Hiérarchie de Chomsky

Hiérarchie de Chomsky

On détermine la classe et la complexité des langages (et des grammaires) en fonction d'un certain nombre de contraintes sur la forme des règles de production. 4 ou 5 types de grammaire.

Types de grammaire

- Type 0
- Type 1
- Type 2
- Type 3
- (Type 4)

Hiérarchie de Chomsky

Grammaire de type 0 ou grammaire générale

Les règles ne sont sujettes à aucune restriction. Il suffit que chaque règle fasse intervenir (au moins) un non terminal à gauche.

Exemple / Contre exemple

- aAbb → ba ou aAbB → ϵ OK
- ab → ba ou ϵ → aa OK

- Elle correspond aux langages récursivement énumérables
- Le problème de l'analyse pour de tels langages est indécidable. Un langage est décidable si pour toute phrase on peut savoir en temps fini si elle est du langage ou pas
- Les langages correspondants sont reconnus par des machines de Turing.

Hiérarchie de Chomsky

Exemple 1 : Grammaire de type 0 ou grammaire générale

Une grammaire qui engendre tous les mots qui contiennent un nombre égal de a, b et c.

$$G_1 \begin{cases} S \to SABC \mid \varepsilon \\ AB \to BA \\ BA \to AB \\ AC \to CA \\ CA \to AC \\ BC \to CB \\ CB \to BC \\ A \to a \\ B \to b \\ C \to c \end{cases}$$

Cette grammaire fonctionne en produisant des chaînes de la forme $(ABC)^n$ puis en permutant les non terminaux, et enfin en produisant les chaînes terminales.

Hiérarchie de Chomsky

Exemple 2 : Les mots jumeaux

Langage vu plus pour les machines de Turing

$$G_2 : \begin{cases} S \to \$S'\$ \\ S \to aAS' \\ S \to bBS' \\ S' \to \varepsilon \\ \$\$ \to \# \end{cases} \quad \begin{cases} Aa \to aA \\ Ab \to bA \\ Ba \to aB \\ Bb \to bB \end{cases} \quad \begin{cases} \$a \to a\$ \\ \$b \to b\$ \\ A\$ \to \$a \\ B\$ \to \$b \end{cases}$$

Hiérarchie de Chomsky

Grammaire de type 1 ou grammaire sensible au contexte (context-sensitive) ou monotone

Les règles sont de la forme $\alpha \to \beta$ avec $|\alpha| \leq |\beta|$. On dit alors que le langage engendré est propre. Exceptionnellement, afin d'engendrer le mot vide, on introduit $S \to \epsilon$ pour autant que S n'apparaisse pas dans le membre de droite d'une production.

Définition alternative des grammaires de type 1

Les règles sont de la forme $\alpha A \gamma \to \alpha \beta \gamma$ avec $\beta \in (V_t \cup V_n)^+$
Ou bien de la forme $S \to \epsilon$ et S n'apparaît dans aucun membre de droite. On préfère souvent $S' \to S + \epsilon$ (langage propre).

- Autrement dit, toute règle comprend un non-terminal entouré de deux mots qui décrivent le contexte dans lequel la variable peut être remplacée.

Définition alternative des grammaires de type 1

- Grammaires dites contextuelles car le remplacement d'un élément non-terminal peut dépendre des éléments autour de lui - son contexte.
- Les langages contextuels qui en résultent sont exactement ceux reconnus par une machine de Turing non déterministe à mémoire linéairement bornée, appelés couramment automates linéairement bornés.

Exemple de grammaire de type 1

Grammaire contextuelle pour le langage $L(G_3) = \{a^{2^n} \mid n > 0\}$

$$G_3 \begin{cases} S \to DT & T \to XT & Xaa \to aaXa \\ S \to AA & T \to aF & XaF \to aaF \\ Daaa \to aaDaa & DaaF \to aaaa \end{cases}$$

Hiérarchie de Chomsky

Grammaire de type 2 : Grammaire hors-contexte (context-free) ou algébrique

Règles de la forme $A \to \beta$ où $A \in V_n$ et pas de restriction sur β, i.e $\beta \in \Sigma^*$

- Elles sont particulièrement étudiées
- Les langages algébriques correspondants sont reconnus par des automates à pile non-déterministes.
- L'analyse de ces langages est polynomiale.

