \geq 0$ tel que $u \overset{n}{\Rightarrow} v$. ▲

Définition 4.8. Soit $G = (V, \Sigma, R, S)$ une GHC.

$$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

est le langage engendré par G. ▲

Définition 4.9. Un langage Y est hors-contexte si il existe une GHC G telle que $Y = L(G)$. ▲

Définition 4.10. La classe des langages hors-contexte est notée HC. ▲

Exemple 4.11. Soit $G = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$ où

$$R = \{S \to \mathtt{a}S\mathtt{b} \mid \varepsilon\}.$$

Alors

$$\begin{aligned} L(G) &= \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \ldots\} \\ &= \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}. \end{aligned}$$

Dérivation du mot aaabbb par exemple :

$$\begin{aligned} S &\Rightarrow \mathtt{a}S\mathtt{b} \\ &\Rightarrow \mathtt{aa}S\mathtt{bb} \\ &\Rightarrow \mathtt{aaa}S\mathtt{bbb} \\ &\Rightarrow \mathtt{aaabbb} \end{aligned}$$

▲

Exemple 4.12. Soit (V, Σ, R, S) la GHC où

$$\begin{aligned} V &= \{S, M, N\} \\ \Sigma &= \{0, 1, \ldots, 9, +, *, (,)\} \end{aligned}$$

$$R : \begin{cases} S \to N \mid (S{+}S) \mid (S{*}S) \\ N \to 1M \mid 2M \mid \ldots \mid 9M \\ M \to \varepsilon \mid 0M \mid 1M \mid \ldots \mid 9M \end{cases}$$

$$R : \begin{cases} S \to N \mid (S{+}S) \mid (S{*}S) \\ N \to 1M \mid 2M \mid \ldots \mid 9M \\ M \to \varepsilon \mid 0M \mid 1M \mid \ldots \mid 9M \end{cases}$$

La variable N donne les nombres positifs :

$$N \Rightarrow 2M \Rightarrow 23M \Rightarrow 23.$$

Et S donne des expressions arithmétiques :

$$\begin{aligned} S \Rightarrow (S{+}S) &\Rightarrow (S{+}(S{*}S)) \\ &\overset{*}{\Rightarrow} (N{+}(N{*}N)) \\ &\overset{*}{\Rightarrow} (123 + (245 * 19)) \end{aligned}$$ ▲

Exemple 4.13. Soit $G = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$ la GHC où

$$R \ : \ S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon.$$

Quel est le langage $L(G)$?

Vérifions que

$$L(G) = Y$$

où

$$Y = \{w \ : \ |w|_{\mathtt{a}} = |w|_{\mathtt{b}}\}.$$

$$R \ : \ S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

Preuve. Il est facile de voir que $L(G) \subseteq Y$ car la partie droite de chaque règle ci-dessus est une forme σ telle que $|\sigma|_a = |\sigma|_b$.

Il reste à montrer que $Y \subseteq L(G)$. Prouvons par induction généralisée que pour tout n,

$$w \in Y \text{ et } |w| = n \ \Rightarrow \ w \in L(G).$$

$$R \ : \ S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

Base de l'induction. Soit $n = 0$. Alors

$$w \in Y \text{ et } |w| = n \ \Rightarrow \ w = \varepsilon.$$

Or

$$S \ \Rightarrow \ \varepsilon$$

donc $w \in L(G)$.

$$R \ : \ S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

Pas de l'induction généralisée. Soit $n > 0$. Notre hypothèse d'induction est :

$$z \in Y \text{ et } |z| < n \ \Rightarrow \ S \overset{*}{\Rightarrow} z.$$

Nous devons considérer $w \in Y$ de longueur n.
Remarquons que tous les mots $w \in Y$ sont de longueur paire.
Considérons 4 cas pour w :

$$R \;:\; S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

1) $w = \mathtt{a}z\mathtt{b}$. Dans ce cas, on a la dérivation :

