Université de Nancy — Faculté des Sciences

Institut Universitaire de Calcul Automatique
42, avenue de la Libération
54 - NANCY

NOTIONS SUR LA THEORIE DES LANGAGES

Cours de Monsieur PAIR

Rédigé par A. QUERE

NOTIONS SUR LA THEORIE DES LANGAGES

CHAPITRE 1 : GENERALITES

1.1. Notions de monoïde

1.1.1. Définition : Monoïde

> On appelle monoïde un ensemble M muni d'une loi de composition interne associative et possédant un élément neutre.

Exemples :

- $\mathbb{N}$ muni de l'addition est un monoïde.
- soit E un ensemble ; l'ensemble $\mathcal{P}(E)$ des parties de E muni de la loi $\cup$ est un monoïde (l'élément neutre est $\emptyset$) ; l'ensemble $\mathcal{C}(E)$ des applications surjectives de E dans E est un monoïde pour la composition des applications.

Remarques : certains auteurs n'exigent pas la présence d'un élément neutre dans la définition d'un monoïde ; en réalité, on peut toujours adjoindre à un tel ensemble M un nouvel élément e et prolonger la loi de composition de M en une loi associative dans $M \cup \{e\}$ dont e est élément neutre.

1.1.2. Définition : sous-monoïde

> Soit M un monoïde. Une partie M' de M est un sous-monoïde de M si elle est stable par la loi de M.

M' est alors un monoïde pour la restriction de la loi de M.

1.1.3. Opérations sur les parties d'un monoïde

Soit M un monoïde ; sur $\mathcal{P}(M)$ définissons, en plus des opérations booléennes $\cap$, $\cup$, $\complement$, $\setminus$ ($A \setminus B = A \cap \complement B$), ..., les opérations produit et itération :

Définition :

Soit M un monoïde d'élément neutre e.

Soient $A \subset M$, $B \subset M$; le produit de A par B est l'ensemble :

$$AB = \{ xy \mid x \in A, y \in B \} \ ;$$

en outre $A^* = \bigcup_{i=0}^{+\infty} A^i$ (où $A^0 = \{e\}$ et $A^i = A^{i-1} A$ pour $i \geqslant 1$).

A^* est appelé itéré de A.

Remarquons que $\mathcal{P}(M)$ muni du produit est un monoïde d'élément neutre $\{e\}$.

Propriétés :

Soient A, B, C des parties d'un monoïde M,

$$A(B \cup C) = AB \cup AC$$

$$(B \cup C)A = BA \cup CA$$

Démontrons la première égalité :

$x \in A(B \cup C) \iff (\exists\, a,b)\ (x = ab \text{ et } a \in A \text{ et } b \in B \cup C) \iff$

$((\exists\, a,b)\ (x = ab \text{ et } a \in A \text{ et } b \in B))$ ou $((\exists\, a,b)\ (x = ab \text{ et } a \in A \text{ et } b \in C))$

$\iff x \in AB \cup AC.$

Attention, en général $A(B \cap C) \neq AB \cap AC$

$A(B \setminus C) \neq AB \setminus AC.$

1. 1. 4. Homomorphisme de monoïde

Définition :

Un homomorphisme du monoïde M d'élément neutre e, dans le monoïde M' d'élément neutre e' est une application h de M dans M' telle que :

$$(\forall x, y \in M) \begin{cases} h(x,y) = h(x)\, h(y) \\ h(e) = e' . \end{cases}$$

Si h est une application surjective de M dans M' qui vérifie $(\forall x, y \in M)$ $h(xy) = h(x)\, h(y)$, alors h est un homomorphisme.

En outre, si h est un homomorphisme, h(M) est un sous monoïde de M

1. 1. 5. Monoïde libre sur un ensemble V

Proposition et définition

Soit un ensemble V et L(V) l'ensemble des suites finies d'éléments de V (y compris la suite vide notée Λ) : $L(V) = \{ a_1 a_2 \dots a_k \mid a_i \in V \text{ pour } 1 \leqslant i \leqslant k \} \cup \{\Lambda\}$. Munissons L(V) de la loi de composition suivante appelée concaténation : $(a_1 a_2 \dots a_p , b_1 b_2 \dots b_q) \longrightarrow a_1 a_2 \dots a_p b_1 b_2 \dots b_q$. L(V) est un monoïde, appelé monoïde libre sur V.

Il est évident, en effet, que la loi de la concaténation est associative et admet Λ comme élément neutre.

Définitions

- les suites d'éléments de V sont appelées mots sur V
- soit un mot $\alpha = a_1 \dots a_k$: k est la longueur de α, notée $|\alpha|$. Le mot vide a pour longueur 0. Si $a_i = b$, i est une occurrence de b dans α. Le mot $a_k \dots a_2 a_1$ est appelé réfléchi de α et noté $\tilde{\alpha}$.
- soient trois mots α, β, γ tels que $\alpha = \beta \gamma$. On dit que β est facteur gauche et γ facteur droit de α .

Propriétés immédiates :

1°) Identifions $a \in V$ et le mot $a \in L(V)$; ainsi tout mot sur V est produit d'un nombre fini d'éléments de V et réciproquement ; par suite, en considérant V comme une partie du monoïde L(V) : $L(V) = \bigcup_{i=0}^{\infty} V^i = V^*$. Nous adopterons désormais cette notation.

2°) L'application qui, à un mot, associe sa longueur est un homomorphisme de V^* dans $\mathbb{N}$ muni de l'addition.

3°) Pour deux mots α et β : $\widetilde{\alpha\beta} = \tilde{\beta}\tilde{\alpha}$, $\tilde{\tilde{\alpha}} = \alpha$.

4°) Tout élément de V^* est régulier ; c'est-à-dire :

$$(\forall \alpha, \beta, \gamma \in V^*) \quad (\alpha\beta = \alpha\gamma \implies \beta = \gamma)$$
$$(\beta\alpha = \gamma\alpha \implies \beta = \gamma)$$

ainsi V^* est un semi-groupe.

Remarque :

Soit $\alpha, \beta \in V^*$ tels que $\alpha = \beta$; si i est une occurrence de b dans α, i est aussi une occurrence de b dans β, autrement dit il n'existe pas de "relations" dans le monoïde V^* (mais dans le monoïde $\mathbb{N}$, par exemple, il est bien connu que $x+y = y+x$!) c'est pourquoi V^* est appelé monoïde libre. Cette absence de "relations" dans V^* est essentielle pour le théorème qui suit .

1.1.6. Théorème

Soient un ensemble V, un monoïde M et une application f de V dans M ; il existe un homomorphisme φ et un seul de V^* dans M tel que f soit la restriction de φ à V.

Démonstration :

Soit e l'élément neutre de M. Si un tel homomorphisme φ existe, $\varphi(\Lambda) = e$ et pour tout mot non vide $\alpha = a_1 \dots a_p$

$$\varphi(\alpha) = \varphi(a_1) \dots \varphi(a_p) = f(a_1) \dots f(a_p)$$

donc φ est unique.

Réciproquement, définissons φ par

$$\varphi(\Lambda) = e$$

$$\varphi(\alpha) = f(a_1) \dots f(a_p) \quad \text{pour } \alpha = a_1 \dots a_p$$

et montrons que l'application φ est un homomorphisme.

Soit $\beta \in V^*$, $\beta = b_1 \dots b_q$:

$$\varphi(\alpha\beta) = \varphi(a_1 \dots a_p\, b_1 \dots b_q) = f(a_1) \dots f(a_p)\, f(b_1) \dots f(b_q)$$ par définition

$$= \varphi(\alpha)\, \varphi(\beta)$$ par associativité dans M.

Exemples :

Soient deux ensembles V et V' ; pour définir un homomorphisme φ de V^* dans V'^* il suffit de définir $\varphi(a)$ pour tout $a \in V$.

- si φ est choisi de façon que

  $(\forall a \in V) \quad (|\varphi(a)| \leqslant 1)$

  φ est appelé homomorphisme alphabétique et vérifie :

  $(\forall \alpha \in V^*) \quad (|\varphi(\alpha)| \leqslant |\alpha|)$

  (cette relation se démontre par récurrence sur la longueur de α).

- si, pour tout a de V, $\varphi(a) \in V'$, φ est un homomorphisme alphabétique particulier appelé transcription. Alors, pour tout $\alpha \in V^*$, $|\varphi(\alpha)| = |\alpha|$.

- si φ est injectif, φ est appelé un codage.

1.1.7. Théorème

Tout monoïde est image homomorphe d'un monoïde libre.

En effet, il suffit de choisir V=M et de remplacer f par l'identité dans le théorème précédent.

1. 2. Définition d'un langage

1. 2. 1. Définition

> Soit un ensemble V, un langage de vocabulaire V, ou langage d'alphabet V, ou simplement langage sur V est une partie de V^*.

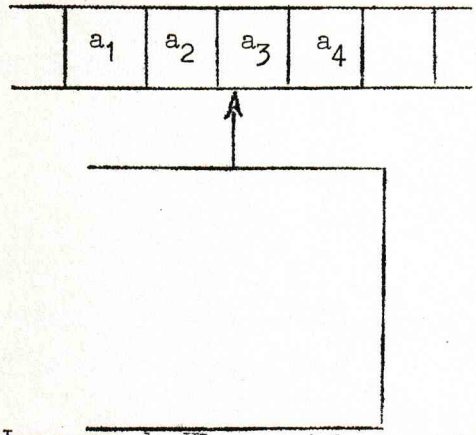
Les opérations union, intersection, passage au complémentaire, produit, itération sont donc applicables aux langages. En outre, si L est un langage, le réfléchi de L, noté $\tilde{L}$, est défini par $\tilde{L} = \{\tilde{\alpha} \mid \alpha \in L\}$.

Par la suite, nous allons étudier quelques types simples de langages. Aux définitions de type axiomatique, nous préférerons les définitions faisant apparaître soit un algorithme de reconnaissance du langage envisagé (point de vue de l'auditeur, - du compilateur), soit un algorithme de construction (point de vue du locuteur, - du programmeur).

Le plus souvent, dans la pratique, le vocabulaire V des langages étudiés est fini. Lorsque, dans la suite, cette hypothèse sera nécessaire, nous la préciserons.

CHAPITRE 2 : LANGAGES DE KLEENE

Pour définir un langage L nous allons utiliser le procédé suivant qui permet de reconnaître et de construire les mots de L : une boîte, susceptible de prendre un certain nombre d'états, peut "lire" successivement chaque lettre d'un mot; à chaque étape la boîte change d'état en tenant compte de son état précédent et de la lettre lue. En outre, parmi les états distinguons des états dits"de satisfaction" et convenons qu'un mot est dans L si et seulement si la boîte achève sa lecture dans un état de satisfaction.



L'objet des paragraphes 1 et 2 est de mettre en forme cette idée de deux manières différentes.

2.1. Langages de Kleene et homomorphismes de mémoire.

2.1.1. Définition : Mémoire sur un ensemble V.

Une mémoire sur un ensemble V est un ensemble S muni d'une loi de composition externe à opérateur dans V notée :

$$S \times V \ni (s, a) \longrightarrow s \cdot a \in S$$

Par exemple V^* est une mémoire sur V en posant :

$$(\forall \alpha \in V^*, a \in V) \qquad \alpha \cdot a = \alpha a$$

Remarquons que, contrairement à l'habitude, la loi externe est notée "à droite"; cette convention facilite les notations, en particulier dans la proposition suivante :

2.1.2. Proposition.

Soit S une mémoire sur V, la loi . peut être prolongée en une loi à opérateur dans V* définie par récurrence sur la longueur de $\alpha \in V^*$:

$$(\forall s \in S) \qquad s \,.\, \Lambda = s$$

$$(\forall s \in S, \alpha \in V^*, a \in V) \qquad s \,.\, \alpha a = (s \,.\, \alpha) \,.\, a$$

De plus :

$$(\forall s \in S, \alpha \in V^*, \beta \in V^*) \qquad (s \,.\, \alpha) \,.\, \beta = s \,.\, (\alpha \beta)$$

Il est évident que la nouvelle loi prolonge l'ancienne. La propriété d'associativité se démontre par récurrence sur la longueur de β.

2.1.3. Définition : homomorphisme de mémoire.

Soient S_1 et S_2 deux mémoires sur V, une application φ de S_1 dans S_2 est un homomorphisme de mémoire si

$$(\forall \alpha \in S_1, a \in V) \qquad \varphi(\alpha \,.\, a) = \varphi(\alpha) \,.\, a$$

2.1.4. Définition d'un langage de Kleene.

Un langage L sur V est appelé langage de Kleene s'il existe une mémoire finie de S sur V, un homomorphisme de mémoire φ de V* dans S et une partie S' de S tels que :

$$L = \varphi^{-1}(S')$$

Il est essentiel que S soit fini; en effet si nous acceptions des mémoires non finies, toute partie L de V* satisferait à (2.1.4) ainsi modifiée (il suffirait de choisir S = V*, S' = L et de prendre, pour φ, l'identité).

Les langages de Kleene sont aussi appelés K-langages ou encore langages d'état fini (finite state langage).

2.1.5. Théorème.

Etant donné une mémoire S sur V et un élément s_0 de S il existe un homomorphisme de mémoire φ de V* dans S et un seul tel que $\varphi(\Lambda) = s_0$.

Démonstration (par récurrence sur la longueur des mots) :

a) $\varphi(\Lambda) = s_0$ définit φ sur le mot vide.

b) Supposons φ défini de manière unique sur les mots de longueur au plus n et vérifiant :

$$(\forall \alpha \in V^*,\ |\alpha| \leq n-1,\ \forall a \in V)\ (\varphi(\alpha a) = \varphi(\alpha) \cdot a) \qquad (1)$$

c) Soit α un mot de longueur $n+1$. α est du type $\alpha' a$ avec $|\alpha'| \leq n$ et nécessairement :

$$\varphi(\alpha) = \varphi(\alpha' a) = \varphi(\alpha') \cdot a \qquad (2)$$

ainsi φ est défini de manière unique pour les mots de longueur au plus $n+1$; (1) et (2) montrent que φ est un homomorphisme de mémoire.

Conséquences :

- Un langage de Kleene est défini par la donnée de V, S, s_0 et S'.
- Soit L un langage de Kleene défini par V, S, s_0, S'; l'application : $V^* \ni \alpha \longrightarrow s_0 \cdot \alpha \in S$ (défini en 2.1.2.)) est un homomorphisme de mémoire tel que

$$s_0 \cdot \Lambda = s_0$$

donc c'est l'homomorphisme appelé φ dans le théorème :

$$s_0 \cdot \alpha = \varphi(\alpha).$$

Par suite

$$(\forall a_1 \ldots a_n \in V^*)\ \varphi(a_1 a_2 \ldots a_n) = (\ldots((s_0 \cdot a_1) \cdot a_2)\ldots) \cdot a_n.$$

Ainsi nous disposons d'un algorithme de reconnaissance pour le langage L.