Exemple

$$P = \{S \longrightarrow aSb, S \longrightarrow ab\}$$
$$\Longrightarrow L(G) = \{a^n b^n \mid n > 0\}$$

Hiérarchie de Chomsky

Grammaire de type 3 : Grammaire régulière

Les règles peuvent prendre deux formes :

- linéaire à gauche : $\alpha \rightarrow \beta$ où $\alpha \in V_n$ et $\beta \in V_n V_t^*$ i-e $A \rightarrow Bw$ ou $A \rightarrow w$ avec $A, B \in V_n$ et $w \in V_t^*$
- linéaire à droite : $\alpha \rightarrow \beta$ où $\alpha \in N$ et $\beta \in V_t^* V_n \cup V_t^*$ i-e $A \rightarrow wB$ ou $A \rightarrow w$ avec $A, B \in V_n$ et $w \in V_t^*$

- Les langages réguliers correspondants sont construits et reconnus par des automates finis
- L'analyse de ces langages est polynomiale.

Exemple

$$P = \{A \rightarrow aB, A \rightarrow a, B \rightarrow b\}$$

Hiérarchie de Chomsky

Grammaire de type 4

Les parties droites de toutes les règles sont des terminaux.
Les règles sont de la forme $X \to \alpha$ où $X \in V_n$ et $\alpha \in V_t$

- Une telle grammaire ne fait qu'énumérer les phrases de son langage sur V_t.

Hiérarchie de Chomsky

Grammaires	Langages	
Langages récursivement énumérables		Machine de reconnaissance de Turing
G_1	Langages contextuels	Machine à mémoire linéaire bornée
G_2	Langages algébriques	Automates à pile
G_3	Langages réguliers	Automates finis

Dans ce cours, on étudiera :

- Les langages réguliers : l'analyse lexicale,
- Les langages algébriques : l'analyse syntaxique.

Définitions de base

Alphabet

- Ensemble Σ constitué de lettres ou caractères ou symboles.
  - Différents des caractères blancs : espaces, tabulations, fin de ligne
- Exemple : {0,1}, {A,C,G,T}, l'ensemble des chiffres, code ASCII.

Mot

- Suite finie (éventuellement vide) d'éléments de Σ
- Un mot u sur l'alphabet Σ est une application
  $u : \{1, \ldots, m\} \rightarrow \Sigma$
  - m est un entier appelé la longueur de u
  - {1, …, m} ensemble des $i \in \mathbb{N}$ tels que $1 \leq i \leq m$
  - u(i) est appelée la $i^{ème}$ lettre, le $i^{ème}$ caractère ou le $i^{ème}$ symbole de u

Définitions de base (Suite)

- L'ensemble des mots (resp. non-vides) formés à partir de Σ est Σ^* (resp. Σ^+)
- La k^{eme} puissance de Σ est notée Σ^k et se définit par : $\Sigma^0 = \{\epsilon\}$ et $\forall k > 0, \Sigma^k = \Sigma^{k-1}\Sigma^1$, ens. des mots de longueur k.
  - Exemple : $\Sigma = \{a, b\}$, alors $\Sigma^0 = \{\epsilon\}, \Sigma^1 = \{a, b\}$, $\Sigma^2 = \{aa, ab, ba, bb\}, ...$
- La longueur de $u \in \Sigma^*$ notée $|u|$ est le nombre total de symboles de u.
  - Lorsque $|u| = 0$, $u : \emptyset \to \Sigma$ est le mot vide, noté ϵ, 1 ou 1_Σ.
  - Lorsque $|u| = 1$, $u : \{1\} \to \Sigma$ est défini par le seul caractère $u(1) \in \Sigma$
- Tout $x \in \Sigma$ est identifié au mot de longueur 1 qu'il définit.
- $|u|_a$ compte le nombre total d'occurrences du symbole a dans u.