$$\begin{aligned} S &\Rightarrow \mathtt{a}S\mathtt{b} \\ &\overset{*}{\Rightarrow} \mathtt{a}z\mathtt{b} \quad \text{(par hypothèse d'induction)} \\ &= w. \end{aligned}$$

$$R \;:\; S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

2) $w = \mathtt{b}z\mathtt{a}$. Alors

$$\begin{aligned} S &\Rightarrow \mathtt{b}S\mathtt{a} \\ &\overset{*}{\Rightarrow} \mathtt{b}z\mathtt{a} \quad \text{(par hypothèse d'induction)} \\ &= w. \end{aligned}$$

$$R \;:\; S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

3) $w = \mathtt{a}z\mathtt{a}$. Dans ce cas, z contient deux $\mathtt{b}$ de plus que de $\mathtt{a}$, et on peut donc le décomposer en $z = z_1 \cdot z_2$, où z_1 et z_2 contiennent chacun un $\mathtt{b}$ de plus que de $\mathtt{a}$. D'où :

$$\begin{aligned} S &\Rightarrow SS \\ &\overset{*}{\Rightarrow} \mathtt{a}z_1S \quad \text{(car } \mathtt{a}z_1 \in Y \text{ et par hyp. d'ind.)} \\ &\overset{*}{\Rightarrow} \mathtt{a}z_1z_2\mathtt{a} \quad \text{(car } z_2\mathtt{a} \in Y \text{ et par hyp. d'ind.)} \\ &= \mathtt{a}z\mathtt{a} \\ &= w. \end{aligned}$$

$$R \;:\; S \rightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon$$

4) $w = \mathtt{b}z\mathtt{b}$. Ce cas se traite comme le cas 3 *mutatis mutandis*.

Dans tous les cas, on a $w \in L(G)$. ▲

Construction de grammaires

Construire une GHC pour un langage donné demande souvent de la créativité.

Cependant, un certain nombre de techniques peuvent aider, dont :

- décomposer le langage en l'union de deux langages plus simples ;
- utiliser des règles qui produisent des suites.

Exemple 4.14. Soit

$$Y = \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k\mathtt{b}^l \mid (i = k \text{ ou } i = l);\ j \geq 0;\ k, l \geq 1\},$$

c'est-à-dire :

$$\begin{aligned} Y &= \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^i\mathtt{b}^l \mid i \geq 1,\ j \geq 0,\ l \geq 1\} \\ &\cup \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k\mathtt{b}^i \mid i \geq 1,\ j \geq 0,\ k \geq 1\}. \end{aligned}$$

Une GHC pour Y est donnée par :

$$G = (\{S, S_1, S_2, A, B, C, D\},\ \{\mathtt{a}, \mathtt{b}\},\ R,\ S)$$

avec

$$R : \begin{cases} S & \to & S_1 \mid S_2 \\ S_1 & \to & C\mathtt{b}B \\ C & \to & \mathtt{a}C\mathtt{a} \mid \mathtt{a}B\mathtt{a} \\ B & \to & \mathtt{b}B \mid \varepsilon \\ S_2 & \to & \mathtt{a}S_2\mathtt{b} \mid \mathtt{a}D\mathtt{b} \\ D & \to & B\mathtt{a}A \\ A & \to & \mathtt{a}A \mid \varepsilon \quad \blacktriangle \end{cases}$$

Ambiguïté

Définition 4.15. Un mot w est ambigu selon une GHC G s'il possède au moins deux arbres de dérivation distincts selon G. ▲

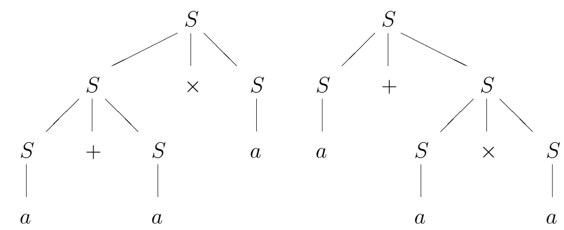
Remarque 4.16. Un arbre de dérivation de w donne lieu à plusieurs dérivations de w, mais à une seule dérivation de w à gauche. ▲

Définition 4.17. (Sipser 2.7) Une GHC G est ambiguë s'il existe un mot $w \in L(G)$ qui est ambigu selon G. ▲

Exemple 4.18. La GHC $(\{S\}, \{a, +, \times\}, R, S)$ avec R donné par

$$S \to S \times S \mid S + S \mid a$$

est une grammaire ambiguë.



