CH.6 Propriétés des langages non contextuels

- 6.1 Le lemme de pompage
- 6.2 Les propriétés de fermeture
- 6.3 Les problèmes de décidabilité
- 6.4 Les langages non contextuels déterministes

6.1 Le lemme de pompage

On a une propriété de "pompage" pour les langages non contextuels. Elle servira entre autres à montrer que certains langages ne peuvent pas être engendrés par une grammaire non contextuelle.

On utilise un lemme relatif aux arbres binaires complets, dont la démonstration est laissée en exercice :

Lemme : Dans un arbre binaire complet de hauteur k, le nombre de nœuds externes est au plus 2^k.

Si x est un nœud d'un arbre, on appellera profondeur de ce nœud la hauteur du sous-arbre enraciné en x.

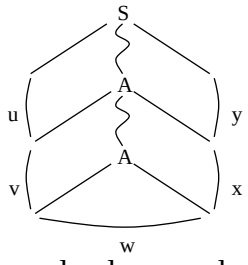
Théorème (lemme de pompage) :
Soit L un langage non contextuel. Il existe un nombre n dépendant uniquement de L tel que la propriété suivante soit vraie.
Si z est un mot de L de longueur $\geq n$, alors z se factorise en $z = uvwxy$, où $|vwx| \leq n$, v et x ne sont pas tous les deux vides et, pour tout $i \geq 0$, on a $uv^iwx^iy \in L$.

Démonstration :
On suppose que L est engendré par une grammaire sous forme normale de Chomsky et que le nombre de variables de cette grammaire est k.
On va montrer que le nombre $n = 2^k$ convient.

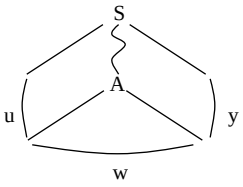
Pour cela, soit z un mot de L et considérons son arbre de dérivation.
Comme la grammaire est sous forme de Chomsky, cet arbre est un arbre binaire complet, sauf pour les feuilles qui sont seuls fils de nœuds. Si on néglige les feuilles, il reste un arbre binaire complet

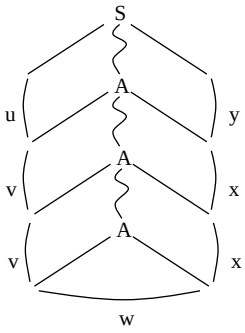
dont les nœuds externes sont en même nombre que la longueur de z.

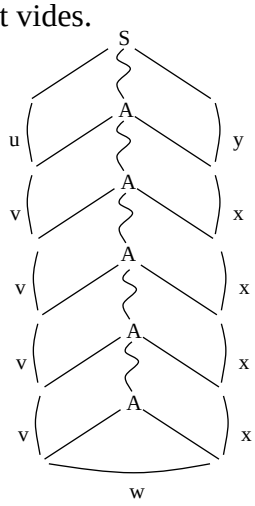
Si la longueur de z est au moins 2^k, la hauteur de l'arbre est au moins k.
Sur un chemin de longueur k étiqueté par des noms de variables, un même nom doit apparaître deux fois (principe des tiroirs).
Considérons les nœuds i qui sont étiquetés par un nom de variable qui apparaît encore une fois sur un chemin partant de i. Il existe au moins un tel i. Parmi tous les i, considérons a, celui qui a la plus petite profondeur et soit A le nom attaché à a.
Cela assure que, dans le sous-arbre enraciné en a, le nom de la racine est répété exactement une fois sur (au moins) un chemin issu de a et qu'aucun chemin issu de a ne contient deux nœuds ayant le même nom.
Le sous-arbre enraciné en a est donc de hauteur au plus k. En conséquence, le nombre de ses nœuds externes est au plus 2^k.
La situation est la suivante :



La longueur de vwx est au plus $2^k = n$.
Puisque la grammaire est sous forme de Chomsky, chaque nœud interne a exactement deux fils. Donc v et x ne sont pas simultanément vides.
Enfin, le plus petit sous-arbre de racine A peut être mis à la place du plus grand, produisant le mot uwy. Le plus grand peut être raccroché au plus petit autant de fois qu'on veut, produisant uv^iwx^iy.







Exemples :
$L_1 = \{0^n1^n2^n, n \geq 0\}$ n'est pas non contextuel.