Récurrence sur les mots

Adjonction d'une occurrence à droite d'un mot

- Elle se définit de la façon suivante $\forall u : \{1, \ldots, m\} \to \Sigma$ et $\forall x \in \Sigma$
  - $ux : \{1, \ldots, m+1\} \to \Sigma$ est le mot défini par :
    - $ux(i) = u(i)$ pour tout $i \in \{1, \ldots, m\}$
    - $ux(m+1) = x$

Remarque : définition inductive de Σ^*

- Tout élément de Σ^* ou bien est ϵ ou bien s'obtient à partir d'un élément de Σ par "adjonction" d'un caractère à droite :
  - $\epsilon \in \Sigma^*$ (le mot sans caractère)
  - $\forall x \in \Sigma$: si $u \in \Sigma^*$ alors $ux \in \Sigma^*$ (adjonction de x à droite de u)

Récurrence sur les mots(suite)

Construction Mot initial	Résultat	En abrégé
Adjonction de a	ε	ε
Adjonction de a	εa	a
Adjonction de b	εaa	aa
	εaab	aab

TABLE: Adjonction d'occurrence à droite

Exercice 1 :

- Définir par récurrence la longueur d'un mot
- Représenter le mot ababb avec l'alphabet {aba, ab, bb, b}

La concaténation

- Elle est l'opération $?.? : \Sigma^* \times \Sigma^* \to \Sigma^*$ qui consiste à mettre deux mots "bout à bout"
- Pour tout $u \in \Sigma^*$ et tout $v \in \Sigma^*$, $u.v$ est définie par récurrence sur v de la façon suivante :
  - $u.\epsilon = u$
  - $u.(vx) = (u.v)x$ pour tout $v \in \Sigma^*$ et tout $x \in \Sigma$
- Elle est une opération interne de Σ^* ;
- Elle est associative, mais pas commutative.
- ϵ est l'élément neutre pour la concaténation : $u\epsilon = \epsilon u = u$; ceci justifie la notation 1 ou encore 1_Σ .

Sous-mot

- Un sous-mot v est une sous-suite de la suite des symboles d'un mot w.
- Le facteur v d'un mot w est un mot tel qu'il existe des mots u et u' tels que w = u.v.u'.
  - Un facteur gauche (ou préfixe) est un facteur tel que u = ε.
  - Un facteur droit (ou suffixe) est un facteur tel que u' = ε.
- Un mot fini u est périodique si $u = x^n$ pour $n \geq 2$. (# primitif)
- On notera u^n la concaténation de n copies de u, $u^0 = \varepsilon$.
- Si u se factorise sous la forme u = xy, alors on écrira parfois $y = x^{\overline{1}}u$ et $x = uy^{\overline{1}}$.

Généralités sur les Langages

- Un langage L se définit à partir d'un alphabet Σ comme étant un sous- ensemble de Σ^*
- L'ensemble des langages sur Σ est désigné par $P(\Sigma^*)$
- L'ensemble des parties de Σ^* souvent noté 2^{Σ^*}
- Exemples : $\emptyset \subseteq \Sigma^*$, $\Sigma \subseteq \Sigma^*$, $\Sigma^* \subseteq \Sigma^*$ sont des languages sur Σ

Somme de langages

- La réunion de deux langages L_1 et L_2 de Σ^* est définie par :
  - $L_1 \cup L_2 = L_1 + L_2 = \{u \in \Sigma^* | u \in L_1 \text{ ou } u \in L_2\}$
- Exercice 2. Montrer que $\forall L_1, L_2, L_3$ des langages de Σ^* :
  - $(L_1 + L_2 \subseteq L_3) \rightarrow (L_1 \subseteq L_3 \text{ et } L_2 \subseteq L_3)$

Exercice 3 : Démontrer les propriétés de la somme

Règle	Réunion
Neutralité de $\emptyset$	$\emptyset + L = L + \emptyset = L$
Associativité	$L_1 + (L_2 + L_3) = (L_1 + L_2) + L_3$
Commutativité	$L_1 + L_2 = L_2 + L_1$
Idempotence	$L + L = L$
Croissance	$(L_1 \subseteq L_2) \Rightarrow (L + L_1 \subseteq L + L_2)$