2.2. Langages de Kleene et automates finis.

2.2.1. Définition d'un automate fini :

Un automate fini A sur V est un quadruplet comprenant :

- un ensemble fini S non vide (appelé ensemble des états),
- une loi de composition externe sur S à opérateur dans V notée . ,
- un élément $s_0 \in S$ appelé état initial,
- une partie S' de S appelée ensemble final.

Nous noterons $A = (S, ., s_0, S')$

2.2.2. Définition d'un calcul.

Soit $A = (S, \cdot ; s_0, S')$ un automate fini. Un calcul de A est une suite finie d'éléments de $S \times V$: $(s_1, a_1) \ldots (s_n, a_n)$ telle que :

$s_i = s_{i-1} \cdot a_i \qquad 1 \leq i \leq n.$

La donnée du calcul est le mot $a_1 a_2 \ldots a_n$; s_n est le dernier état du calcul. Le calcul de donnée Λ est appelé calcul vide; par convention, son dernier état est s_0.

2.2.3. Proposition.

Soit $A = (S, ., s_0, S')$ un automate fini sur V. Pour tout mot α de V* il existe un calcul et un seul de donnée α ; son dernier état est $\varphi(\alpha)$, où φ est l'homomorphisme de mémoire de V* dans S défini par $\varphi(\Lambda) = s_0$.

La démonstration se fait par récurrence sur la longueur de α :

Si $\alpha = \Lambda$, s_0 est le dernier état et la proposition est trivialement vérifiée.

Supposons la propriété vraie pour tout mot de longueur n et soit α un mot de longueur n + 1; α se décompose en α' a ($a \in V$, $\alpha' \in V^*$).

D'après la définition (2.2.2.) un calcul de donnée α est le calcul de donnée α' (unique par hypothèse de récurrence) de dernier état s', suivi d'un couple (s , a) tel que $s = s' \cdot a$, donc s est unique. En définitive le calcul de donnée α est unique. Calculons $\varphi(\alpha)$: $\varphi(\alpha) = \varphi(\alpha' a) = \varphi(\alpha') \cdot a = s' \cdot a$ (hypothèse de récurrence)

$= s$,

ce qui achève la démonstration.

2.2.4. Définition.

Soit $A = (S, ., s_0, S')$ un automate fini sur V. Un mot α est dit accepté par A si le dernier état du calcul de donnée α appartient à S'. L'ensemble des mots acceptés par A est le langage accepté par A.

2.2.5. Théorème.

Un langage sur V est un langage de Kleene si et seulement s'il existe un automate fini qui l'accepte.

En effet si la langage L est accepté par l'automate $A = (S, ., s_0, S')$, l'homomorphisme φ associé à A grâce à (2.2.3.) est tel que

$$\varphi^{-1}(S') = L$$

donc L est un langage de Kleene. Réciproquement soit L un langage de Kleene associé à la mémoire S, la loi de composition externe notée . , l'homomorphisme φ et la partie S' de S. Posons $A = (S, ., \varphi(\lambda), S')$. D'après (2.2.3.) et la définition (2.2.4.) :

$$\varphi(\alpha) \in S' \iff (\alpha \text{ est accepté par A})$$

donc L est le langage accepté par A.

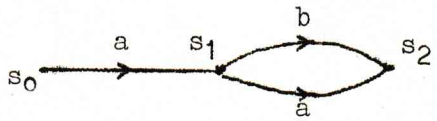
2.3. Exemples.

① V^* est un K langage; en effet il suffit de choisir la mémoire $S = \{s_0\}$ munie du produit $s_0 . a = s_0$ pour tout a de V, donc :

$$(\forall \alpha \in V^*) \quad \varphi(\alpha) = s_0$$
$$\text{et} \quad V^* = \varphi^{-1}(\{s_0\})$$

② $V = \{a, b\}$ et $L = \{aa, ab\}$. L est-il un K-langage ?

Pour tenter de construire un automate acceptant L précisons les transitions possibles sur un schéma : si l'automate lit a dans l'état s_0 il passe dans l'état s_1; alors, qu'il lise a ou b, il prend l'état s_2 qui sera le seul état de satisfaction.



Les autres cas qui peuvent se présenter correspondent à des mots qui ne sont pas dans L. Ajoutons donc à s_0, s_1, s_2 un état 0 (état de "refus") et posons $S = \{s_0, s_1, s_2, 0\}$. La loi externe sur S, notée ., est définie par la table :

	a	b
s_0	s_1	0
s_1	s_2	s_2
s_2	0	0
0	0	0

Soit l'automate $(S, ., s_o, \{s_2\})$ et φ l'homomorphisme de mémoire associé à cet automate. Montrons que :

$$\varphi^{-1}(\{s_2\}) = L$$

en effet :

$$\begin{aligned}\varphi(aa) &= \varphi(a) \cdot a \\ &= (\varphi(\Lambda) \cdot a) \cdot a \\ &= (s_o \cdot a) \cdot a \\ &= s_2\end{aligned}$$

et, de même, $\varphi(ab) = s_2$

Par suite tout mot de L est dans $\varphi^{-1}(\{s_2\})$. Il reste à montrer que si α n'est pas dans L, $\varphi(\alpha)$ n'est pas s_2 :

soit $\alpha \notin L$; distinguons plusieurs cas :

- a n'est pas facteur gauche de α :

ou $\alpha = \Lambda$ et $\varphi(\alpha) = s_o$

ou il existe α' dans V* tel que $\alpha = b\alpha'$

$$\begin{aligned}\varphi(\alpha) &= s_o \cdot \alpha \\ &= (s_o \cdot b) \cdot \alpha' \\ &= 0 \cdot \alpha' \\ &= 0\end{aligned}$$

- a est facteur gauche de α ;

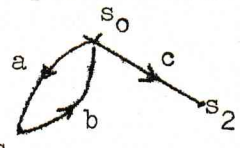
ou $\alpha = a$ et $\varphi(\alpha) = s_1$

ou $\alpha = a\,x\,y\,\beta$ où x et y sont dans V et β dans V*.

$$\begin{aligned}\varphi(\alpha) = s_o \cdot \alpha &= (s_o \cdot a) \cdot (x\,y\,\beta) \\ &= (s_1 \cdot x) \cdot (y\beta) \\ &= (s_2 \cdot y) \cdot \beta \\ &= 0 \cdot \beta \\ &= 0\end{aligned}$$

(3) $V = \{a, b, c\}$ $L = \{(ab)^n c \mid n \geq 0\}$ (où n représente une puissance de concaténation) c'est-à-dire $L = \{ab\}^* \{c\}$

Construisons un automate acceptant L. Nous adopterons le schéma suivant :



Par suite $S = \{s_o, s_1, s_2, 0\}$ la loi externe notée . est donnée par le table

	a	b	c
s_o	s_1	0	s_2
s_1	0	s_o	0
s_2	0	0	0
0	0	0	0

Montrons que l'automate fini $(S, ., s_o, \{s_2\})$ accepte L. Appelons φ l'homomorphisme associé à cet automate. Etablissons, par récurrence sur n, que $\varphi((ab)^n c) = s_2$. La relation est triviale pour n = 0; supposons la établie à l'ordre n-1.

$$\begin{aligned}
\varphi((ab)^n c) &= s_o \cdot [(ab)^n c] \\
&= (s_o \cdot ab) \cdot [(ab)^{n-1} \cdot c] \\
&= s_o \cdot [(ab)^{n-1} c] \qquad \text{(d'après la table)} \\
&= \varphi((ab)^{n-1} c) \\
&= s_2 \qquad \text{(hypothèse de récurrence).}
\end{aligned}$$

Réciproquement soit α dans V* tel que $\varphi(\alpha) = s_2$. D'après la table si s_2 est le dernier état s_o est nécessairement l'avant dernier et il existe α' dans V* tel que

$$\alpha = \alpha' c$$
$$\varphi(\alpha') = s_o$$

Montrons enfin, par récurrence sur la longueur de α', que si $\varphi(\alpha') = s_o$, il existe un entier n tel que

$$\alpha' = (ab)^n$$

ce qui achèvera la démonstration.

Pour $\alpha' = \lambda$ la propriété est triviale. Supposons la propriété vraie pour tout α' de longueur inférieure à p, et soit α' de longueur $p + 1$; $\alpha' = \beta x$ où β est dans V* et x dans V,

$$s_o = \varphi(\alpha') = \varphi(\beta) \cdot x$$

D'après la table $\varphi(\beta) = s_1$, $x = b$ et β admet a comme facteur droit :

$$(\exists\, \gamma \in V^*) \qquad (\beta = \gamma\, a)$$

$$s_1 = \varphi(\gamma\, a) = \varphi(\gamma) \cdot a$$

d'où

$$\varphi(\gamma) = s_o$$

Par hypothèse de récurrence :

$$(\exists\, n \in \mathbb{N}) \qquad (\gamma = (ab)^n)$$

en définitive $\alpha' = (ab)^{n+1}$

2.4. Langages locaux sur V.

2.4.1. Définition

Soient deux sous-ensembles D et F de V et une partie τ de V^2.

Considérons les mots non vides sur V, $a_1\ a_2 \cdots a_n$, qui possèdent les propriétés suivantes :

1) $a_1 \in D$
2) $a_n \in F$
3) Pour i tel que $2 \leqslant i \leqslant n$: $(a_{i-1}, a_i) \in \tau$.

Leur ensemble L s'appelle un langage local.

2.4.2. Théorème

Un langage local sur un vocabulaire fini est un langage de Kleene.

Démonstration : Prenons l'automate défini par le quadruplet : $(S, ., s_o, S')$, avec :

- $S = V \cup \{ 0, s_o \}$; s_o étant l'état initial.
- La loi externe notée : $\ll . \gg$ et définie par :

  $a . b$ = SI $(a, b) \in \tau$ ALORS b SINON 0.

  $s_o . b$ = SI $b \in D$ ALORS b SINON 0.

  $0 . b = 0$

- $F = S'$.

Soient φ l'homomorphisme de mémoire de V* dans S associé à l'automate et $\alpha = a_1\ a_2 \cdots a_n$ dans V*. Montrons que :

(1) Si $a_1 \in D$ et $(a_{i-1}, a_i) \in \tau$ pour i = 2, 3, ..., n alors $\varphi(\alpha) = a_n$

(2) Sinon $\varphi(\alpha) = 0$.

Il en résultera alors immédiatement que

$$\alpha \in L \Longleftrightarrow \varphi(\alpha) \in S'.$$

Faisons une récurrence sur la longueur de α :

1) $|\alpha| = 1$, $\alpha = a_1$.

$\varphi(\alpha) = \varphi(a_1) = s_o . a_1$

$= $ SI $a_1 \in D$ ALORS a_1 SINON 0.

2) $|\alpha| = n > 1$:

$\varphi(\alpha) = \varphi(a_1\ a_2 \cdots a_{n-1}) . a_n$

- Si a_1 appartient à D et (a_{i-1}, a_i) appartient à τ pour tout i = 2, 3 ... n alors $\varphi(\alpha) = a_{n-1} \cdot a_n$; mais (a_{n-1}, a_n) appartient à τ donc $a_{n-1} \cdot a_n = a_n$ et (1) est vérifiée.
- Si a_1 n'appartient pas à D, $\varphi(a_1 \ldots a_{n-1}) = 0$, donc $\varphi(\alpha) = 0 \cdot a_n = 0$ et (2) est vérifiée.
- Si (a_{i-1}, a_i) n'appartient pas à τ pour un i inférieur à n, $\varphi(\alpha) = 0 \cdot a_n = 0$ et (2) est vérifiée.
- Si n est le premier i tel que (a_{i-1}, a_i) n'appartient pas à τ alors $a_{n-1} \cdot a_n = 0$, $\varphi(\alpha) = a_{n-1} \cdot a_n = 0$ et (2) est vérifiée.

Exemple de langage local : $\{(ab)^n \mid n \in \mathbb{N}\}$, avec $D = \{a, c\}$, $F = \{c\}$ et $\tau = \{(a, b), (b, a), (b, c)\}$.

2.4.3. Théorème.

Les calculs non vides d'un automate fini, dont le dernier état appartient à l'ensemble final de l'automate, forment un langage local, dont le vocabulaire est fini si il en est ainsi pour celui de l'automate.

Soit (S, ., s_o, S') l'automate (S fini). Les calculs considérés sont les suites finies $[(s_i, a_i)]_{1 \leq i \leq m}$ non vides avec : $a_i \in V$, $s_i \in S$, $s_i = s_{i-1} \cdot a_i$ pour i = 1, 2, ..., n, et $s_n \in S'$.

Le vocabulaire du langage local que nous définissons est S x V; quant aux ensembles τ, D, F ils sont définis par :

$\tau = \{((s, a), (s', a')) \mid s' = s \cdot a'\} = \{((s, a), (s \cdot a', a'))\}$.

$D = \{(s_o \cdot a, a)\}$

$F = S' \times V$

Remarque : La précision "non vide" est indispensable car les langages locaux, par définition, ne contiennent pas le mot vide.

2.4.4. Conséquence.

Tout langage de Kleene est l'ensemble des données des calculs d'un automate fini (S, ., s_o, S') dont le dernier état appartient à S'; mais nous pouvons passer d'un calcul non vide à sa donnée par la transcription (paragraphe 1.1.6) t de $(S \times V)^*$ dans V^* définie par

$$t((s, a)) = a.$$

On en déduit le théorème suivant :

Théorème.

Tout langage de Kleene ne contenant pas le mot vide, est déduit d'un langage local par une transcription.

Nous démontrerons plus tard (2.8.5.) que la réciproque est exacte lorsque le K-langage a un vocabulaire fini.

D'où la nouvelle caractérisation :

Les langages de Kleene, sur un vocabulaire fini, ne contenant pas le mot vide, sont les transformés des langages locaux par transcription.

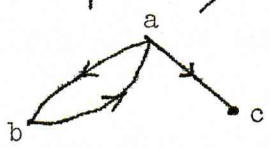
2.4.5. Représentation d'un langage local par un graphe.

Soit un langage local défini par les ensembles D, F, τ comme plus haut. Envisageons le graphe (V, τ) où la relation binaire τ est définie par :

$$a \; \tau \; b \iff (a, b) \in \tau .$$

Appelons chemin du graphe une suite finie et non vide de points $(x_1 \; x_2 \; \dots \; x_n)$ tels que $x_{i-1} \; \tau \; x_i$ pour $2 \leq i \leq n$; le langage local est l'ensemble des chemins du graphe dont l'origine appartient à D et l'extrémité appartient à F.