▲

Définition 4.19. Un langage hors-contexte Y est essentiellement ambigu si toute GHC G telle que $L(G) = Y$ est ambiguë. ▲

Exemple 4.20. $\{a^i b^j c^k : i = j \text{ ou } j = k\}$ est un langage essentiellement ambigu. ▲

Remarque 4.21. Prouver qu'un langage est essentiellement ambigu n'est pas une mince tâche ! ▲

Forme normale de Chomsky

Jusqu'où peut-on simplifier une GHC sans changer le langage qu'elle décrit ?

Définition 4.22. (Sipser 2.8) (V, Σ, R, S) est une grammaire en forme normale de Chomsky (une GFNC) si... ▲

...toutes ses règles sont

$$\begin{array}{rcl} \text{de la forme } A & \to & BC \\ \text{ou de la forme } A & \to & a \end{array}$$

avec

$$\begin{array}{rcl} A, B, C & \in & V, \\ B & \neq & S, \\ C & \neq & S, \\ a & \in & \Sigma, \end{array}$$

en permettant aussi

$$S \to \varepsilon.$$

▲

Théorème 4.23. (Sipser 2.9) Tout langage hors-contexte est $L(G)$ pour une certaine GFNC G.

Aperçu de la preuve. Exhibons un algorithme qui prend en entrée une GHC G et qui retourne une GFNC G' telle que $L(G) = L(G')$:

Algorithme de conversion d'une GHC en forme normale de Chomsky

En entrée : $G = (V, \Sigma, R, S)$, une GHC.

1. Ajouter la nouvelle variable initiale S_0 à V et ajouter la règle $S_0 \to S$ à R.

2. Éliminer de R toutes les règles de la forme $A \to \varepsilon$, sauf possiblement la règle $S_0 \to \varepsilon$, en itérant les étapes suivantes :

2.1. Enlever $A \to \varepsilon$.

2.2. Pour toute règle $B \to uAv$ faire :
ajouter $B \to uv$, à moins que $uv = \varepsilon$ et que la règle $B \to \varepsilon$ ait déjà été traitée en **2.1**.

3. Éliminer de R toutes les règles unitaires, ie de la forme $A \to B$ où $B \in V$, en itérant les étapes suivantes :

3.1. Enlever $A \to B$.

3.2. Pour toute règle $B \to w$ faire :
ajouter $A \to w$, à moins que $w \in V$ et que la règle $A \to w$ ait déjà été traitée en **3.1**.

4. Éliminer de R toutes les règles de la forme

$$A \to u_1 u_2 \dots u_k$$

où $k \geq 3$ et où $u_1, u_2, \dots, u_k \in V \cup \Sigma$:

4.1. Enlever $A \to u_1 u_2 \dots u_k$.

4.2. Ajouter $A_1,\ A_2,\ \dots,\ A_{k-2}$ à V.

4.3. Ajouter les règles

$$\begin{aligned} A &\to u_1 A_1 \\ A_1 &\to u_2 A_2 \\ &\dots \\ A_{k-2} &\to u_{k-1} u_k. \end{aligned}$$

5. Éliminer de R toutes les règles de la forme $A \to aB$ où $a \in \Sigma$ et $B \in V$:

5.1. Enlever $A \to aB$.

5.2. Ajouter A' à V.

5.3. Ajouter les règles $A \to A'B$ et $A' \to a$.

Idem pour les règles de la forme $A \to Ba$.