Concaténation des langages

- La concaténation de deux langages L_1 et L_2 de Σ^* est définie par :
  - $L_1.L_2 = \{u \in \Sigma^* | \exists v \in L_1$ et $w \in L_2$ tels que $u = v.w\}$
- Exercice 4. Montrer que $\forall L_1, L_2$ des langages de Σ^* :
  - $|L_1.L_2| \leq |L_1|.|L_2|$

Exercice 5 : Démontrer les propriétés de la concaténation

Règle	Réunion
Neutralité	$\epsilon.L = L.\epsilon = L$
Associativité	$L_1.(L_2.L_3) = (L_1.L_2).L_3$
Croissance	$(L_1 \subseteq L_2) \Rightarrow (L.L_1 \subseteq L.L_2$ et $L_1.L \subseteq L_2.L)$
Nullité	$L.\emptyset = \emptyset.L = \emptyset$
Distributivité	$L_1.(L_2 + L_3) = L_1.L_2 + L_1.L_3$ et $(L_1 + L_2).L_3 = L_1.L_3 + L_2.L_3$

Puissance de langages

- La puissance $L^i \subseteq \Sigma^*$ de L est définie $\forall i \in \mathbb{N}$ par la récurrence :
  - $L_0 = \epsilon$
  - $L^{i+1} = L^i L, \forall i$

Exercice 6 : Démontrer les propriétés de la puissance

1. $L^1 = L$
2. $L^i L^j = L^{i+j}$
3. $L^i L = L L^i = L^{i+1}$
4. $(\epsilon + L)^k = \sum_{i=0}^{k} L^i$

Itération et quotient de langages

- Le langage itéré de L est défini par : $L^* = \sum_{i \geq 0} L^i$
- Le quotient de L_1 par L_2 est : $L_2^{-1}.L_1 = \{v \in \Sigma^* | \exists u \in L_2, u.v \in L_1\}$

Exercice 7 : Démontrer les propriétés de de l'itération.

1. Stabilité par concaténation : Si $L_1 \subseteq L^*$ et $L_2 \subseteq L^*$ alors $L_1 L_2 \subseteq L^*$.
2. Stabilité par itération : Si $L_1 \subseteq L^*$ alors $L_1^* \subseteq L^*$
3. Croissance : Si $L_1 \subseteq L$ alors $L_1^* \subseteq L^*$
4. $\emptyset^* = \epsilon$ et $\epsilon^* = \epsilon$
5. $L^* L = L L^*$
6. $L^* L^* = (L^*)^* = L^*$
7. $L^* = \epsilon + L^* L = \epsilon + L(L^*)$
8. $(L_1 + L_2)^* = (L_1^* + L_2^*)^* = (L_1^*.L_2^*)^* = L_1^*(L_1.L_2^*)^* = (L_1^* L_2)^*.L_1^*$

Généralités

- L'écriture des équations algébriques sur Σ^* est possible avec :
  - La concaténation (notation multiplicative) et
  - la réunion (notation additive).
- Étapes de la section :
  - équation linéaire à une inconnue,
  - systèmes de n équations linéaires à n inconnues.

Exemple de systèmes :

$$(1)\begin{cases} bX_0 + aX_1 = X_0 \\ aX_2 + bX_3 = X_1 \\ aX_1 + bX_3 + \epsilon = X_2 \\ bX_1 + aX_3 = X_3 \end{cases} \qquad (2)\begin{cases} bX_0 + aX_1 + \epsilon = X_0 \\ bX_1 + aX_0 = X_1 \end{cases}$$

Equations linéaires à une inconnue

Lemme d'Arden

Soient $A \subsetneq \Sigma^*$ et $B \subsetneq \Sigma^*$ deux langages formels. Nous appellerons solution de l'équation :

$$X = AX + B \qquad (E)$$

Tout langage $L \subseteq \Sigma^*$ vérifiant la relation $L = AL + B$.