Exemple : le langage $\{(ab)^n \, c \mid n \in \mathbb{N}\}$ est associé au graphe suivant :



avec $D = \{a, c\}$ et $F = \{c\}$.

D'après les théorèmes précédents il s'en suit que tout langage de Kleene sur un vocabulaire fini, ne contenant pas le mot vide, est déduit de l'ensemble des chemins d'un graphe fini (E, T) dont l'origine appartient à une partie D de E et l'extrémité à une partie F de E, par une transcription.

Remarque : la réciproque est vraie.

2.5. Catégories grammaticales.

Les caractérisations des langages de Kleene que nous avons étudiées permettent de construire facilement des K-langages; par contre elles sont peu efficaces lorsqu'il s'agit de démontrer si un langage donné est, ou non, un langage de Kleene. Pour remédier à cet inconvénient nous allons étudier la syntaxe des langages en classant les mots en différentes "catégories grammaticales".

2.5.1. Définitions : contextes d'un mot, classe des contextes d'un mot.

Soient un ensemble V, un langage L sur V et un mot α de V*. Un contexte du mot α pour L est un couple (β, γ) de $(V^*)^2$ tel que $\beta\alpha\gamma$ soit un mot de L. La classe des contextes de α pour L, notée $C_L(\alpha)$, est l'ensemble des contextes de α pour L.

N.B. Lorsqu'il n'y a pas d'ambiguïté possible $C_L(\alpha)$ est notée $C(\alpha)$.

2.5.2. Définition et proposition.

Soit un langage L sur V. La relation définie sur V* par

$$\alpha \sim \alpha' \iff C(\alpha) = C(\alpha')$$

est une relation d'équivalence. Les classes d'équivalence sont appelées catégories grammaticales.

2.5.3. Propriétés.

Soit L un langage sur V.

a) Les catégories grammaticales forment une partition de V*

b) $(\forall \alpha \in V^*) \quad (\alpha \in L \iff (\Lambda, \Lambda) \in C(\alpha))$

c) L est réunion de catégories grammaticales

d) L'équivalence $\sim$ est une congruence; par suite l'ensemble quotient peut être muni, canoniquement, d'une structure de monoïde.

Démonstration. L'assertion b) est évidente. Le a) est vérifié par toute relation d'équivalence. Soient α et α' deux mots équivalents :

$$\alpha \in L \iff (\Lambda, \Lambda) \in C(\alpha) \iff (\Lambda, \Lambda) \in C(\alpha') \iff \alpha' \in L$$

ce qui prouve c).

Enfin :

$$(\forall \alpha, \alpha', \beta \in V^*) \qquad C(\alpha\beta) = \{(\gamma_1, \gamma_2) \in (V^*)^2 \mid \gamma_1\alpha\beta\gamma_2 \in L\}$$
$$= \{(\gamma_1, \gamma_2) \in (V^*)^2 \mid (\gamma_1, \beta\gamma_2) \in C(\alpha)\}.$$

Par suite :

$$\alpha \sim \alpha' \Leftrightarrow C(\alpha) = C(\alpha') \Rightarrow C(\alpha\beta) = C(\alpha'\beta) \Leftrightarrow \alpha\beta \sim \alpha'\beta.$$

De même

$$\alpha \sim \alpha' \Rightarrow \beta\alpha \sim \beta\alpha'$$

Donc la relation $\sim$ est une congruence et si $\psi(\alpha)$ est la catégorie grammaticale de α le produit défini sur l'ensemble quotient par

$$(\forall \alpha, \beta \in V^*) \quad \psi(\alpha) \times \psi(\beta) = \psi(\alpha\beta)$$

en fait un monoïde. En outre, ψ est alors un homomorphisme de monoïde.

2.5.4. Définition : langage régulier.

Un langage L est dit régulier si l'ensemble des catégories grammaticales pour L est fini, ou si l'ensemble des classes des contextes pour L est fini.

(En effet, d'après les définitions, l'ensemble des catégories grammaticales et celui des classes de contextes sont équipotents).

2.5.5. Théorème.

Un langage L est régulier si et seulement s'il existe un monoïde fini M, un homomorphisme de monoïde ψ de V^* dans M et M' une partie de M tels que

$$L = \psi^{-1}(M').$$

Démonstration. Soit L un langage régulier; il suffit de choisir comme monoïde M l'ensemble des catégories grammaticales, comme homomorphisme ψ, l'application canonique de V^* dans M et pour M' l'ensemble des classes dont L est réunion.

Réciproquement, soit un langage L tel qu'il existe un monoïde fini M, un homomorphisme de monoïde ψ de V^* dans M et une partie M' de M tels que

$$L = \psi^{-1}(M')$$

$$C(\alpha) = \{(\beta, \gamma) \in V^* \times V^* \mid \beta\alpha\gamma \in L\}$$

$$= \{(\beta, \gamma) \in V^* \times V^* \mid \psi(\beta\alpha\gamma) \in M'\}$$

$$= \{(\beta, \gamma) \in V^* \times V^* \mid \psi(\beta)\psi(\alpha)\psi(\gamma) \in M'\}$$

$(V^*)^2$

Par suite :

$$(\forall \alpha, \alpha' \in V^*) \quad (\psi(\alpha) = \psi(\alpha') \Rightarrow C(\alpha) = C(\alpha'))$$

c'est-à-dire :

$$(\forall \alpha, \alpha' \in V^*) \quad (C(\alpha) \neq C(\alpha') \Rightarrow \psi(\alpha) \neq \psi(\alpha')).$$

Si le nombre des C (α_i) n'était pas fini il en irait de même pour le nombre des $\psi(\alpha)$ ce qui est contraire au fait que M est fini. En définitive L est un langage régulier.

2.5.6. Théorème

Les deux assertions suivantes sont équivalentes :

i) L est un langage régulier sur V.

ii) L est un langage de Kleene sur V.

Démonstration. Supposons L régulier. Soient donc M, M' et ψ associés à L comme nous venons de l'indiquer (2.5.5.).

Munissons $\psi(V^*)$ d'une structure de mémoire sur V en posant :

$(\forall \alpha \in V^*, a \in V) \quad \psi(\alpha) \cdot a = \psi(\alpha a)$

Ainsi ψ est un homomorphisme de mémoire et $L (= \psi^{-1}(M'))$ est un langage de Kleene.

Réciproquement, soit un langage de Kleene L; il existe une mémoire finie S sur V, un homomorphisme de mémoire φ de V^* dans S et une partie S' de S tels que $L = \varphi^{-1}(S')$.

Soit $\mathcal{M}(S)$ l'ensemble des applications de S dans S. Posons :

$(\forall \sigma, \sigma' \in \mathcal{M}(S)) \quad \sigma \times \sigma' = \sigma' \circ \sigma$.

Comme la loi de composition des applications, la loi $\times$ est une loi de monoïde sur $\mathcal{M}(S)$. De plus $\mathcal{M}(S)$ est fini car S l'est.

Si α est un mot de V^* appelons a_α l'application de S dans S défini par

$(\forall s \in S) \quad a_\alpha(s) = s \cdot \alpha$ ($s \cdot \alpha$ a été défini en (2.1.2.)).

Soit alors ψ l'application de V^* dans $\mathcal{M}(S)$ définie par :

$(\forall \alpha \in V^*) \quad \psi(\alpha) = a_\alpha$.

ψ est un homomorphisme de monoïde :

$(\forall \alpha, \beta \in V^*, \forall s \in S) \quad a_{\alpha\beta}(s) = s \cdot \alpha\beta = (s \cdot \alpha) \cdot \beta = a_\beta \circ a_\alpha(s)$

donc

$$\psi(\alpha\beta) = a_{\alpha\beta} = a_\beta \circ a_\alpha = a_\alpha \times a_\beta = \psi(\alpha) \times \psi(\beta).$$

De plus :

$$\alpha \in L \iff \psi(\alpha) \in S'$$
$$\iff s_0 \cdot \alpha \in S' \qquad \text{où } s_0 = \varphi(\Lambda)$$
$$\iff \psi(\alpha)(s_0) \in S'$$

et en posant $M' = \{\sigma \in \mathcal{M}(S) \mid \sigma(s_0) \in S'\}$,

$$L = \psi^{-1}(M').$$

ce qui achève la démonstration.

Exemple d'application : Sur le vocabulaire $V = \{a, b\}$ étudions le langage $L = \{a^n b^n\}$. Soit $a^p b$ dans V^* :

$$C(a^p b) = \{(\alpha, \beta) \in (V^*)^2 \mid \alpha a^p b \beta \in L\}$$
$$= \{(a^r, b^{r-1+p})\}$$

Donc, si $p \neq p'$, $C(a^p b) \neq C(a^{p'} b)$

Par suite le nombre des classes de contextes des mots du type $a^p b$ (p entier naturel) n'est pas fini. A fortiori L n'est pas régulier, c'est-à-dire L n'est pas un langage de Kleene.

Dans ce qui suit nous allons classer les mots de V^* en tenant compte uniquement de leurs contextes à droite; l'étude est assez semblable à celle que nous venons de mener.

2.6. Contextes à droite.

2.6.1. Définitions : contexte à droite pour un langage L, relation d'équivalence associée.

Soient un ensemble V, un langage L sur V et un mot α de V^*. Un contexte à droite de α pour L est un mot β de V^* tel que $\alpha\beta$ soit un mot de L. La classe des contextes à droite de α pour L, notée $D_L(\alpha)$ (ou $D(\alpha)$ lorsqu'il n'y a pas ambiguïté), est l'ensemble des contextes à droite de α. La relation définie sur V^* par

$$\alpha \sim \alpha' \iff D(\alpha) = D(\alpha')$$

est une relation d'équivalence.

2.6.2. Propriétés.

Soit L un langage sur V.

a) $(\forall \alpha \in V^*) \quad (\alpha \in L \iff \Lambda \in D(\alpha))$.

b) L est réunion de classes d'équivalence.

c) L'équivalence $\sim$ est compatible à droite avec la concaténation; par suite l'ensemble quotient peut être muni, canoniquement, d'une structure de mémoire sur V.

Les assertions a) b) sont évidentes.

Soient α, α' et β dans V^* :

$$D(\alpha\beta) = \{\gamma \in V^* \mid \alpha\beta\gamma \in L\}$$
$$= \{\gamma \in V^* \mid \beta\gamma \in D(\alpha)\}.$$

Par suite :

$$\alpha \sim \alpha' \iff D(\alpha) = D(\alpha') \implies D(\alpha\beta) = D(\alpha'\beta) \iff \alpha\beta \sim \alpha'\beta.$$

Ainsi la relation $\sim$ est compatible à droite avec la concaténation (mais pas à gauche en général !). La classe d'un mot α est notée $\varphi(\alpha)$ et l'ensemble quotient S. En posant $(\forall \alpha \in V^*, a \in V) \quad \varphi(\alpha) \cdot a = \varphi(\alpha a)$ nous définissons une loi externe sur S à opérateur dans V qui fait de S une mémoire sur V. En effet :
$(\forall \alpha, \alpha' \in V^*, a \in V) \quad \varphi(\alpha) = \varphi(\alpha') \Leftrightarrow \alpha \sim \alpha' \Rightarrow \alpha a \sim \alpha' a \Leftrightarrow \varphi(\alpha a) = \varphi(\alpha' a)$.
En outre φ est un homomorphisme de mémoire.

2.6.3. Définition : langage régulier à droite.

Un langage L est dit régulier à droite si l'ensemble quotient pour la relation $\sim$ est fini, ou si l'ensemble des $D(\alpha)$, pour α dans V^*, est fini.

2.6.4. Théorème.

Les deux assertions suivantes sont équivalentes :
i) L est un langage régulier à droite sur V.
ii) L est un langage de Kleene sur V.

Démonstration : Supposons L régulier à droite; utilisons les notations précédentes (2.6.2.) : l'ensemble quotient fini S, muni de sa structure de mémoire, l'homomorphisme canonique φ et l'ensemble S' des classes d'équivalence, dont la réunion est L, sont tels que :

$$L = \varphi^{-1}(S')$$

autrement dit L est un langage de Kleene.

Réciproquement, soit un langage de Kleene L :

$$(\forall \alpha \in V^*) \quad D(\alpha) = \{\beta \in V^* \mid (\Lambda, \beta) \in C(\alpha)\}$$

donc $(\forall \alpha, \alpha' \in V^*) \quad D(\alpha) \neq D(\alpha') \Rightarrow C(\alpha) \neq C(\alpha')$

Si le nombre des $D(\alpha)$ n'était pas fini, il en irait de même pour celui des $C(\alpha)$ ce qui est absurde car L est régulier (2.5.6.). Donc L est régulier à droite.

Nous allons étudier aussi les contextes à gauche; il est moins facile de les relier directement aux automates, comme nous venons de le faire pour les contextes à droite (en effet les automates ont un "sens de lecture"), mais nous pourrons nous ramener à l'étude des contextes à droite en considérant des langages réfléchis.

2.7. Contextes à gauche.

Les définitions et propriétés sont tout à fait analogues à celles relatives aux contextes à droite (cf. (2.6.1.), (2.6.2.)). Pour un langage L, la classe des contextes à gauche d'un mot α sera notée $G_L(\alpha)$ (ou $G(\alpha)$) :

$$G(\alpha) = \{\beta \in V^* \mid \beta\alpha \in L\}.$$

2.7.1. Lemme 1

Soit L un langage sur V. Si L est régulier à gauche (resp. à droite), le langage miroir $\tilde{L}$ est régulier à droite (resp. à gauche).

Démonstration :

$(\forall \alpha \in V^*)$ $\quad D_{\tilde{L}}(\alpha) = \{\beta \in V^* \mid \alpha\beta \in \tilde{L}\}$

$= \{\beta \in V^* \mid \tilde{\beta}\tilde{\alpha} \in L\}$

$= G_L(\tilde{\alpha})$

Par suite si L est régulier à gauche le nombre des $G_L(\tilde{\alpha})$ (pour $\alpha \in V^*$) est fini; il en va de même pour celui des $D_{\tilde{L}}(\alpha)$; par conséquent $\tilde{L}$ est régulier à droite.

De même : $(\forall \alpha \in V^*)$ $\quad G_{\tilde{L}}(\alpha) = D_L(\tilde{\alpha})$

et si L est régulier à droite, $\tilde{L}$ est régulier à gauche.

2.7.2. Lemme 2.

Soit L un langage sur V. Si L est un langage de Kleene $\tilde{L}$ est un langage de Kleene.