Le résultat est la GFNC G'. ▲

Exemple 4.24.

$$G = (\{S, A, B\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$$

où

$$R \ : \ \begin{cases} S & \to \quad ASA \mid \mathtt{a}B \\ A & \to \quad B \mid S \\ B & \to \quad \mathtt{b} \mid \varepsilon \end{cases}$$

Étape 1. Ajouter une nouvelle variable initiale.

Avant :

$$\begin{aligned} S &\rightarrow ASA \mid \mathtt{a}B \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

Étape 2. Éliminer les règles ε.

Avant :

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \end{aligned}$$

Étape 3. Éliminer les règles unitaires.

Avant :

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\rightarrow \mathtt{b} \mid ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ B &\rightarrow \mathtt{b} \end{aligned}$$

Étape 4. Éliminer les règles à formes trop longues.

Avant :

$$\begin{aligned} S_0 &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS & A &\rightarrow \mathtt{b} \mid ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS & B &\rightarrow \mathtt{b} \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\rightarrow \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} & A &\rightarrow \mathtt{b} \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AA_1 \\ S_{0,1} &\rightarrow SA & A_1 &\rightarrow SA \\ S &\rightarrow \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 & B &\rightarrow \mathtt{b} \\ S_1 &\rightarrow SA \end{aligned}$$

Étape 5. Éliminer les règles mixtes.

Avant :

$$\begin{aligned} S_0 &\to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} & A &\to \mathtt{b} \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AA_1 \\ S_{0,1} &\to SA & A_1 &\to SA \\ S &\to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 & B &\to \mathtt{b} \\ S_1 &\to SA \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\to S_0{}'B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} & S_1 &\to SA \\ S_0{}' &\to \mathtt{a} & A &\to \mathtt{b} \mid A'B \mid \mathtt{a} \mid SA \mid AS \mid AA_1 \\ S_{0,1} &\to SA & A' &\to \mathtt{a} \\ S &\to S'B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 & A_1 &\to SA \\ S' &\to \mathtt{a} & B &\to \mathtt{b} \end{aligned}$$

▲

Remarque 4.25. L'exemple précédent peut être simplifié à la GFHC suivante :

$$\begin{aligned} S_0 &\to DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\ S &\to DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\ A &\to \mathtt{b} \mid DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\ B &\to \mathtt{b} \\ C &\to SA \\ D &\to \mathtt{a} \end{aligned}$$

▲

Deux évidences sur une grammaire en forme normale de Chomsky

Soit $G = (V, \Sigma, R, S)$ une GFNC.

Remarque 4.26. Si $a \in \Sigma \cup \{\varepsilon\}$ et $a \in L(G)$, alors forcément

$$S \to a$$

est une règle de G. ▲

Deux évidences (suite et fin)

Remarque 4.27. Si $A \in V$ et $A \overset{*}{\Rightarrow} w_1 w_2 \ldots w_n$ où $n > 1$, alors forcément $A \overset{*}{\Rightarrow} BC$ où $B, C \in V$ et

$$B \overset{*}{\Rightarrow} w_1 w_2 \ldots w_{i-1}$$

et

$$C \overset{*}{\Rightarrow} w_i w_{i+1} \ldots w_n$$

pour un certain $1 < i \leq n$. ▲

Appartenance d'un mot à un langage hors-contexte

Théorème 4.28. Il existe un algorithme qui prend en entrée un mot w et une GHC G et qui détermine si $w \in L(G)$.

Preuve.
Étape 1 : mettre en forme de Chomsky.
Étape 2 : si $w \neq \varepsilon$, dire oui ssi $S \overset{n}{\Rightarrow} w$,
où $n = 2|w| - 1$. (Pourquoi ça marche ?) ■

Remarque 4.29. Pas brillant ! ▲

Un bien meilleur algorithme

Théorème 4.30. Pour G fixée, il est possible de déterminer si $w_1 w_2 \dots w_n \in L(G)$ en temps $O(n^3)$.