Si c'était le cas, soit n le nombre dépendant de L_1 et considérons
$z = 0^n1^n2^n = uvwxy$.
Puisque vwx est de longueur au plus n, au moins l'une des lettres 0, 1 ou 2 n'y apparaît pas, d'où contradiction avec le fait que $uwy \in L_1$.

Le même raisonnement vaut pour les mots ayant autant de 0, de 1 et de 2.

$L_2 = \{a^ib^ja^ib^j, i, j \geq 1\}$ n'est pas non contextuel.

On fait de même en examinant $z = a^nb^na^nb^n$. Le facteur vwx ne peut alors contenir des a des deux facteurs a^n, d'où la contradiction.

6.2 Les propriétés de fermeture

On utilise la définition par grammaire ou celle par automate à pile.

Théorème : Les langages non contextuels sont fermés par les opérations régulières.

Démonstration : On part des grammaires G_1 et G_2 engendrant les langages L_1 et L_2. On fabrique une grammaire G engendrant $L_1 \cup L_2$ en introduisant un nouvel axiome pouvant produire l'un ou l'autre des axiomes de G_1 et de G_2. Pour la concaténation, ce nouvel axiome produit la concaténation des axiomes de G_1 et de G_2. Pour la fermeture transitive de L_1, si S_1 est son axiome, le nouvel axiome S produit : $S \rightarrow \varepsilon \mid S_1S$.

Théorème : Les langages non contextuels sont fermés par substitution.

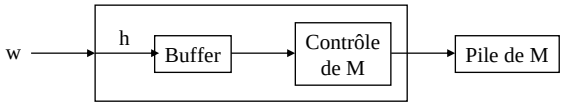
Démonstration : Cela revient à dire que chaque terminal d'un langage devient l'axiome d'un autre langage. Ceci passe facilement aux grammaires.

Corollaire : Les langages non contextuels sont fermés par homomorphisme.

Théorème : Les langages non contextuels sont fermés par homomorphisme inverse.

Démonstration : Soit h un homomorphisme.
Si L est un langage non contextuel, $h^{-1}(L) = \{w : h(w) \in L\}$.
On raisonne avec les automates à pile comme on l'avait fait pour les langages réguliers.

Automate à pile pour $h^{-1}(L)$



Contrôle de M'

On entre un caractère a ; il est stocké dans le buffer (file). Puis celui-ci est vidé caractère par caractère.

Supposons que $\Delta = \{0, ...\}$ et $\Sigma = \{a, ...\}$.

Formellement, si $L = L(M)$ avec $M = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$, on fabrique $M' = (Q', \Sigma, \Gamma, \delta', [q_0, \varepsilon], Z_0, F \times \{\varepsilon\})$.

Les états sont un couple formé d'un état de M et d'un suffixe de h(a).

$\delta'([q, \varepsilon], a, Y) = ([q, h(a)], Y)$ chargement du buffer ;

$\delta'([q, x], \varepsilon, Y) = \{([p, x], \gamma) : (p, \gamma) \in \delta(q, \varepsilon, Y)\}$ transition vide ;

$\delta'([q, 0x], \varepsilon, Y) = \{([p, x], \gamma) : (p, \gamma) \in \delta(q, 0, Y)\}$ transition sur 0.

La reconnaissance par état final permet d'assurer que le buffer est bien vide après lecture de w.

Attention : Les langages non contextuels ne sont pas fermés par intersection. Ils ne le sont donc pas non plus par complémentation (identités de de Morgan).

Exemple :

$L_1 = \{0^n1^n2^m, n, m \geq 0\}$ et $L_2 = \{0^m1^n2^n, n, m \geq 0\}$ sont tous deux non contextuels, car produits d'un langage non contextuel et d'un langage régulier (donc non contextuel).

Néanmoins, $L_1 \cap L_2 = \{0^n1^n2^n, n \geq 0\}$, dont on a vu qu'il ne l'est pas.

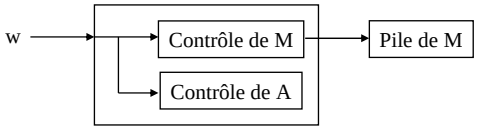
L'explication vient de ce que, si on fait fonctionner en parallèle deux automates à pile, on a besoin de deux piles. Si l'un des deux automates est un automate fini (sans pile), le même fonctionnement se fait avec une seule pile. D'où le théorème suivant.

Théorème : Si L est non contextuel et R est régulier, alors $L \cap R$ est non contextuel.