$(\forall \alpha \in V^*)$ $\quad C_{\tilde{L}}(\alpha) = \{(\beta, \gamma) \in (V^*)^2 \mid \beta\alpha\gamma \in \tilde{L}\}$

$= \{(\beta, \gamma) \in (V^*)^2 \mid \tilde{\gamma}\tilde{\alpha}\tilde{\beta} \in L\}$

$= \{(\beta, \gamma) \in (V^*)^2 \mid (\tilde{\gamma}, \tilde{\beta}) \in C_L(\tilde{\alpha})\}$

Si L est un langage de Kleene les $C_L(\tilde{\alpha})$ (pour α dans V^*) sont en nombre fini; à fortiori les $C_{\tilde{L}}(\alpha)$ le sont aussi et $\tilde{L}$ est un langage de Kleene.

2.7.3. Théorème.

Les deux assertions suivantes sont équivalentes :

i) L est un langage régulier à gauche sur V.

ii) L est un langage de Kleene sur V.

Démonstration : Si L est régulier à gauche $\tilde{L}$ est régulier à droite (2.7.1); donc $\tilde{L}$ est un langage de Kleene (2.6.4); par suite $\tilde{\tilde{L}} = L$ est un langage de Kleene (2.7.2.), et réciproquement.

2.8. Propriétés des langages de Kleene.

Nous étudierons principalement dans ce paragraphe le comportement des langages de Kleene vis-à-vis des opérations ensemblistes (réunion intersection, etc...), vis-à-vis de certaines transformations et vis-à-vis des opérations produit et itération (définies au chapitre 1). Cette étude aboutira à une nouvelle caractérisation des langages de Kleene sur un vocabulaire fini : le théorème de Kleene.

2.8.1. Proposition.

Soit une partie V d'un ensemble V'. Si K est un langage de Kleene sur V, K est aussi un langage de Kleene sur V'.

Pour α dans V^* (resp. dans V'^*) notons $D(\alpha)$ (resp. $D'(\alpha)$) la classe des contextes à droite de α pour le langage K sur V (resp. sur V'. Alors :

$(\forall \alpha \in V^*)$ $D'(\alpha) = \{\beta \in V'^* \mid \alpha\ \beta \in K\}$

mais comme K est un langage sur V les β sont dans V^* et

$(\forall \alpha \in V^*)$ $D'(\alpha) = D(\alpha)$.

De plus : $(\forall \alpha \in V'^* \setminus V^*)$ $D'(\alpha) = \emptyset$

Par suite les ensembles $D'(\alpha)$, comme les ensembles $D(\alpha)$ sont en nombre fini et K est un langage de Kleene sur V'.

2.8.2. Proposition.

Soit un ensemble V :

i) $\{\Lambda\}$ est un langage de Kleene

ii) pour tout a $\in$ V, $\{a\}$ est un langage de Kleene.

iii) V est un langage de Kleene sur V.

En effet, pour le langage $\{\Lambda\}$ il est facile d'exhiber les classes des contextes à droite :

si $\alpha \neq \Lambda$, $D(\alpha) = \emptyset$

et $D(\Lambda) = \Lambda$.

De même pour le langage $\{a\}$:

$D(\Lambda) = \{a\}$

$D(a) = \{\Lambda\}$

et pour $\alpha \in V^*$, $\alpha \neq a$, et $\alpha \neq \Lambda$, $D(\alpha) = \emptyset$.

Enfin cherchons les classes de contextes à droite des mots α de V^* par rapport à V :

si $|\alpha| = 0$ $\quad D_V(\alpha) = V$

si $|\alpha| = 1$ $\quad D_V(\alpha) = \{\Lambda\}$

si $|\alpha| \geqslant 2$ $\quad D_V(\alpha) = \emptyset$.

Nous avons trois classes seulement, donc V est un langage de Kleene sur V.

2.8.3. Proposition.

Soient K et K' deux langages de Kleene sur V.

a) Le complémentaire de K dans V* est un langage de Kleene sur V.

b) $K \cap K'$, $K \cup K'$, $K \setminus K'$ sont des langages de Kleene sur V.

Démonstration :

a) il existe un monoïde fini M, une partie M' de M et un homomorphisme de monoïde ψ tels que :

$$K = \psi^{-1}(M') ;$$

par suite $\complement_{V^*} K = \psi^{-1}\left(\complement_M M'\right)$

ce qui prouve que le complémentaire de K est un langage de Kleene.

b) envisageons les classes de contexte à droite :

$$(\forall \alpha \in V^*)\ D_{K \cap K'}(\alpha) = \{\beta \mid \alpha\beta \in K \text{ et } \alpha\beta \in K'\}$$
$$= D_K(\alpha) \cap D_{K'}(\alpha),$$

ainsi les $D_{K \cap K'}(\alpha)$ sont en nombre fini et $K \cap K'$ est un langage de Kleene. Par conséquent $K \cup K' = \complement_{V^*}\left(\complement_{V^*} K \cap \complement_{V^*} K'\right)$ et $K \setminus K' = K \cap \complement_{V^*} K'$ sont des langages de Kleene.

Remarque :

- L'assertion b) s'étend trivialement à des réunions ou intersections d'un nombre fini de langages de Kleene. Par contre la réunion (et donc l'intersection) dénombrable de langages de Kleene n'est pas, en général, un langage de Kleene; par exemple nous savons que le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas un langage de Kleene; or $\{a^n b^n \mid n \in \mathbb{N}\} = \bigcup_{p=1}^{\infty} \{a^p b^p\}$ et les $\{a^p b^p\}$ sont, comme nous le montrerons plus bas (2.8.9.), des langages de Kleene..

- Une application de (2.8.3.) et la proposition suivante, réciproque de (2.8.1.) :

Soit K un langage de Kleene sur V et soit une partie V' de V; alors $K \cap V'^*$ est un langage de Kleene sur V'.

En effet V'^* est un langage de Kleene sur V' donc sur V d'après (2.8.1.), donc $K \cap V'^*$ est un langage de Kleene sur V (2.8.3.) mais aussi sur V', par construction.

2.8.4. Proposition.

L'image réciproque d'un langage de Kleene par homomorphisme entre monoïdes libres est un langage de Kleene.

Soit h un homomorphisme de V^* dans V'^*. Si K' est un langage de Kleene sur V', il existe un monoïde libre fini M, une partie M' de M et un homomorphisme ψ de V'^* dans M tels que :

$$K' = \psi^{-1}(M')$$

alors
$$h^{-1}(K') = h^{-1}(\psi^{-1}(M))$$
$$= (\psi \circ h)^{-1}(M').$$

$\psi \circ h$ est un homomorphisme de monoïde donc $h^{-1}(K')$ est un langage de Kleene.

2.8.5. Théorème.

Le transformé par transcription d'un langage de Kleene est un langage de Kleene.

Soient V et V' deux ensembles, K un langage de Kleene sur V et t une transcription de V^* sur V'^*.

Considérons les classes des contextes à droite pour $t(K)$:

$$D_{t(K)}(\alpha') = \{\beta' \in V'^* \mid \alpha'\beta' \in t(K)\}$$

or, puisque t est une transcription,

$$\alpha'\beta' \in t(K) \Longleftrightarrow (\exists \gamma \in K)\ (\alpha'\beta' = t(\gamma))$$
$$\Longleftrightarrow (\exists \alpha, \beta \in V^*)\ (\alpha\beta \in K \text{ et } \alpha' = t(\alpha) \text{ et } \beta' = t(\beta)).$$

$$D_{t(K)}(\alpha') = \{\beta' \in V'^* \mid (\exists \alpha, \beta \in V^*)\ (\alpha\beta \in K,\ t(\alpha) = \alpha',\ t(\beta) = \beta')\}$$
$$= \{\beta' \in V'^* \mid (\exists \alpha, \beta \in V^*)\ (\beta \in D_K(\alpha),\ t(\alpha) = \alpha',\ t(\beta) = \beta')\}$$
$$= \{\beta' \in V'^* \mid (\exists \alpha \in t^{-1}(\alpha'))\ (\exists \beta \in V^*)\ (\beta \in D_K(\alpha),\ t(\beta) = \beta')\}$$
$$= t^{-1}\left(\{\beta \mid (\exists \alpha \in t^{-1}(\alpha'))\ \beta \in D_K(\alpha)\}\right)$$
$$= t^{-1}\left(\bigcup_{\alpha \in t^{-1}(\alpha')} D_K(\alpha)\right)$$

$D_{t(K)}(\alpha')$

Comme les $D_K(\alpha)$ sont en nombre fini, il en est de même pour les $D_{t(K)}(\alpha')$ ce qui achève la démonstration.

2.8.6. Conséquences.

a) Tout langage déduit par transcription d'un langage local sur un vocabulaire fini est un langage de Kleene ne contenant pas le mot vide.

b) Les langages de Kleene sur un vocabulaire fini et ne contenant pas le mot vide sont les langages transcrits des langages locaux sur un vocabulaire fini et eux seuls.

c) Les langages de Kleene sur un vocabulaire fini et ne contenant pas le mot vide sont les transcrits des ensembles de chemins d'un graphe fini joignant un ensemble de points du graphe à un second ensemble de points du graphe.

Démonstration : L'assertion a) découle du théorème (2.4.2.) qui affirme que tout langage local sur un vocabulaire fini est un langage de Kleene; de plus a) n'est autre que la réciproque, déjà annoncée, du théorème (2.4.4.), d'où b) et c) (cf. (2.4.5.)).

A part la restriction faite à propos du mot vide, nous disposons d'une nouvelle caractérisation des langages de Kleene sur un vocabulaire fini; ainsi pour démontrer une propriété des langages de Kleene il suffira de la prouver pour les langages locaux et de montrer qu'elle est conservée par transcription.

2.8.7. Proposition : produit de deux langages de Kleene.

Le produit de deux langages de Kleene sur un vocabulaire fini est un langage de Kleene.

Soient deux langages de Kleene K_1 et K_2 sur V fini.

- Dans un premier temps supposons que K_1 et K_2 ne contiennent pas le mot vide. K_1 (resp. K_2) est le transcrit d'un langage local L_1 (resp. L_2) sur le vocabulaire V_1 (resp. V_2) par la transcription t_1 (resp. t_2). En outre nous pouvons choisir V_1 et V_2 (qui sont déterminés à une bijection près) disjoints; soit $V' = V_1 \cup V_2$; L_1 et L_2 sont évidemment des langages locaux sur V'; posons :

$$(\forall a \in V_1) \quad t(a) = t_1(a)$$
$$(\forall a \in V_2) \quad t(a) = t_2(a).$$

Ainsi t est une transcription de V'* dans V* et

$$K_1 = t(L_1) \quad , \quad K_2 = t(L_2)$$

donc $K_1 K_2 = t(L_1 L_2)$.

Pour achever la démonstration il suffit de montrer que $L_1 L_2$ est un langage local ((2.8.6.) a)). Associons à L_1 le triplet (D_1, F_1, τ_1) où D_1 est l'ensemble des débuts, F_1 l'ensemble des finales et τ_1 l'ensemble des transitions; de même, le triplet (D_2, F_2, τ_2) est associé à L_2 et envisageons le langage local L' associé à $(D_1, F_2, \tau_1 \cup \tau_2 \cup (F_1 \times D_2))$. Il est clair que $L_1 L_2$ est inclus dans L'. Réciproquement soit un mot α de L' : $\alpha = a_1 \ldots a_n$ avec $a_1 \in D_1$, $a_n \in F_2$; par suite $a_1 \in V_1$, $a_n \in V_2$. Soit i le premier entier tel que

$$a_i \in V_2 \qquad a_{i-1} \in V_1$$

nécessairement $(a_{i-1}, a_i) \in F_1 \times D_2$

donc a_{i-1} est dans F_1 et $a_1, \ldots, a_{i-1}$ sont dans V_1 par définition de i, c'est-à- dire que le facteur gauche $a_1 \ldots a_{i-1}$ de α est dans L_1. Par ailleurs a_i est dans D_2 et toute transition de $\tau_1 \cup \tau_2 \cup (F_1 \times D_2)$ qui a son premier élément dans V_2 a son deuxième élément dans V_2 : c'est donc une transition de τ_2 et par conséquent le facteur droit $a_i \ldots a_n$ est un élément de L_2; finalement α est dans $L_1 L_2$ qui n'est autre que le langage local L'.

- Si K_1 et K_2 sont deux langages de Kleene quelconques sur un vocabulaire fini il est facile de se ramener au cas précédent; par exemple si K_1 et K_2 contiennent tous deux Λ :

$K_1 = K'_1 \cup \{\Lambda\}$ $\qquad K_2 = K'_2 \cup \{\Lambda\}$ $\qquad$ où K'_1 et K'_2 ne contiennent pas Λ

$K_1 K_2 = K'_1 K'_2 \cup K'_1 \cup K'_2 \cup \{\Lambda\}$

2.8.8. Proposition : itéré d'un langage de Kleene.

Si K est un langage de Kleene sur un vocabulaire fini V, K* est un langage de Kleene sur V.

On peut toujours se ramener à un langage de Kleene K ne contenant pas le mot vide. En effet, si $K' = K \cup \{\Lambda\}$, $K'^* = K^*$.

Il existe un langage local L sur un vocabulaire fini V', associé au triplet (D, F, τ) et une transcription t tels que :

$$K = t(L),$$

par suite $\qquad K^* = t(L^*)$.

Il suffit de montrer que $L^* \setminus \{\Lambda\}$ est un langage local; en effet, dans ces conditions, $L^* \setminus \{\Lambda\}$ sera un langage de Kleene donc L^* ((2.8.2.) et (2.8.3.)) puis K^* (2.8.5.) seront aussi des langages de Kleene. Soit le langage local L' défini par $(D, F, \tau \cup (F \times D))$; montrons que $L^* \setminus \{\Lambda\}$ est L' sont égaux. L'inclusion

$$L^* \setminus \{\Lambda\} \subset L'$$

est évidente. Réciproquement soit α dans L'; en repérant dans α chaque transition appartenant à $F \times D$ décomposons α en

$$\alpha = \alpha_1 \dots \alpha_p \qquad p \geq 1$$

de telle façon que pour $i = 1, \dots, p$ la première lettre de α_i est dans D, sa dernière lettre dans F et toutes les transitions de α_i dans τ ; autrement dit α_i est dans L et α dans $L^* \setminus \{\Lambda\}$. Enndéfinitive $L^* \setminus \{\Lambda\} = L'$.