Aperçu de la preuve.
Étape 1 : supposer (V, Σ, R, S) une GFNC.
Étape 2 : construire un tableau T dont l'entrée (i, j), $1 \leq i \leq j \leq n$, contient

$$\{A \in V : A \overset{*}{\Rightarrow} w_i w_{i+1} \dots w_j\}.$$

Étape 3 : accepter $w_1 w_2 \dots w_n$ ssi $S \in T(1, n)$. ■

Pour les curieu(x + ses)

Le tableau T se construit par balayage en partant de la diagonale principale.

C'est une application classique de la technique de la programmation dynamique.

Remarque 4.31. Lorsque G est la grammaire d'un langage de programmation, l'analyse syntaxique d'un mot $w \in L(G)$ est la première tâche d'un compilateur de ce langage. ▲

En réalité, des propriétés plus fines que "hors-contexte" sont exploitées pour la compilation car $O(n^3)$ serait trop inefficace.

Par ailleurs, l'analyse syntaxique est compliquée du fait de devoir
- détecter sans délai les erreurs de syntaxe,
- interagir avec l'analyse sémantique.

Théorème 4.32.

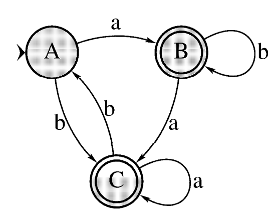
$$\mathsf{REG} \subseteq \mathsf{HC}$$

Preuve. Un langage régulier est $L(M)$ pour un AFD $M = (Q, \Sigma, \delta, q_0, F)$. On voit facilement que

$$\begin{aligned} V &= Q, \\ R &= \{(A \to xB) \;:\; A, B \in Q,\ x \in \Sigma,\ \delta(A, x) = B\} \\ &\quad \cup \{(A \to \varepsilon) \;:\; A \in F\} \text{ et} \\ S &= q_0 \end{aligned}$$

définissent une GHC $G = (V, \Sigma, R, S)$ qui vérifie $L(G) = L(M)$.

Exemple de cette construction :



$G = (\{A, B, C\}, \{a, b\}, R, A)$ où R est défini par

$$\begin{aligned} A &\to aB \mid bC \\ B &\to bB \mid aC \mid \varepsilon \\ C &\to bA \mid aC \mid \varepsilon. \end{aligned}$$ ▲

Corollaire 4.33. La classe REG est strictement incluse dans la classe HC.

Preuve. Soit $Y = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$.

Nous avons vu en exemple que Y est hors-contexte, et au chapitre précédent que Y n'est pas régulier. ■

Peut-on montrer qu'un langage n'est PAS hors-contexte ?

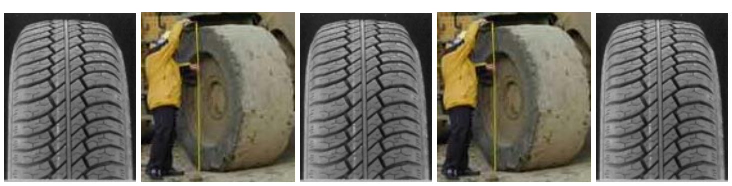
Intuition : une GHC peut balancer des parenthèses à l'aide de

$$A \to (A) \;\Rightarrow\; ((A)) \;\Rightarrow\; (((A))) \;\overset{*}{\Rightarrow}\; \underbrace{((\cdots(}_{n \text{ fois}} A \underbrace{))\cdots)}_{n \text{ fois}}$$

mais pourrait-elle en même temps compter n répétitions d'un 3ième symbole, par exemple

$$A \overset{*}{\Rightarrow} \underbrace{((\cdots(}_{n \text{ fois}} \underbrace{\$\$\cdots\$}_{n \text{ fois}} \underbrace{))\cdots)}_{n \text{ fois}} \quad ?$$

Lemme du bonhomme Michelin à cinq pneus
(dixit Claude Christen)
(alias "du Michelin hors-contexte")
(alias "du pompiste hors-contexte")
(alias "Pumping lemma for context-free languages")