Démonstration : Si L = L(M) avec $M = (Q_M, \Sigma, \Gamma, \delta_M, q_0, Z_0, F_M)$ (reconnaissance par état final) et R = L(A) avec $A = (Q_A, \Sigma, \delta_A, p_0, F_A)$, déterministe (non indispensable ici).

On fabrique $M' = (Q_A \times Q_M, \Sigma, \Gamma, \delta, [p_0, q_0], Z_0, F_A \times F_M)$, avec
$\delta([p, q], a, Y) = \{([p', q'], \gamma) : \delta_A(p, a) = p' \text{ et } (q', \gamma) \in \delta_M(q, a, Y)\}$ si $a \neq \varepsilon$
$\delta([p, q], \varepsilon, Y) = \{([p, q'], \gamma) : (q', \gamma) \in \delta_M(q, \varepsilon, Y)\}$ si $a = \varepsilon$.

Automate à pile pour $L \cap R$



Exemple : $L = \{ww : w \in (a + b)^*\}$, ensemble des carrés, ne peut pas être engendré par une grammaire non contextuelle.

Si L était non contextuel, il en serait de même de $L \cap aa^*bb^*aa^*bb^*$, qui vaut $\{a^i b^j a^i b^j, i, j \geq 1\}$, d'où une contradiction.

Les langages non contextuels sont également fermés par d'autres opérations : quotient par un langage régulier, préfixe, permutation circulaire, image-miroir.
Les démonstrations sont identiques à celles utilisées pour les langages réguliers.

6.3 Les problèmes de décidabilité

Dans le cas des langages non contextuels, un certain nombre de problèmes sont indécidables. La démonstration de l'indécidabilité est impossible ici. Elle nécessite le développement de la théorie de la calculabilité.

Dans le cas d'un problème concernant les langages non contextuels, le problème de leur codage est important. C'est pourquoi ces problèmes sont formulés en termes de problèmes relatifs aux grammaires ou aux automates à pile. Puisque le passage entre grammaires et automates à pile est algorithmique (donc effectif), on peut formuler ces problèmes au choix en termes de grammaires ou d'automates à pile.

Théorème : Si G est une grammaire non contextuelle, les problèmes suivants sont décidables :

1.— Le langage engendré par G est non-vide ;

2.— Le langage engendré par G est infini.

Démonstration :

Pour le premier problème, on applique l'algorithme d'élimination des variables inutiles : le langage est non-vide si et seulement si l'axiome est un symbole utile.

Pour le second, on peut utiliser le lemme de pompage (exercice) en montrant que le langage est infini si et seulement s'il contient un mot de longueur comprise entre n et 2n.

On peut plus efficacement procéder directement : on suppose que G est sous forme de Chomsky sans variable inutile (algorithme pour cela), puis on fabrique un graphe entre variables dans lequel si $A \rightarrow BC$ est une production, on place un arc de A vers B et de A vers C. On montre (exercice) que le langage est infini si et seulement si ce graphe contient un cycle ou un circuit. Ce graphe contient les informations relatives à la récursivité dans la grammaire.

Théorème : Soit x un mot en symboles terminaux et G une grammaire non contextuelle ; le problème suivant est décidable : $x \in L(G)$.

Démonstration : Mettre la grammaire sous forme normale de Greibach, puis essayer toutes les dérivations possibles jusqu'à la longueur de x.

Il existe des algorithmes plus efficaces (en $O(n^3)$) tel que celui de Cocke, Younger et Kasami.

Par contre, un certain nombre de problèmes faisant intervenir des grammaires non contextuelles sont indécidables.
Dans la liste, G, G_1 et G_2 désignent de telles gammaires et R est une expression régulière (ou un automate fini).

Les problèmes suivants sont indécidables :

1.— $L(G_1) \cap L(G_2) = \varnothing$;
2.— $L(G) = \Sigma^*$;
3.— $L(G_1) = L(G_2)$;
4.— $L(G_1) \subset L(G_2)$;
5.— $L(G) = R$;
6.— $R \subset L(G)$;
7.— Le complémentaire de L(G) est non contextuel ;
8.— $L(G_1) \cap L(G_2)$ est non contextuel ;
9.— L(G) est régulier ;
10.— L(G) est inhéremment ambigu.