Remarque :

- On retrouve le fait que V^* est un langage de Kleene, par suite $\emptyset = \begin{cases} V^* \\ V^* \end{cases}$ est un langage de Kleene.
- Un langage réduit à un mot est un produit fini de langages réduits à une lettre : c'est un langage de Kleene; tout langage fini étant vide ou réunion finie de mots est un langage de Kleene. Ce résultat est inclus dans le théorème suivant :

2.8.9. Théorème de Kleene.

L'ensemble des langages de Kleene sur un vocabulaire V fini est le plus petit ensemble de $\mathcal{K}$ de langages sur V tel que :
a) $\mathcal{K}$ contient les langages finis.
b) Si A et B appartiennent à $\mathcal{K}$, $A \cup B$, $A \cdot B$ et A^* appartiennent à $\mathcal{K}$.

Démonstration. D'après ce qui précède, l'ensemble des langages de Kleene vérifie a) et b). Soit $\mathcal{E}$ un ensemble de langages vérifiant a) et b); montrons que tout langage de Kleene est dans $\mathcal{E}$; ainsi nous aurons prouvé que l'ensemble des langages de Kleene est le plus petit des ensembles $\mathcal{E}$. Si K est un langage de Kleene il existe un langage local L sur V', associé au triplet (D, F, τ) et une transcription t tels que :

$$K = t(L) \qquad \text{ou} \qquad K = t(L) \cup \{\Lambda\} \quad (\text{si } \Lambda \in K).$$

Il reste à démontrer que t (L) est dans $\mathcal{E}$; procédons par récurrence sur le nombre des transitions de L :

- $\tau = \emptyset$, alors $L = D \cap F$ donc t (L) est fini et, à ce titre, est dans $\mathcal{E}$.
- Si τ est non vide envisageons $(a, b) \in \tau$ et $\tau' = \tau \setminus \{(a, b)\}$.

Soit $\alpha \in L$. Si la transition (a, b) n'intervient pas dans α, α est dans le langage local L_1 associé à (D, F, τ'), sinon séparons dans α les facteurs α_i ne faisant pas intervenir la transition (a, b) :

$$\alpha = \alpha_1 \, ab \, \alpha_2 \, ab \ldots ab \, \alpha_p \qquad p \geqslant 2$$

avec $\alpha_1 \, a \in L_2$ où L_2 est le langage local associé à $(D, \{a\}, \tau')$

$(\forall i = 2, \ldots, p-1)$ $b \, \alpha_i \, a \in L_3$ où L_3 est le langage local associé à $(\{b\}, \{a\}, \tau')$

$b \, \alpha_p \in L_4$ où L_4 est le langage local associé à $(\{b\}, F, \tau')$

finalement $\alpha \in L_1 \cup L_2 \, L_3^* \, L_4$

donc $L \subset L_1 \cup L_2 \, L_3^* \, L_4$.

La réciproque est triviale et $L = L_1 \cup L_2 \, L_3^* \, L_4$; par suite :

$$t(L) = t(L_1) \cup t(L_2) \, t(L_3)^* \, t(L_4).$$

L'hypothèse de récurrence s'applique aux langages L_1, L_2, L_3, L_4 (car τ' est leur ensemble de transitions), c'est-à-dire que $t(L_1)$, $t(L_2)$, $t(L_3)$, $t(L_4)$ sont dans $\mathcal{E}$ donc t (L) est aussi dans $\mathcal{E}$ ce qui achève la démonstration.

2.8.10 Corollaire.

Les langages de Kleene sur un vocabulaire fini sont les composés de langages finis par un nombre fini de réunions de produits et d'itérations.

Soit $\mathcal{E}$ l'ensemble de tous ces composés; d'après le théorème de Kleene : $\mathcal{K} \subset \mathcal{E}$. En outre d'après l'étude antérieure les langages finis et leurs composés par un nombre fini de réunions de produits et d'itérations sont des langages de Kleene c'est-à-dire que $\mathcal{E} \subset \mathcal{K}$

en définitive $\mathcal{E} = \mathcal{K}$.

2.8.11 Proposition.

L'image homomorphe d'un langage de Kleene sur un vocabulaire fini est un langage de Kleene.

Soient deux ensembles finis V et V', un langage de Kleene K sur V et un homomorphisme h de V* dans V'*. K est un composé par un nombre fini de réunions, produits, itérations de langages finis. Autrement dit, avec la notation précédente : $K \in \mathfrak{E}$. Or pour deux langages A et B sur V :

i) si A est fini h (A) est fini

ii) $h(A \cup B) = h(A) \cup h(B)$, $h(AB) = h(A)\,h(B)$ et $h(A^*) = h(A)^*$.

Par suite h (K) est encore dans $\mathfrak{E}$ et d'après (2.8.10) h (K) est un langage de Kleene.

Remarque.

Pour le théorème de Kleene et ses deux corollaires l'hypothèse "V fini" est essentielle. En effet soit, sur V infini, l'ensemble $\mathfrak{E}$ des langages L sur V tels que l'ensemble des éléments de V qui ont une occurrence dans un mot de L soit fini. $\mathfrak{E}$ vérifie a) et b) de (2.8.9.) mais $\mathfrak{E}$ ne contient pas le langage de Kleene V*, autrement dit l'ensemble $\mathfrak{K}$ des langages de Kleene sur V n'est pas le plus petit ensemble vérifiant a) et b). D'autre part les composés des langages finis par un nombre fini de réunions, produits, itérations appartiennent tous à $\mathfrak{E}$. De même l'image homomorphe d'un langage de Kleene sur un vocabulaire infini V n'est pas, en général un langage de Kleene :
choisissons, par exemple, $V = \{a^n b^n \mid n \in \mathbb{N}\}$ et $V' = \{a, b\}$. V est un langage de Kleene sur V (2.8.2.). Si h est l'homomorphisme de V^* dans V'^* défini sur V par :

$$h(a^n b^n) = a^n b^n$$

et prolongé naturellement à V^*, $h(V) = \{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas un langage de Kleene sur V (c.f. (2.5.6) application)

2.9. Automates finis indéterministes.

Lorsqu'un automate fini "lit" une donnée, à chaque étape de sa lecture, il est dans un état déterminé de manière unique par la partie déjà lue. Imaginons des automates qui, à chaque étape, prennent, au hasard, un état parmi un ensemble d'états; la classe de ces automates contient les automates finis; la question se pose alors de savoir si ces automates acceptent d'autres langages que les langages de Kleene.

Dans ce paragraphe V désignera un vocabulaire fini.

2.9.1. Définition ; Automate indéterministe.

Un automate indéterministe sur V est un quadruplet comprenant :

- un ensemble S fini appelé ensemble des états.
- un ensemble P de triplets $(s, a, s') \in S \times V \times S$
- un élément s_o de S appelé état initial.
- une partie S' de S appelé ensemble final.

Un tel automate est noté (S, P, s_o, S').

2.9.2. Définition : Calcul d'un automate indéterministe.

Un calcul d'un automate indéterministe (S, P, s_o, S') est une suite finie de couples $(s_1, a_1), \ldots, (s_n, a_n)$ tels que

$$\text{pour } i = 1, \ldots, n \qquad (s_{i-1}, a_i, s_i) \in P$$

Le mot $a_1 \ldots a_n$ est appelé donnée du calcul, et s_n est le dernier état du calcul. Par convention s_o est le dernier état du calcul vide.

2.9.3. Définition : Mot et langage acceptés par un automate indéterministe.

Un mot α sur V est accepté par l'automate (S, P, s_o, S') s'il existe un calcul de donnée α et d'état final dans S'.

Le langage accepté par (S, P, s_o, S') est l'ensemble des mots acceptés.

Remarquons que tout automate fini $(S, ., s_o, S')$ est un automate indéterministe (S, P, s_o, S') avec $P = \{(s, a, s \,.\, a)\}$.

Par conséquent tout langage de Kleene est accepté par un automate indéterministe; nous allons démontrer la réciproque :

2.9.4. Théorème.

Tout langage accepté par un automate indéterministe est un langage de Kleene.

Soit $\mathcal{A} = (S, P, s_0, S')$ un automate indéterministe sur un vocabulaire fini V; et soit A le langage accepté par $\mathcal{A}$. Comme pour les automates finis, considérons l'ensemble des calculs non vides de $\mathcal{A}$ dont le dernier état est dans S' : ce n'est autre que le langage local L sur le vocabulaire fini $S \times V$, défini par :

$$D = \{ (s_1, a_1) \mid (s_0, a_1, s_1) \in P \}$$

$$F = \{ (s, a) \mid s \in S' \}$$

$$\tau = \{ ((s, a), (s', a')) \mid (s, a', s') \in P \}$$

Posons :

$$(\forall (s, a) \in S \times V) \qquad t(s, a) = a$$

l'application t se prolonge naturellement en une transcription de $(S \times V)^*$ dans V^* qui transforme le langage local L en $A \setminus \{\Lambda\}$ donc $A \setminus \{\Lambda\}$ est un langage de Kleene (2.8.6.) ainsi que A.

CHAPITRE 3 : LANGAGES de CHOMSKY

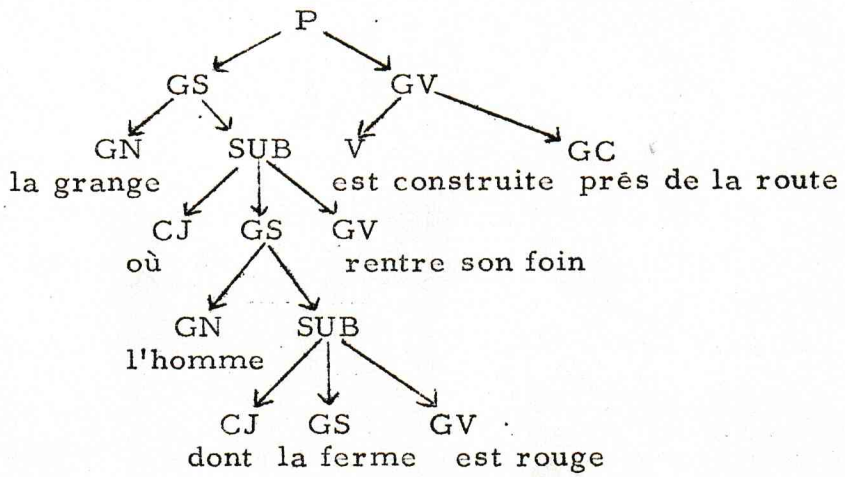
Les langages de Kleene ne sont pas suffisants pour décrire les langages de programmation ; en effet ALGOL 60, par exemple, n'est pas un langage de Kleene (pour s'en convaincre il suffit d'examiner les mots $\underline{\text{DEBUT}}^n$, où n est une puissance de concaténation ; chacun d'eux a une classe de contextes différente). D'une manière plus générale il apparaît qu'il n'est pas possible de décrire des structures imbriquées, comme des parenthèses, à l'aide des langages de Kleene ; or ces structures parenthétiques sont très fréquentes même dans les langues naturelles. Par exemple, dans la phrase suivante, les liens, représentés par des traits, sont imbriqués :

"La grange où l'homme dont la ferme est rouge rentre son foin est construite près de la route"

et rien n'interdit, en principe, d'intercaler au centre de nouvelles liaisons :

" ... dont la ferme que le voyageur a longée est rouge ... "

Ce genre de structure peut aussi être représenté par un schéma arborescent :



(où P représente la phrase, GS un groupe sujet, GV un groupe verbal, GN un groupe nominal, SUB une subordonnée, CJ une conjonction, GC un groupe complément).

Une des causes de ces insuffisances des langages de Kleene est que nous leur avons imposé des propriétés trop exigeantes : par exemple les catégories grammaticales liées à un langage sur V forment rarement une partition de V^* ; toutefois il arrive que ces catégories vérifient un système de relations : revenons au langage ALGOL 60, en appelant IC la catégorie des instructions conditionnelles EB celle des expressions booléennes, II celle des instructions inconditionnelles, I celle des instructions, et IP celle des instructions POUR, nous avons :

IC = SI EB ALORS ((II SINON I) $\cup$ II $\cup$ IP) (1)

(pour simplifier l'écriture nous notons sans accolade les ensembles réduits à un seul élément, - nous garderons cette convention au cours du chapitre). Dans le paragraphe suivant, nous allons définir les langages de Chomsky sur un vocabulaire T comme des sous-ensembles de T^* vérifiant certaines relations du type (1).

3.1 Définition d'un langage de Chomsky

3.1.1 Définition

Soit T un vocabulaire fini. Considérons un système (S) de n équations à n inconnues $A_1, A_2, \ldots, A_n$ à valeurs dans $\mathcal{P}(T^* \setminus \{\Lambda\})$:

$$(S) \quad \text{pour } i=1\ldots n \quad A_i = \bigcup_{j=1}^{p_i} B_{ij}^1 B_{ij}^2 \ldots B_{ij}^{q_{ij}} \quad \text{avec } B_{ij}^k \in \{A_1, \ldots, A_n\} \cup T.$$

(k est un indice)

Si (S) a une solution unique, chaque langage du n-uplet solution est appelé langage de Chomsky sur T (ou C-langage sur T).

Chacun des produits $B_{ij}^1 \ldots B_{ij}^{q_{ij}}$ sera appelé dans la suite un terme de la ième équation.

Remarquons que, si L est un langage de Chomsky, L ne contient pas le mot vide.

Exemple : sur le vocabulaire $T = \{a, b, c\}$ soit le langage

$L = \{a^n b^n c \mid n \in \mathbb{N}\}$ (L n'est pas un langage de Kleene (2.5)).

Posons $A = \{a^n b^n \mid n > 0\}$.

Il est évident que :

$$L = Ac \cup c$$
$$A = aAb \cup ab$$

Par suite le système

$$\begin{cases} A_1 = A_2\, c \cup c \\ A_2 = a A_2\, b \cup ab \end{cases}$$

admet pour solution $A_1 = L$ et $A_2 = A$; avant d'affirmer que L est un langage de Chomsky il nous faut montrer que cette solution est unique; cette difficulté est résolue par le théorème suivant, sans lequel la définition (3.1.1) serait inapplicable.

3.1.2 Théorème.

Si le système (S) défini en (3.1.1) est tel que :

$$(\forall i, j) \quad (q_{ij}^{l} = 1 \implies B_{ij}^{l} \in T)$$

Alors (S) admet au plus une solution dans $\mathcal{P}(T^* \setminus \{\Lambda\})$.

Préliminaires

Dans le système (S) notons $\mathcal{Q}$ le n-uplet $(A_i)_{1 \leq i \leq n}$; ainsi (S) peut être mis sous la forme :

$$\mathcal{Q} = \Phi(\mathcal{Q})$$

où Φ est l'application définie sur $(\mathcal{P}(T^* \setminus \{\Lambda\}))^n$ et à valeurs dans $(\mathcal{P}(T^* \setminus \{\Lambda\}))^n$ telle que :

pour $\mathcal{E} = (E_i)_{1 \leq i \leq n}$ $\Phi(\mathcal{E}) = (E'_i)_{1 \leq i \leq n}$, avec,

pour $i = 1, \ldots, n$, $E'_i = \bigcup_{j=1}^{p_i} C_{ij}^{1} \ldots C_{ij}^{q_{ij}}$ où $C_{ij}^{k} = E_p$ s'il existe p tel que

$B_{ij}^{k} = A_p$, et $C_{ij}^{k} = B_{ij}^{k}$ si $B_{ij}^{k} \in T$.