(http ://tcpneus.ifrance.com/tcpneus/images/Classic%20serie%2080.jpg)
(http ://www.polarsolid.com/solid_tires_picture_6.htm)

Énoncé formel du lemme du Michelin hors-contexte

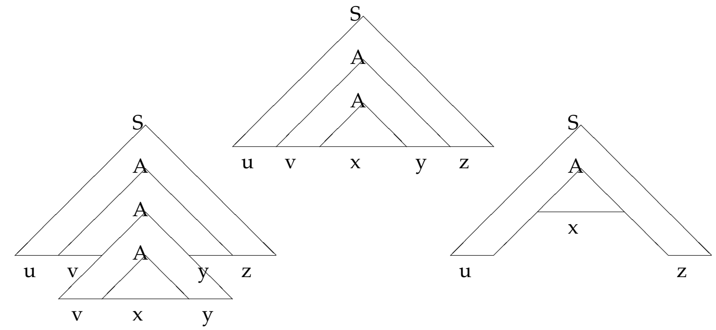
Lemme 4.34. (Sipser 2.34)
Soit $Y \in$ HC. Il existe un entier $p \geq 1$ tel que pour tout $w \in Y$ vérifiant $|w| \geq p$, on peut écrire $w = uvxyz$ de manière à ce que

- $|vy| > 0$ et
- $|vxy| \leq p$ et
- $uv^ixy^iz \in Y$ pour tout $i \geq 0$.

Aperçu de la preuve.

Si un mot $w \in Y$ est suffisamment long, alors il aura un arbre de dérivation très haut.

Cet arbre possèdera un chemin, du sommet à une feuille, avec répétition d'une certaine variable, disons A, comme ci-dessous :



Exemple 4.35. Le langage

$$Y = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$$

n'est pas hors-contexte.

Preuve.
Supposons au contraire que $Y \in \mathsf{HC}$.
Soit $p \geq 1$ tel que fourni par le lemme du Michelin hors contexte.

Prenons le mot $w = \mathtt{a}^p\mathtt{b}^p\mathtt{c}^p$.

On a bien $w \in Y$ et $|w| = 3p \geq p$.

Est-il possible de décomposer w en $uvxyz$ tout en respectant $|vy| > 0$ et $|vxy| \leq p$ et $uv^ixy^iz \in Y$ pour tout $i \geq 0$?

Essayons. Regroupons les cas possibles d'une décomposition $w = uvxyz$ vérifiant déjà $|vy| > 0$ et $|vxy| \leq p$.

Cas 1. Le sous-mot v ou le sous-mot y contient plus d'une sorte des symboles a, b ou c.

Alors uv^2xy^2z n'est forcément pas de la forme $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ (pourquoi ?)

Donc aucune des décompositions du présent cas ne satisfait $uv^ixy^iz \in Y$ pour tout i.

Cas 2. Les sous-mots v et y contiennent chacun une seule sorte de symbole.

Alors $uv^0xy^0z = uxz$ n'est forcément pas de la forme $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ (pourquoi ?)

Donc aucune des décompositions du cas 2 ne satisfait $uv^ixy^iz \in Y$ pour tout i.

Puisque toutes les décompositions pertinentes de w en $uvxyz$ violent la promesse du lemme, nous obtenons la contradiction désirée.

Donc $Y \notin$ HC. ▲

Propriétés de fermeture

Théorème 4.36. HC est fermée sous l'union.
Preuve. Soient

$$L_1 = L(G_1) \text{ où } G_1 = (V_1, \Sigma, R_1, S_1)$$
$$L_2 = L(G_2) \text{ où } G_2 = (V_2, \Sigma, R_2, S_2)$$

en supposant que V_1 et V_2 sont disjoints, de même que R_1 et R_2.

Alors la GHC suivante :

$$G = (V_1 \cup V_2 \cup \{S\},\ \Sigma,\ R_1 \cup R_2 \cup \{S \rightarrow S_1 \mid S_2\},\ S)$$

engendre le langage $L_1 \cup L_2$. ■

Surprise

Théorème 4.37. La classe HC n'est pas fermée sous l'intersection.