Résolution de l'équation

1. $L = A^*B$ est une solution de (E).
2. L est la plus petite solution de (E), c'est-à-dire que si $M \subseteq \Sigma^*$ vérifie $M = AM + B$, alors $A^*B \subseteq M$.
3. Si $\epsilon \notin A$, alors (E) admet une solution unique

Exercice 8 : Démontrer chacun des trois points.

Inéquations linéaires à une inconnue

Soient $A \subseteq \Sigma^*$ et $B \subseteq \Sigma^*$ deux langages formels. Nous appellerons solution de l'inéquation :

$$AX + B \subseteq X \qquad (I)$$

Tout langage $L \subseteq \Sigma^*$ vérifiant la relation $AL + B \subseteq L$.

Exercice 9 : Résolution de l'inéquation

- Montrer que $L = A^*B$ est la plus petite solution de (I).
- Montrer que $M = A^*(B + C)$ est une solution de (I) $\forall C \subseteq \Sigma^*$
- Que se passe-t-il si le coefficient de X dans (E) contient ϵ ?

Système d'équations linéaires

Soit m un entier naturel et soient $A_{i,j} \subseteq \Sigma^*$ et $B_i \subseteq \Sigma^*$ pour $1 \leq i, j \leq m$.

On appelle solution du système :

$X_i = \sum_{j=1}^{m} A_{i,j} X_j + B_i,\ 1 \leq i \leq m$ (SE),

Tout m-uplet $L = \begin{pmatrix} L_1 \\ L_2 \\ \dots \\ L_m \end{pmatrix}$ avec $L_i \subseteq \Sigma^*$, satisfaisant

$L_i = \sum_{j=1}^{m} A_{i,j} L_j + B_i$.

Sous-forme matricielle, (SE) s'écrit : $X = AX + B$

où $X = \begin{pmatrix} X_1 \\ X_2 \\ \dots \\ X_m \end{pmatrix}$, $A = \begin{pmatrix} A_{1,1} & \dots & A_{1,m} \\ A_{2,1} & \dots & A_{2,m} \\ \dots & \dots & \dots \\ A_{m,1} & \dots & A_{m,m} \end{pmatrix}$, $B = \begin{pmatrix} B_1 \\ B_2 \\ \dots \\ B_m \end{pmatrix}$

Système d'équations linéaires

Résolution du système d'équations

- Sous forme matricielle, la plus petite solution est $A^* B$.
  - Mais le calcul de A^* n'est pas évident.
- (SE) admet une plus petite solution.
- Si $\epsilon \notin A_{i,j} \ \forall i, j \in [1, m]$ alors (SE) a une solution est unique.