Démonstration. Appelons ordre d'une partie E non vide de T^* la longeur minimum des mots de E notée ordre [E] ; posons par convention

ordre $[\emptyset] = +\infty$.

L'odre d'un n-uplet $\mathcal{E} = (E_i)_{1 \leqslant i \leqslant n}$ de $\mathcal{P}(T^*)$ est défini par

$$\text{ordre } [\mathcal{E}] = \min_{1 \leqslant i \leqslant n} (\text{ordre } [E_i]).$$

Si E et F sont deux parties de T^* rappelons que la différence symétrique de E et F notée $E \triangle F$ est définie par :

$$E \triangle F = (E \cup F) \setminus (E \cap F)$$

Pour la différence symétrique de deux n-uplets $\mathcal{E} = (E_i)_{1 \leqslant i \leqslant n}$ et $\mathcal{F} = (F_i)_{1 \leqslant i \leqslant n}$ il est naturel de poser :

$$\mathcal{E} \triangle \mathcal{F} = (E_i \triangle F_i)_{1 \leqslant i \leqslant n}$$

Lemme : Sous les hypothèses du théorème et avec les notations que nous venons d'introduire :

$$(\mathcal{E} \neq \mathcal{F}) \Rightarrow (\text{ordre } [\varphi(\mathcal{E}) \triangle \varphi(\mathcal{F})] > \text{ordre } [\mathcal{E} \triangle \mathcal{F}]).$$

Démonstration du lemme. Si $\varphi(\mathcal{E}) = \varphi(\mathcal{F})$ alors $\varphi(\mathcal{E}) \triangle \varphi(\mathcal{F})$ est le n-uplet comportant n fois l'ensemble vide, par suite

$$\text{ordre } [\varphi(\mathcal{E}) \triangle \varphi(\mathcal{F})] = +\infty$$

et la relation à prouver est trivialement vérifiée. Supposons désormais $\varphi(\mathcal{E})$ et $\varphi(\mathcal{F})$ distincts et notons :

$$\Phi(\mathcal{E}) = (E'_i)_{1 \leqslant i \leqslant n}$$

$$\Phi(\mathcal{F}) = (F'_i)_{1 \leqslant i \leqslant n}$$

Par hypothèse il existe i tel que $E'_i \neq F'_i$. Soit donc un élément α de $E'_i \triangle F'_i$, par exemple :

$$\alpha \in E'_i \text{ et } \alpha \notin F'_i ,$$

ainsi α appartient à l'un des termes de E'_i noté $C^1 \ldots C^q$ avec :

pour $k = 1, \ldots, q$ (1) $C^k = E_{i_k}$

ou bien (2) $C^k \in T$ (c. f. définition de (S) et de Φ).

De même pour F'_i soit $D^1 \dots D^q$ l'homologue du terme $C^1 \dots C^q$:

$(1)'$ $(C^k = E_{i_k}) \Rightarrow (D^k = F_{i_k})$

$(2)'$ $(C^k \in T) \Rightarrow (D^k \in T)$.

Comme α appartient à $C^1 \dots C^q$, α se décompose :

$$\alpha = \alpha_1 \dots \alpha_q \quad \text{avec, pour } k = 1, \dots, q, \quad \alpha_k \in C^k.$$

Or α n'est pas dans F'_i c'est à dire qu'il existe j tel que

$$\alpha_j \notin D^j \quad \text{et} \quad \alpha_j \in C^j$$

Par suite D^j et C^j ne sont pas dans T (c. f. $(1)'$) et

$$D^j = F_{i_j}$$

$$C^j = E_{i_j} \qquad \text{(c. f. (1) et (1)').}$$

d'où ordre $[\, C^j \Delta D^j \,]$ = ordre $[\, E_{i_j} \Delta F_{i_j} \,]$

et les inégalités :

$$|\alpha_j| \geq \text{ordre } [\, C^j \Delta D^j \,] \geq \text{ordre } [\, \mathcal{E} \Delta F \,].$$

D'après l'hypothèse, comme C^j n'appartient pas à T, q est strictement supérieur à 1, de plus les α_k (k = 1,...,q) ne sont pas réduits au mot vide car les C_k sont des parties de $T^* \setminus \{\Lambda\}$, par conséquent :

$$|\alpha| > |\alpha_j|$$

Enfin les inégalités : et α pouvant être choisi de façon que :

ordre $[\Phi(\mathcal{E}) \Delta \Phi(F)] \geq$ ordre $[\, E'_i \Delta F'_i \,] \geq |\alpha| > |\alpha_j| \geq$ ordre $[\mathcal{E} \Delta F]$

prouvent le lemme.

Revenons à la démonstration du théorème : Si $\mathcal{E}$ et F sont deux solutions distinctes :

$\Phi(\mathcal{E}) = \mathcal{E}$

$\Phi(\mathcal{F}) = \mathcal{F}$ et $\mathcal{E} \neq \mathcal{F}$

ce qui contredit le lemme.

N.B. Le lecteur pourra comparer cette démonstration à celle du théorème des approximations successives dans un espace vectoriel normé (preuve de l'unicité de la solution).

3.1.3 Théorème.

Soit le système (S) défini en (3.1.1). Munissons l'ensemble $\mathcal{A} = \{A_1, \ldots, A_n\}$ de la relation ::= définie par :

$A_i ::= A_j$ si A_j est un terme de la ième équation de (S).

Si le graphe $(\mathcal{A}, ::=)$ est sans circuit le système (S) possède au plus une solution dans $\mathcal{P}(T^* \setminus \{\Lambda\})$.

Démonstration. Remarquons tout d'abord que ce théorème est une extension du précédent : en effet l'hypothèse du théorème précédent revient à supposer que le graphe $(\mathcal{A}, ::=)$ est sans arc.

A_1 n'est pas un terme de la première équation car $A_1 ::= A_1$ est faux par hypothèse.

Dans chacun des seconds membres où A_1 est un terme remplaçons A_1 par sa valeur donnée par la première équation : ainsi nous obtenons le système S_1 où A_1 n'est terme d'aucune équation ; réitérons le procédé avec A_2 : nous construisons S_2, et ainsi de suite. En éliminant successivement tous les A_i qui sont des termes nous nous ramenons aux conditions d'application du théorème (3.1.2).

Il reste à prouver que cette élimination est toujours possible c'est à dire que pour tout k, A_i n'est pas terme de la ième équation de S_k. Comme le graphe $(\mathcal{A}, ::=)$ est sans circuit il suffit de montrer que pour tout k, si A_j est terme de la ième équation de S_k, alors il existe un chemin non vide de A_i à A_j. Procédons par récurrence sur k : pour k = 0 la propriété est évidente ; supposons la vérifiée à l'ordre k-1 et soit A_j un terme de la ième équation de S_k :

- ou bien A_j est terme de la iême équation de S_{k-1} et, par hypothèse de récurrence, il existe un chemin de A_i à A_j.
- ou bien A_k est terme de la iême équation de S_{k-1} et A_j terme de la k iême équation de S_{k-1} ; donc, par hypothèse de récurrence il existe un chemin de A_i à A_k puis de A_k à A_j , ce qui achève la démonstration.

Remarque.

a) Si l'hypothèse du théorème n'est pas vérifiée, le système peut posséder plusieurs solutions ; par exemple sur un vocabulaire T contenant au moins la lettre a, les systèmes suivants ont une infinité de solutions :

$$\{A_1 = A_1 \qquad \begin{cases} A_1 = A_2 \\ A_2 = A_1 \end{cases} \qquad \begin{cases} A_1 = A_2 \cup a \\ A_2 = A_1 \end{cases}$$

b) Même si les hypothèses sont satisfaites, le système peut avoir plusieurs solutions dans $\mathcal{P}(T^*)$: par exemple :

$$\{ A = AA$$

admet les solutions $A = \emptyset$ $A = \{\Lambda\}$ $A = \{T^*\}$.

D'après le théorème 3.1.2, $\emptyset$ est la seule solution ne contenant pas le mot vide.

3.2. Notion de grammaire

Nous allons, dans ce paragraphe, envisager les langages du point de vue de leur génération : pour cela nous introduirons la notion de grammaire. Une grammaire décrit la manière d'engendrer les éléments des diverses catégories grammaticales. Nous montrerons ultérieurement que les langages ainsi définis ne sont autres que les langages de Chomsky.

3.2.1 Définition : grammaire.

Une grammaire G est un quadruplet comprenant :

- un ensemble fini T appelé vocabulaire terminal (ses éléments sont les symboles terminaux)
- un ensemble fini N disjoint de T appelé vocabulaire auxiliaire, ou vocabulaire non terminal (ses éléments sont les symboles non terminaux ou auxiliaires)

- une relation binaire, notée ::=, dans V^* ($V = T \cup N$) appelée relation de production, telle que
  i) le nombre des couples en relation est fini
  ii) $(\forall A, \varphi \in V^*)$ $(A ::= \varphi \Rightarrow A \in N$ et $\varphi \in V^* \setminus \{\Lambda\})$,
- un élément X distingué dans N et appelé axiomede la grammaire.

G est notée : G = (T, N, ::=, X). Un couple de la relation de production est appelé règle de la grammaire G. (i) indique que le nombre des règles est fini).

Précisons le rôle de la grammaire G à l'aide des définitions suivantes :

3.2.2 Définition

Etant donné une grammaire G, soit la relation dans V^*, notée $\rightarrowtail$, et définie par :

$(\forall \alpha, \beta \in V^*)$ $\alpha \rightarrowtail \beta \Longleftrightarrow (\exists A \in N, \lambda \in V^*, \lambda' \in V^*, \varphi \in V^* \setminus \{\Lambda\})(\alpha = \lambda A \lambda'$ et $\beta = \lambda \varphi \lambda'$ et $A ::= \varphi)$.

Nous disons que "α se réécrit β".

La fermeture transitive de la relation $\rightarrowtail$ est notée $\overset{*}{\rightarrowtail}$ et $\alpha \overset{*}{\rightarrowtail} \beta$ s'énonce "de α dérive β".

Une suite $\alpha \rightarrowtail \gamma_1 \rightarrowtail \gamma_2 \rightarrowtail \cdots \rightarrowtail \gamma_{n-1} \rightarrowtail \beta$ $(\alpha, \gamma_1, \ldots, \gamma_{n-1}, \beta \in V^*)$ est appelée dérivation de longueur n (une dérivation de longueur n est donc un chemin de longueur n du graphe infini $(V^*, \rightarrowtail)$. Si $\alpha = \beta$, nous dirons, par convention, qu'une dérivation de α à β a pour longueur 0.

3.2.3 Définition

Le langage engendré par la grammaire (T, N, ::=, X) est l'ensemble des mots de T^* qui dérivent de X.

Un langage engendré par une grammaire est appelé langage à contexte libre.

Remarque

Dans la définition d'une grammaire il est nécessaire de faire une hypothèse restrictive du genre de (i) dans 3.2.1 ; en effet, dans le cas contraire on engendrerait n'importe quel langage L de T^* en posant que X est en relation ::= avec toute phrase de L. Cependant on pourrait envisager des hypothèses plus larges par exemple que, pour tout $A \in N$, l'ensemble des mots φ tels que $A ::= \varphi$ soit un langage de Kleene.

Exemples :

1) Soit T = { le, chien, boulanger, promène } ,

N = { P, GS, GV, GN, V, GCD, AR, N }

Donnons nous les règles suivantes :

P ::= GS GV	AR ::= le
GS ::= GN	N ::= boulanger
GV ::= V GCD	N ::= chien
GCD ::= GN	V ::= promène
GN ::= AR N	

Si P est axiome, la grammaire, ainsi construite, engendre les phrases (et elles seules) :

le boulanger promène le chien

le chien promène le boulanger

2) Pour le langage ALGOL 60 nous pouvons envisager des règles du genre :

<E. A. S.> ::= <E. A. S.> <O. A> <T>

<E. A. S.> ::= <T.> où E. A. S. désigne une expression arithmétique simple, O. A. un opérateur additif et T. un terme (c.f. rapport ALGOL).

Avant d'étudier d'autres exemples, précisons quelques propriétés de la relation $\overset{*}{\longrightarrow}$, commodes dans les démonstrations ultérieures.

3.2.4 Proposition

Soit G une grammaire.

a) $(\forall \alpha, \beta \in V^*)\ ((\alpha \overset{*}{\longrightarrow} \beta) \Longrightarrow (|\alpha| \leq |\beta|))$

b) La relation $\overset{*}{\longrightarrow}$ est le plus petit préordre compatible avec la concaténation et qui contient ::=.

c) $(\forall \alpha, \alpha', \beta, \beta' \in V^*)\ ((\alpha \overset{*}{\longrightarrow} \beta \text{ et } \alpha' \overset{*}{\longrightarrow} \beta') \Longrightarrow (\alpha\alpha' \overset{*}{\longrightarrow} \beta\beta'))$.

d) Soient $B, C \in N \cup T$, $\alpha \in V^*$ tels que $BC \overset{*}{\longrightarrow} \alpha$; alors il existe $\beta, \gamma \in V^*$ tels que :

$$B \overset{*}{\longrightarrow} \beta,\ C \overset{*}{\longrightarrow} \gamma \text{ et } \alpha = \beta\gamma.$$

Démonstration. L'assertion a) est une conséquence directe de la condition ii) imposée à la relation ::= (c.f. 3.2.1).

b) D'après la définition de $\succ\!\!-$ il est évident que :

$(\alpha \succ\!\!- \beta) \Rightarrow ((\forall \gamma, \gamma' \in V^*) \quad \gamma\alpha\gamma' \succ\!\!- \gamma\beta\gamma')$

et par conséquent :

$(\alpha \overset{*}{\succ\!\!-} \beta) \Rightarrow ((\forall \gamma, \gamma' \in V^*) \quad \gamma\alpha\gamma' \overset{*}{\succ\!\!-} \gamma\beta\gamma')$.