Preuve. Soient

$$\begin{aligned} L_1 &= \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n \mid m \geq 0, n \geq 0\}, \\ L_2 &= \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m \mid m \geq 0, n \geq 0\}. \end{aligned}$$

Alors

$$L_1 \cap L_2 = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}.$$

Or nous avons démontré en exemple que ce langage n'est pas hors-contexte. ■

Sur...prise II

Théorème 4.38. La classe HC n'est pas fermée sous la complémentation.

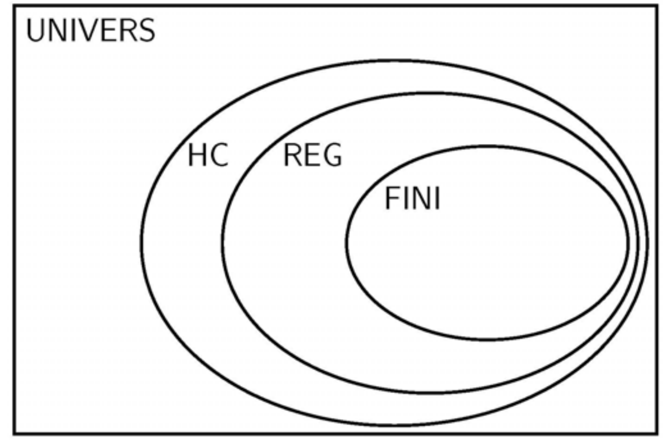
Preuve. Supposons au contraire que HC est fermée sous la complémentation.

Alors HC est fermée sous intersection puisque HC est fermée sous union (théorème 4.36) et pour tout L_1 et L_2,

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

Ceci contredit le théorème 4.37. ■

Le monde des langages, prise 2



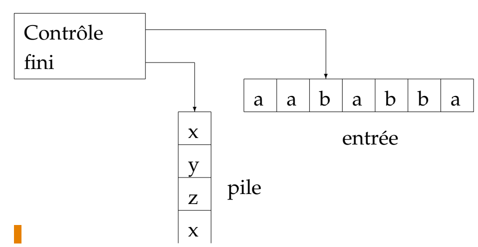
Un modèle de type "automate" pour les langages hors-contexte

Que manquerait-il à un automate fini pour être capable de reconnaître, par exemple, le langage non régulier $\{ww^R : w \in \{a, b\}^*\}$?

On aimerait :
- préserver le nombre fini d'états
- préserver le parcours de droite à gauche du mot lu.

Réponse : une pile !

L'automate à pile vu informellement



L'automate à pile vu formellement

Définition 4.39. (Sipser 2.13)
Un automate à pile non déterministe (APN) est un 6-tuplet $(Q, \Sigma, \Gamma, \delta, q_0, F)$ où

- $Q, \Sigma, q_0, F \subseteq Q$ sont comme pour un AFD,
- Γ est l'**alphabet de pile** (fini),
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$
  est la **fonction de transition**.

▲

Fonctionnement de l'automate à pile

Si $(q, Z) \in \delta(p, a, X)$ et
- p est l'état courant et
- a est le prochain symbole de l'entrée et
- X est sur le dessus de la pile,

alors

une des transitions possibles consiste à
- dépiler X et
- empiler Z et
- passer de l'état p à l'état q.

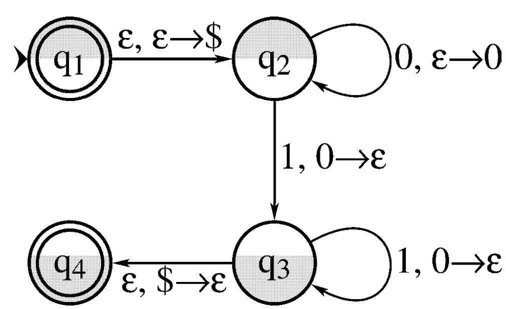
Remarque 4.40. Pour accepter, un APN doit

atteindre un état final
et
avoir lu toute son entrée.