6.4 Les langages non contextuels déterministes

Langages pour lesquels il existe un automate à pile déterministe le reconnaissant. Seront abrégés en langages NCD.

Seuls langages utilisables en pratique car l'analyse syntaxique se fait en temps linéaire. En fait, les algorithmes de génération d'analyseurs syntaxiques produisent effectivement des automates à pile déterministes.

Mais ces langages sont plus délicats à manipuler que les langages non contextuels en général :

Aucun lemme de pompage n'est connu pour les langages NCD.

Théorème : Les langages NCD sont fermés par complémentation.

Démonstration : Pour faire comme pour les langages réguliers (inversion des états terminaux et non terminaux), il faut auparavant normaliser les automates à pile déterministes :

Si L = L(M) où M est déterministe, il existe un automate à pile déterministe M' tels que :

1.— M' lit entièrement tout mot présenté en entrée (création d'un puits) ;

2.— M' n'a aucune transition vide sur un état final.

Ces deux points sont assez délicats à établir complètement.

Autres propriétés de clôture :

Théorème : Les langaes NCD sont fermés par :

1.— quotient par un langage régulier ;

2.— homomorphisme inverse ;

3.— intersection avec un ensemble régulier.

Démonstration :

Le premier point est assez technique, les deux autres se font comme pour les langages non contextuels.

Théorème : Les langages NCD ne sont pas fermés par :
1.— homomorphisme ;
2.— union ;
3.— concaténation ;
4.— fermeture transitive ;
5.— intersection.

Théorème : Il existe des langages non contextuels non déterministes.

On peut en effet montrer que si L est NCD, alors MIN(L), mots de L dont aucun préfixe propre n'est dans L est aussi NCD (Modifier l'automate préalablement normalisé de façon à supprimer toute transition sur un état terminal.)
Si donc $L = \{ww^t\}$ était NCD, alors il en serait de même de
$L_1 = L \cap (01)^*(10)^*(01)^*(10)^* = \{(01)^i(10)^j(01)^j(10)^i : i, j \geq 0, i + j \neq 0\}$;
pareil pour $MIN(L_1) = \{(01)^i(10)^j(01)^j(10)^i : 0 \leq j < i\}$;

Par homomorphisme inverse, il en est de même de
$L_2 = \{a^ib^ja^jb^i : 0 \leq j < i\}$;
mais ce dernier n'est pas un langage non contextuel. Pour montrer ce point, on applique le lemme de pompage sur le mot
$z = a^{n+1}b^na^nb^{n+1}$; puisque z = uvwxy et que $|vwx| \leq n$, selon la place possible de vwx, soit les exposants ne sont plus égaux, soit le nombre de a et de b au centre devient trop grand.

Propriétés de décidabilité.
On n'a aucune caractérisation des langages NCD par leur grammaire. Le seul modèle est l'automate à pile déterministe. En effet,

Théorème : Soit G une grammaire non contextuelle ; le problème L(G) est NCD est indécidable.

Ce théorème est admis.

Théorème : Si M est un automate à pile déterministe, et R une expression régulière, les problèmes suivants sont décidables :

1.— $L(M) = R$;
2.— $R \subset L(M)$;
3.— $L(M) = \Sigma^*$;
4.— $L(M)$ est régulier.

Démonstration : Trois premiers points faciles, quatrième difficile.

Théorème : Si M et M' sont deux automates à pile déterministes, les problèmes suivants sont indécidables :

1.— $L(M) \cap L(M') = \varnothing$;
2.— $L(M) \subset L(M')$;
3.— $L(M) \cap L(M')$ est un langage NCD ;
4.— $L(M) \cap L(M')$ est un langage non contextuel ;
5.— $L(M) \cup L(M')$ est un langage NCD.

Théorème : Si M et M' sont deux automates à pile déterministes, le problème suivant est décidable : $L(M) = L(M')$.

La démonstration en est récente. L'algorithme n'est pas (pas encore ?) praticable.

Enfin, l'importance pratique des langages NCD est contenue dans le théorème suivant :

Théorème : Un langage L est analysable par une technique ascendante LR si et seulement si il est L est non contextuel déterministe et préfixe (c'est-à-dire $L = MIN(L)$).

Démonstration : un sens résulte de l'algorithme de construction d'un analyseur syntaxique (*cf.* cours de traduction). L'autre sens résulte de ce que la construction automate - grammaire produit une grammaire LR(0) lorsque l'automate est déterministe.