La relation $\overset{*}{\succ\!\!-}$ est donc compatible avec la concaténation ; de plus, en tant que fermeture transitive, c'est un préordre ; enfin, il est clair que $\overset{*}{\succ\!\!-}$ contient la relation ::= . Soit R une relation de préordre compatible avec la concaténation et contenant ::=, alors :

$\alpha \succ\!\!- \beta \Rightarrow (\exists A \in N, \lambda, \lambda' \in V^*, \varphi \in V^* \setminus \{\Lambda\})(\alpha = \lambda A \lambda' \text{ et } \beta = \lambda\varphi\lambda' \text{ et } A ::= \varphi)$

or $A ::= \varphi \Rightarrow AR\varphi$

et $AR\varphi \Rightarrow \alpha R\beta$ (compatibilité de R avec la concaténation)

donc R contient $\succ\!\!-$; comme $\overset{*}{\succ\!\!-}$ est le plus petit préordre contenant $\succ\!\!-$, R contient $\overset{*}{\succ\!\!-}$ ce qui achève la démonstration.

c) La relation $\overset{*}{\succ\!\!-}$ est compatible avec la concaténation :

$\alpha \overset{*}{\succ\!\!-} \beta \Rightarrow \alpha\alpha' \overset{*}{\succ\!\!-} \beta\alpha'$

$\alpha' \overset{*}{\succ\!\!-} \beta' \Rightarrow \beta\alpha' \overset{*}{\succ\!\!-} \beta\beta'$

d'où, par transitivité : $\alpha\alpha' \overset{*}{\succ\!\!-} \beta\beta'$.

d) Procédons par récurrence sur la longueur n d'une dérivation de BC à α.

Pour n = 0 : $\alpha = BC$, $\beta = B$ et $\gamma = C$ répondent à la question ; supposons l'assertion vérifiée à l'ordre n-1 et soit α dérivant de BC par une dérivation de longueur n ; il existe $\alpha' \in V^*$ tel que :

$$BC \overset{*}{\succ\!\!-} \alpha' \succ\!\!- \alpha$$

et la dérivation $BC \overset{*}{\succ\!\!-} \alpha'$ satisfait à l'hypothèse de récurrence : soient donc β' et γ' de V^* tels que

$$B \overset{*}{\succ\!\!-} \beta' \qquad C \overset{*}{\succ\!\!-} \gamma' \qquad \alpha' = \beta'\gamma'$$

Or α' se réécrit α c'est à dire que, par exemple, β' se réécrit β et γ' est inchangé dans α :

$$\beta' \succ\!\!- \beta \quad \text{et} \quad \alpha = \beta\gamma',$$

finalement $B \overset{*}{\succ\!\!-} \beta \qquad C \overset{*}{\succ\!\!-} \gamma' \quad \text{et} \quad \alpha = \beta\gamma'$.

3.2.5 Remarque

L'assertion d) se généralise immédiatement de la façon suivante : soit une suite $(B_i)_{1 \leqslant i \leqslant q}$ d'éléments de $N \cup T$ tels que

$$B_1 B_2 \cdots B_q \overset{*}{\longrightarrow} \alpha$$

alors il existe une suite $(\beta_i)_{1 \leqslant i \leqslant q}$ de V^* ($\beta_i = B_i$ si B_i est dans T) tels que :

$$\forall\, i = 1, \ldots, q \qquad B_i \overset{*}{\longrightarrow} \beta_i \qquad \text{et} \qquad \alpha = \beta_1 \beta_2 \cdots \beta_q .$$

Exemple

Montrons que, sur le vocabulaire $T = \{a, b, c\}$ le langage $L = \{a^n b^n c \mid n \geqslant 0\}$ est à contexte libre. Choisissons le vocabulaire auxiliaire $N = \{X, A\}$ où X est l'axiome, et les règles :

(1) $X ::= Ac$ (3) $A ::= aAb$

(2) $X ::= c$ (4) $A ::= ab$

Soit $\mathcal{D}(X)$ l'ensemble des mots dérivant de X. Il s'agit de montrer que $\mathcal{D}(X) \cap T^* = L$. Soit $\alpha \in L$; si $\alpha = c$ de la règle (2)

Nous tirons : $X \longrightarrow c$

donc $c \in \mathcal{D}(X) \cap T^*$.

Supposons désormais que $\alpha = a^n b^n c$ avec $n > 0$; en appliquant la règle (1) puis (n-1) fois la règle (3) et enfin une fois la règle (4) nous obtenons la dérivation :

$$X \longrightarrow Ac \longrightarrow aAbc \longrightarrow \ldots \longrightarrow a^{n-1} A b^{n-1} c \longrightarrow a^n b^n c$$

donc $\alpha \in \mathcal{D}(X) \cap T^*$

Réciproquement soit $\alpha \in \mathcal{D}(X) \cap T^*$:

$$X \overset{*}{\longrightarrow} \alpha$$

nécessairement la première réécriture de X dans une dérivation de X à α est

$$X \longrightarrow c$$

ou

$$X \longrightarrow Ac$$

La longueur des éléments d'une dérivation étant croissante, si $|\alpha| = 1$ alors $\alpha = c$ et $\alpha \in L$, si $|\alpha| > 1$ alors la première réécriture dans la dérivation de X à α est :

$$X \rightarrowtail Ac$$

en outre $$Ac \overset{*}{\rightarrowtail} \alpha$$

donc il existe $\beta \in V^*$ tel que :

$$A \overset{*}{\rightarrowtail} \beta \qquad \alpha = \beta c \qquad (3.2.4\ d))$$

par suite comme α est dans T^* β est dans T^*. Il reste à montrer qu'il existe p tel que $\beta = a^p b^p$. Dans la dérivation de A à β la première réécriture de A est

$$A \rightarrowtail ab$$ dans ce cas $\beta = ab$

où $$A \rightarrowtail aAb.$$

Par conséquent si $|\beta| = 2$ la démonstration est terminée. Supposons que pour $|\beta| < n$ $(n > 2)$ β est du type $a^p b^p$ et soit β de longueur n :

$$A \rightarrowtail aAb \overset{*}{\rightarrowtail} \beta$$

donc il existe γ tel que

$$A \overset{*}{\rightarrowtail} \gamma, \quad \beta = a\gamma b \qquad (3.2.5)$$

et $$|\gamma| < |\beta|$$

donc $$(\exists p \in \mathbb{N})\ (\gamma = a^p b^p).$$

d'où $$\beta = a^{p+1} b^{p+1}$$

Remarque

La dénomination de "langage à contexte libre" (context free) rappelle que, si $A ::= \varphi$ est une règle, en remplaçant dans le mot α A par φ il n'est tenu aucun compte de ce qui entoure A dans α.

3.3. Comparaison des langages de Chomsky et des langages à contexte libre.

3.3.1 Lemme 1.

Soit un langage à contexte libre L de grammaire $G = (T, N, ::=, X)$. Pour tout $A \in N \cup T$ posons :

$$L(A) = \{\alpha \in T^* \mid A \overset{*}{\rightarrowtail} \alpha\}.$$

Si les p règles différentes de premier membre A s'écrivent

$$A ::= B_j^1 \ldots B_j^{q_j} \qquad (j = 1, \ldots, p) \qquad \text{avec } B_j^k \in N \cup T,$$

alors $$L(A) = \bigcup_{j=1}^{p} L(B_j^1) \ldots L(B_j^{q_j})$$

Soit $\alpha \in L(B_j^1) \ldots L(B_j^{q_j})$. Il existe $\beta_1, \beta_2, \ldots, \beta_{q_j}$ tels que

$$B_j^k \overset{*}{\rightarrowtail} \beta_k \qquad (k = 1, \ldots, q_j) \quad \text{et} \quad \alpha = \beta_1 \beta_2 \ldots \beta_{q_j}$$

donc $$A \rightarrowtail B_j^1 \ldots B_j^{q_j} \overset{*}{\rightarrowtail} \beta_1 \ldots \beta_{q_j}$$

et par suite $\alpha \in L(A)$.

Réciproquement soit $\alpha \in L(A)$:

$$A \overset{*}{\rightarrowtail} \alpha$$

Or A est élément de N et α est un mot sur T ; par suite A et α sont distincts et une dérivation de A à α est du type :

$$A \rightarrowtail B_{j_0}^1 \ldots B_{j_0}^{q_{j_0}} \overset{*}{\rightarrowtail} \alpha \qquad \text{où } j_0 \in \{1, \ldots, p\}.$$

D'après (3.2.5) il existe $\beta_1, \ldots, \beta_{q_{j_0}}$ tels que

pour $k = 1, \ldots, q_{j_0}$ $\quad B_{j_0}^k \overset{*}{\rightarrowtail} \beta_k$ et $\alpha = \beta_1 \ldots \beta_{q_{j_0}}$,

c'est à dire que α est élément de $L(B_{j_0}^1) \ldots L(B_{j_0}^{q_{j_0}})$ et, à fortiori :

$$\alpha \in \bigcup_{j=1}^{p} L(B_j^1) \ldots L(B_j^{q_j})$$

En conséquence le système construit à partir des différentes règles de la grammaire G selon le procédé indiqué dans le lemme admet au moins une solution.

3.3.2 **Théorème**

Tout langage de Chomsky est un langage à contexte libre.

Démonstration. Soit (S) un système associé à un langage de Chomsky L sur T :

$$(S) \quad \text{pour } i = 1, \ldots, n \quad \mathcal{A}_i = \bigcup_{j=1}^{p_i} \mathcal{B}_{ij}^1 \ldots \mathcal{B}_{ij}^{q_{ij}} \quad (\mathcal{B}_{ij}^k \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\} \cup T)$$

Par définition, le langage L est l'un des $\mathcal{A}_i$, par exemple $\mathcal{A}_1$. Posons

Posons $N = \{A_1, \ldots, A_n\}$ et soit le langage à contexte libre engendré par $G = (T, N ::=, A_1)$ où les règles sont :

$A_i ::= B_{ij}^1 \ldots B_{ij}^{q_j}$ pour $i = 1, \ldots, n$ et $j = 1, \ldots, p_i$ où les B_{ij}^k sont les éléments de $N \cup T$ tels que :

$$\begin{cases} \mathcal{B}_{ij}^k = \mathcal{A}_q \Rightarrow B_{ij}^k = A_q \\ \text{et} \quad \mathcal{B}_{ij}^k \in T \Rightarrow B_{ij}^k = \mathcal{B}_{ij}^k. \end{cases}$$

D'après le lemme 1 le n-uplet des $L(A_i)$ est solution du système (S) ; or (S) admet, par définition, une solution unique ; L est donc le langage, à contexte libre, $L(A_1)$.

3.3.3 Lemme 2

Tout langage à contexte libre engendré par une grammaire n'admettant pas de règle du type $A ::= A'$, avec $A, A' \in N$, est un langage de Chomsky.

Le lemme 1 prouve que tout langage à contexte libre L est l'une des solutions d'un système (S) ; l'hypothèse du lemme 2 assure que le système (S) n'admet pas l'une des inconnues comme terme · c'est à dire que (S) a, au plus, un n-uplet solution (3.1.2) ; par suite L est un langage de Chomsky. Pour établir la réciproque du théorème (3.3.2) il reste donc à se libérer de l'hypothèse en question ; c'est l'objet du lemme suivant.

3.3.4 Lemme 3

Soit un langage à contexte libre L engendré par une grammaire G. Il existe une grammaire G' telle que G' engendre L et G' n'admet pas de règle du type $A ::= A'$ avec $A, A' \in N$.

...

Soit $G = (T, N, ::=, X)$. Introduisons la grammaire $G' = (T, N, ::\stackrel{!}{=}, X)$ où la relation $::\stackrel{!}{=}$ est définie par :

$$A ::\stackrel{!}{=} \varphi \iff ((|\varphi| \geqslant 2 \text{ ou } \varphi \in T) \text{ et } ((\exists B \in N)(A \underset{G}{\overset{*}{\succ}} B \text{ et } B ::= \varphi)))$$

en notant $\underset{G}{\succ}$ (resp. $\underset{G'}{\succ}$) la relation de réécriture de la grammaire G (resp. G').

Montrons que :

$$(\forall \alpha, \beta \in V^*)\ (\alpha \underset{G'}{\overset{*}{\succ}} \beta \Rightarrow \alpha \underset{G}{\overset{*}{\succ}} \beta) \qquad (1).$$

Opérons par récurrence sur la longueur n d'une dérivation de α à β relativement à G'. Si $n = 0$ alors $\alpha = \beta$ et l'assertion (1) est trivialement vraie. Supposons (1) vérifiée à l'ordre n-1 et soit β dérivant de α par une dérivation de longueur n relativement à G' ; il existe $\gamma \in V^*$ tel que

$$\alpha \underset{G'}{\overset{*}{\succ}} \gamma \underset{G'}{\succ} \beta$$

et d'après l'hypothèse de récurrence :

$$\alpha \underset{G}{\overset{*}{\succ}} \gamma.$$

Soit $A ::\stackrel{!}{=} \varphi$ la règle utilisée pour réécrire γ :

$$(\exists \lambda, \lambda' \in V^*)\ (\gamma = \lambda A \lambda',\ \beta = \lambda \varphi \lambda')$$

de plus $\quad A ::\stackrel{!}{=} \varphi \Rightarrow (\exists B \in N)\ (A \underset{G}{\overset{*}{\succ}} B \text{ et } B ::= \varphi)$

donc $\quad \gamma \underset{G}{\overset{*}{\succ}} \lambda B \lambda' \underset{G}{\succ} \lambda \varphi \lambda' \qquad$ (compatibilité)

et $\quad \gamma \underset{G}{\overset{*}{\succ}} \beta \qquad$ (transitivité)

en définitive $\quad \alpha \underset{G}{\overset{*}{\succ}} \beta \qquad$ (").

Ainsi (1) est prouvé .