▲

Définition 4.41. Soit M un APN. Alors $L(M) = \{w \colon M \text{ accepte } w\}$ est le langage accepté par M. ▲

Exemple 4.42.
$M = (\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, \$\}, \delta, q_1, \{q_1, q_4\})$ où δ est défini par



On a $L(M) = \{0^n 1^n \colon n \geq 0\}$. ▲

Deux faits sans preuve

Théorème 4.43. (Sipser 2.20) Un langage Y est hors-contexte ssi $Y = L(\text{un APN } M)$. ■

Définition 4.44. Une restriction naturelle imposée au δ de l'APN donne l'automate à pile déterministe (APD). ▲

Théorème 4.45. Aucun APD n'accepte

$$\{ww^R \,:\, w \in \{a,b\}^*\}$$

mais un APD accepte sur alphabet $\{a,b,c\}$

$$\{wcw^R \,:\, w \in \{a,b\}^*\}.$$

■

Un modèle de type "grammaire" pour les langages réguliers

Quelle restriction faudrait-il imposer aux GHCs pour capturer précisément la classe des langages réguliers ?

Déjà vu de manière implicite dans la preuve que REG $\subseteq$ HC !

Définition 4.46. Une grammaire linéaire à droite (GLD) est une GHC où chaque règle contient au plus une variable, qui doit être le symbole le plus à droite de la règle ▲

Définition 4.47. Une grammaire linéaire à gauche (GLG) est une GHC où chaque règle contient au plus une variable, qui doit être le symbole le plus à gauche de la règle. ▲

Grammaires régulières

Définition 4.48. Une grammaire linéaire à gauche ou linéaire à droite est appelée une grammaire régulière. ▲

Théorème 4.49. Un langage Y est régulier ssi $Y = L(\text{une grammaire régulière})$. ■

Aperçu de la preuve.
Déjà vu : simulation d'un **AFD** par une **GLD**.
Voyons maintenant la simulation d'une **GLD** par un **AFNG** :

Considérons une GLD $G = (V, \Sigma, R, S)$.

L'AFNG ci-dessous accepte $L(G)$:

$$M = (V \cup \{f\}, \Sigma, \delta, S, \{f\})$$

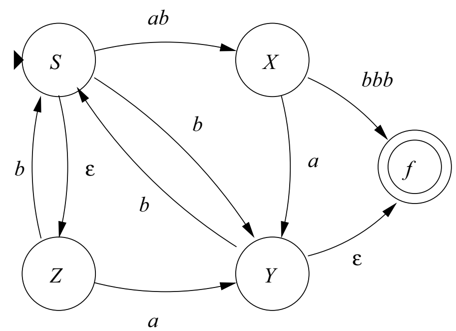
où δ est défini par :

$$\begin{array}{lll} B \in \delta(A, w) & \text{ssi} & A \to wB \text{ est une règle,} \\ f \in \delta(A, w) & \text{ssi} & A \to w \text{ est une règle.} \end{array}$$

En réalité M est un AFN avec des mots pour transitions.

Exemple de cette construction :

$$\begin{array}{lll} S & \to & abX \mid bY \mid Z \\ X & \to & aY \mid bbb \\ Y & \to & bY \mid bS \mid \varepsilon \\ Z & \to & aY \mid bS \end{array}$$



Pour les curieu(x + ses)

De **GLG** à **AFN**

1. Renverser les membres de droite des règles : GLD pour L^R.
2. Construire AFN pour GLD.
3. Renverser l'AFN (transitions, états finaux, état initial).

De **AFN** à **GLG**

1. Renverser l'AFN.
2. Construire GLD.
3. Renverser les membres de droite des règles.

Révision du chapitre

- GHC
- Preuve que $L(G)$ est un certain langage
- Tout langage régulier est hors-contexte
- Lemme du pompiste hors-contexte
- Propriétés de fermeture et de non-fermeture
- Existence de l'automate à pile
- Existence des grammaires régulières