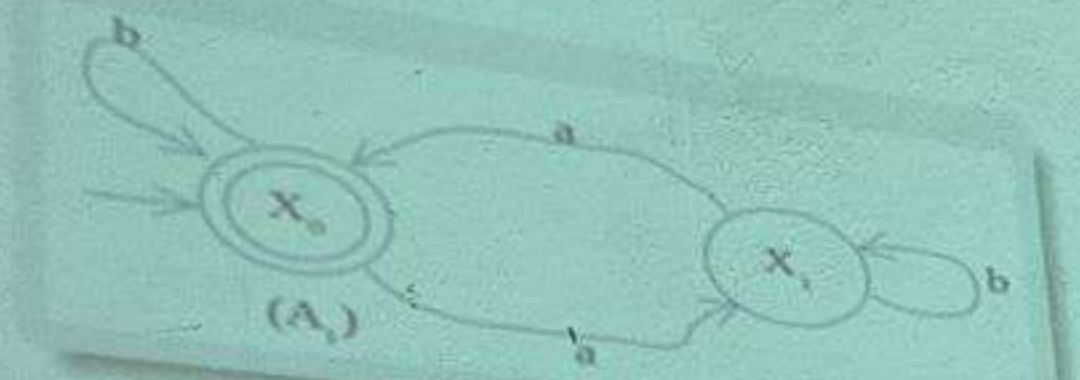
Méthode de GAUSS

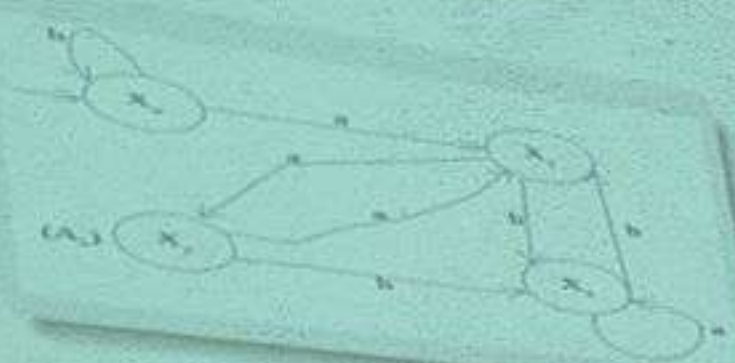
- Mettre l'équation de X_i sous la forme $X_i = A_{l,i} X_i + C_i$ $(X_i = A^*_{i,i} C_i)$
  - C'est la résolvante partielle de l'équation de X_i
- Remplacer dans (SE) l'équation de X_i par sa résolvante partielle
- Remplacer X_i par $\sum_{i=1}^{m} A_{i,j} + B_i$ de son équation dans

Exemples

$$(F)\begin{cases} bX_0 + aX_1 & = X_0 \\ aX_2 + bX_3 & = X_1 \\ aX_1 + bX_3 + \epsilon & = X_2 \\ bX_1 + aX_3 & = X_3 \end{cases} \qquad (G)\begin{cases} bX_0 + aX_1 + \epsilon & = X_0 \\ bX_1 + aX_0 & = X_1 \end{cases}$$

Exercice 10 : 1)- Résoudre le système (F). 2)- Montrer que (G) est engendré par l'automate A_1. 3)-Quel est le langage généré par A_2





Définition

Un monoïde est un triplet $(D, \times, e)$ où

- D est un ensemble, (analogue de Σ^*)
- $\times : D \times D \to D$ est une application, (analogue de .)
- $e \in D$ est un élément particulier de D ; (analogue de ϵ)

Un monoïde vérifie les propriétés suivantes :

- Neutralité : $x \times e = x$ et $e \times x = x$ quel que soit $x \in D$,
- Associativité : $(x \times y) \times z = x \times (y \times z)$ quels que soient $x \in D$, $y \in D$ et $z \in D$

Exemples de monoïdes

Exercice 11 : Montrer que les triplets suivants sont des monoïdes.

- $(\mathbb{N}, +, 0)$ et $(\mathbb{N}, \times, 1)$ où $\mathbb{N}$ est l'ensemble des entiers naturels.
- Soit $L \subseteq \Sigma?$ est un langage sur un alphabet Σ.
  - $(L^*, ., \epsilon)$, en particulier $(\Sigma^*, ., \epsilon)$
  - $(P(L^*), +, \emptyset)$, en particulier $(P(\Sigma^*), +, \emptyset)$
  - $(P(L^*), ., \epsilon)$, en particulier $(P(\Sigma^*), ., \epsilon)$

Définition

Soient $(D, \times, e)$ et $(D', \times', e')$ deux monoïdes. Un morphisme $(D, \times, e) \to (D', \times', e')$ est une application $h : D \to D'$ qui vérifie les propriétés suivantes :

- $h(e) = e'$
- $h(x \times y) = h(x) \times' h(y)$ quels que soient $x \in D$ et $y \in D$.

Exercice 12

- Montrer que : " longueur" , $u \to |u|$ définit un morphisme de $(\Sigma^*, ., \epsilon) \to (\mathbb{N}, +, 0)$.
- Soit $(D, \times, e)$, montrer que toute application $f : \Sigma \to D$ s'étend de façon unique en un morphisme de monoïdes $(\Sigma^*, ., \epsilon) \to (\mathbb{N}, +, 0)$