Montrons aussi que :

$$(\forall A \in N,\ \forall \beta \in T^*)\ \ (A \overset{*}{\underset{G}{\succ}} \beta \Rightarrow A \overset{*}{\underset{G'}{\succ}} \beta) \qquad (2)$$

Cette fois, c'est sur la longueur n de β que porte la récurrence. Soient $A \in N$ et $\beta \in T^*$ tels que

$$A \overset{*}{\underset{G}{\succ}} \beta$$

Toute dérivation de A à β est nécessairement de la forme :

$$A \underset{G}{\succ} A_1 \underset{G}{\succ} \dots \underset{G}{\succ} A_p \underset{G}{\succ} \varphi \overset{*}{\underset{G}{\succ}} \beta$$

avec $p \geqslant 0$, $A_1, \dots, A_p \in N$

et, ou $\varphi \in T^*$ (3)

ou $|\varphi| \geqslant 2$ (4)

Par conséquent $A ::\overset{!}{=} \varphi$

Si $|\beta| = 1$ nous sommes dans le cas (3), nécessairement $\varphi = \beta$ et $A \overset{*}{\underset{G'}{\succ}} \beta$. Supposons (2) vérifiée pour tout mot de longueur inférieure à n et soit β de longueur n ; φ s'écrit

$$\varphi = B_1 \dots B_q \qquad \text{avec, pour } i = 1, \dots, q \quad B_i \in N \cup T ;$$

d'après (3. 2. 5) il existe une suite $(\beta_i)_{1 \leqslant i \leqslant q}$ de T^* telle que

$$\beta = \beta_1 \dots \beta_q \qquad \text{et} \qquad B_i \overset{*}{\underset{G}{\succ}} \beta_i \qquad \text{pour } i = 1, \dots, q ;$$

or $|\beta_i| < |\beta|$ donc, par hypothèse de récurrence :

$$B_i \overset{*}{\underset{G'}{\succ}} \beta_i,$$

et, par concaténation (3. 2. 4 c) :

$$\varphi \overset{*}{\underset{G'}{\succ}} \beta$$

finalement $A \overset{*}{\underset{G'}{\succ}} \varphi \overset{*}{\underset{G'}{\succ}} \beta$

soit $A \overset{*}{\underset{G'}{\succ}} \beta.$

Grâce aux assertions (1) et (2) nous venons de montrer que les grammaires G et G' engendrent le même langage. Enfin la grammaire G' a été construite de façon qu'elle n'admette pas de règle du type :

$$A ::\overset{!}{=} A' \qquad \text{avec} \qquad A, A' \in N$$

Le lemme 2 peut donc s'appliquer à la grammaire G' ; d'où le théorème suivant qui complète le théorème 3. 3. 2.

3. 3. 3 Théorème

L'ensemble des langages de Chomsky sur un vocabulaire T est l'ensemble des langages à contexte libre sur T.

3. 3. 6 Remarques

1) Désormais nous pourrons considérer un langage de Chomsky comme associé à un système ou à une grammaire.

2) Le passage d'un système à la grammaire correspondante (et réciproquement) est immédiat (c. f. lemme 1).

Par exemple, sur $T = \{a, b, c\}$, $L = \{a^n b^n c \mid n \geqslant 0\}$ est associé au système :

$$(S) \quad \begin{cases} L = A\,c \cup c \\ A = a\,A\,b \cup a\,b \end{cases} \qquad L, A \in \mathcal{P}(T^* \setminus \{\Lambda\})$$

ou à la grammaire $G = (T, N, ::= X)$

$N = \{A, X\}$

$X ::= A\,c \mid c$

$A ::= a\,A\,b \mid a\,b.$

Les barres "|" sont des commodités de notation : elles séparent les règles de même premier membre et de seconds membres différents (par exemple $X ::= A\,c$ et $X ::= c$ est écrit $X ::= A\,c \mid c$).

De même, pour montrer que le langage $L = \{a^n b^n c^p \mid n > 0, p > 0\}$ est un langage de Chomsky sur $T = \{a, b, c\}$, il suffit de vérifier que le système d'inconnues A, B, C :

$A = B\,C \cup B \cup C$

$B = a\,B\,b \cup a\,b$

$C = c \cup c\,C$

admet pour solution (unique d'après (3.1.3)) : $A = L$, $B = \{a^n b^n \mid n > 0\}$, $C = \{c^p \mid p > 0\}$. Par suite L est le langage engendré par la grammaire :

$G = (T, N, ::=, X)$ avec $N = \{B, C, X\}$ et

$X ::= BC \mid B \mid C$

$B ::= aBb \mid ab$

$C ::= c \mid cC$.

En résumé, pour passer de (S) à G il suffit, formellement, de remplacer les signes $\cup$ par des $\mid$ et les $=$ par $::=$.

3.4. Propriétés des langages de Chomsky

3.4.1 Proposition : propriétés ensemblistes

Soient L et L', deux langages de Chomsky sur T ; alors :

a) $L \cup L'$ et $L \cdot L'$ sont des langages de Chomsky.

b) $L^* \setminus \{\Lambda\}$, noté $\overline{L}$ est un langage de Chomsky ($\overline{L} = \bigcup_{n=1}^{\infty} L^n$).

a) Par hypothèse L (resp. L') appartient à la solution d'un système (S) (resp. S'). Supposons que les inconnues de (S) et (S') ont des noms distincts entre elles et distincts de X ; appelons A_1, par exemple, l'inconnue relative à L et A'_1 celle relative à L'. Les systèmes :

$$\text{(I)} \quad \begin{cases} (S) \\ (S') \\ X = A_1 \cup A'_1 \end{cases} \qquad \text{(II)} \quad \begin{cases} (S) \\ (S') \\ X = A_1 \, A'_1 \end{cases}$$

ont évidemment une solution unique. Dans (I) (resp. (II)) $X = L \cup L'$ (resp. $X = L \cdot L'$) convient, ce qui prouve a).

b) L est associé à l'inconnue A_1 dans un système (S) choisi de façon qu'aucun terme n'est réduit à une inconnue.

Considérons le système

$$\text{(III)} \quad \begin{cases} (S) \\ X = A_1 \, X \cup A_1 \end{cases}$$

où X n'est pas une inconnue de (S). Ce système à une solution où $\bar{L}$ est associé à X et cette solution est unique car le choix de (S) est tel que (III) satisfait à l'hypothèse du deuxième théorème d'unicité (3. 1. 3).

Remarque :

Comme les démonstrations précédentes font appel à des systèmes il est facile d'écrire, dans chacun des cas, une grammaire correspondante ; par exemple, si la grammaire $G = (T, N, ::=, Y)$ engendre L, $\bar{L}$ est engendré par $\bar{G} = (T, N \cup \{X\}, \bar{::=}, X)$ où : $(X \notin N)$ et $(A \bar{::=} \varphi) \Leftrightarrow ((A ::= \varphi)$ ou $(A=X$ et $\varphi = YX)$ ou $(A=X$ et $\varphi = Y))$.

3. 4. 2 Proposition

Le réfléchi d'un langage de Chomsky est un langage de Chomsky.

Soit un langage de Chomsky L associé au système

$$(S) \qquad L(A_i) = \bigcup_{j=1}^{p_i} L(B_{ij}^1) \dots L(B_{ij}^{q_{ij}}) \qquad \text{pour } i = 1, \dots, n$$

(mêmes notations qu'en (3. 3. 1)), avec, par exemple, $L = L(A_1)$. En outre (S) est choisi de façon qu'il satisfait à l'hypothèse du premier théorème d'unicité (3. 1. 2) ; il est aisé de vérifier que :

$$\widetilde{L}(A_i) = \bigcup_{j=1}^{p_i} \widetilde{L}(B_{ij}^{q_{ij}}) \dots \widetilde{L}(B_{ij}^1) \qquad \text{pour } i = 1, \dots, n.$$

Le système $\widetilde{(S)}$ ainsi construit admet comme (S) une solution unique dont $\widetilde{L} = \widetilde{L(A_1)}$ est élément ; par suite $\widetilde{L}$ est un langage de Chomsky.

Remarquons que, si $G = (T, N, ::=, X)$ est une grammaire engendrant L, la grammaire $\widetilde{G} = (T, N, \widetilde{::=}, X)$, avec :

$$A \widetilde{::=} \varphi \Leftrightarrow A ::= \widetilde{\varphi}$$

engendre $\widetilde{L}$.

3. 4. 3 Proposition : transformation par transcription

Le transcrit d'un langage de Chomsky est un langage de Chomsky.

Soient deux vocabulaires T et T', une transcription t de T^* dans T'^* et un langage de Chomsky L sur T. Soit toujours (S) un système vérifiant

l'hypothèse du premier théorème d'unicité (3. 1. 2) et tel que $L = L(A_1)$:

$$(S) \qquad \text{pour } i=1,\dots,n \quad L(A_i) = \bigcup_{j=1}^{p_i} L(B_{ij}^1) \dots L(B_{ij}^{q_{ij}})$$

Comme t est un homomorphisme :

$$(S') \qquad \text{pour } i=1,\dots,n \quad t[L(A_i)] = \bigcup_{j=1}^{p_i} t[L(B_{ij}^1)] \dots t[L(B_{ij}^{q_{ij}})] .$$

t étant une transcription aucun des $t[L(A_i)]$ et $t[L(B_{ij}^k)]$ n'est réduit au mot vide, ce qui a deux conséquences essentielles :

(1) les n relations (S') constituent un système sur $\mathcal{P}(T'^* \setminus \{\Lambda\})$;

(2) les termes de ce système, comme ceux de (S), ne sont jamais réduits à une inconnue.

Ainsi (S') a une solution unique dont t(L) fait partie ; t(L) est un langage de Chomsky.

Soit, en outre, $G = (T, N, ::=, X)$ une grammaire acceptant L ; de la construction de (S') nous déduisons une grammaire $G' = (T', N, ::\stackrel{!}{=}, X)$ acceptant t(L) en posant :

$$(\forall A \in N, \varphi \in V^*) \qquad A ::= \varphi \iff A ::\stackrel{!}{=} t_1(\varphi)$$

où $t_1(\varphi)$ est déduit de φ en laissant invariant les symboles non terminaux de φ et en remplaçant tout symbole terminal par son image par t (par exemple pour $A, B \in N, a \in T$, $A ::= Ba \iff A ::\stackrel{!}{=} Bt(a)$).

Le raisonnement reste inchangé pour tout homomorphisme h tel que

$$(\forall a \in T) \qquad h(a) \neq \Lambda .$$

Dans le cas d'un homomorphisme quelconque nous admettrons le résultat suivant.

3. 4. 4 Théorème

Le transformé d'un langage de Chomsky par homomorphisme entre monoïdes libres est un langage de Chomsky éventuellement réuni à $\{\Lambda\}$.

3.4.5 Théorème

Tout langage de Kleene ne contenant pas le mot vide est un langage de Chomsky.

Démonstration. Nous allons montrer que l'ensemble $\mathcal{C}$ des langages de Chomsky et des langages de Chomsky réunis au mot vide satisfait aux propriétés suivantes :

a) $\mathcal{C}$ contient les langages finis.

b) Si A et B sont des langages de $\mathcal{C}$, $A \cup B$, AB et A^* sont des langages de $\mathcal{C}$.

Le théorème de Kleene prouve alors que tout K-langage est élément de $\mathcal{C}$.

Isolons le résultat a), qui est intéressant en soi :

Proposition

Soit un vocabulaire T. Tout sous ensemble fini de T^* ne contenant pas le mot vide est un langage de Chomsky.

En effet $\emptyset$ est l'unique solution du système

$$X = XX \qquad \text{(c. f. 3. 1. 3 remarque b))};$$

et tout sous ensemble fini non vide de T^* $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ est l'unique solution du système

$$X = \alpha_1 \cup \alpha_2 \cup \ldots \cup \alpha_n,$$

ce qui prouve a) et la proposition.

b) est une conséquence immédiate de la proposition (3. 4. 1 b) c)) (par exemple si A est dans $\mathcal{C}$, $A^* = \overline{A \setminus \{\Lambda\}} \cup \{\Lambda\}$ est dans $\mathcal{C}$).

Remarques :

- La réciproque du théorème (3. 4. 5) est évidemment fausse et les langages de Chomsky généralisent strictement les langages de Kleene (exemple $L = \{a^n b^n c \mid n \geqslant 0\}$).

- La démonstration précédente a l'inconvénient de ne fournir aucun renseignement sur la forme des grammaires engendrant les langages de Kleene ; étant donné un langage de Kleene K sur T, ne contenant pas le mot vide, nous allons construire un système associé à K. Supposons que K est défini comme

image réciproque d'une partie S' de la mémoire finie S, par l'homomorphisme de mémoire φ de T^* dans S. Posons , pour $s \in S$:

$$K(s) = \{\alpha \in T^* \setminus \{\Lambda\} \mid \varphi(\alpha) = s\}$$
$$= \varphi^{-1}(s) \setminus \{\Lambda\}$$

alors :

$$\alpha \in K(s) \Longleftrightarrow (|\alpha| > 1 \text{ et } \alpha \in K(s)) \text{ ou } (\alpha \in T \text{ et } \alpha \in K(s))$$

$$\Updownarrow$$

$$((\exists s' \in S \text{ et } \exists \alpha' \in T^* \setminus \{\Lambda\} \text{ et } \exists a \in T)(\alpha = \alpha' a \text{ et } \varphi(\alpha') = s' \text{ et } s = s'a))$$

$$\text{ou } (\alpha \in T \text{ et } s_0 . \alpha = s)$$

$$\Updownarrow$$

$$((\exists s' \in S \text{ et } \exists \alpha' \in K(s') \text{ et } \exists a \in T)(\alpha = \alpha' a \text{ et } s = s'a)) \text{ ou } (\alpha \in T \text{ et } s_0 . \alpha = s)$$

$$\Updownarrow$$

$$\alpha \in \left(\bigcup_{\substack{a \in T \\ s' \in \{s' \in S \mid s'.a = s\}}} K(s')a \right) \cup \left(\bigcup_{a \in \{a \in T \mid s_0 . a = s\}} a \right)$$

Ainsi K et les ensembles K(s) vérifient les relations :

$$\begin{cases} K = \bigcup_{s \in S'} K(s) \\ K(s) = \left(\bigcup_{\substack{a \in T \\ s' \in \{s' \in S \mid s'.a = s\}}} K(s')a \right) \cup \left(\bigcup_{a \in \{a \in T \mid s_0 . a = s\}} a \right) \end{cases}$$

Si n est le nombre d'éléments de S ces relations forment un système de n+1 équations et n+1 inconnues sur $\mathcal{P}(T^* \setminus \{\Lambda\})$. Ce système admet au moins une solution dont K est élément et une au plus, car il vérifie l'hypothèse du deuxième théorème d'unicité (3. 1. 3) ; en effet la première équation est la seule ayant des termes réduits à une inconnue ; donc K est un langage de Chomsky. De plus il existe une grammaire G = (T, N, ::=, X)(associée au système obtenu en remplaçant les K(s) par leur valeur dans la première équation) qui engendre K et n'admet que des règles du type :

$$A ::= Ba \qquad \text{ou } A ::= a \qquad \text{avec } A, B \in N,\ a \in T.$$

Donnons un nom à de telles grammaires :

3.4.6 Définitions : Grammaires linéaire gauche, droite ; grammaire linéaire normale.

Une grammaire $G = (T, N, ::=, X)$ ne comprenant que des règles du type :

$A ::= Ba$ $\quad A ::= a$ $\quad (A, B \in N, a \in T)$

(resp. $A ::= aB$ $\quad A ::= a$ $\quad (A, B \in N, a \in T)$)

est appelée grammaire linéaire gauche (resp. grammaire linéaire droite)

Une grammaire G ne comprenant que des règles du type :

$A ::= aB$ $\quad A ::= Ba$ $\quad A ::= a$ $\quad (A, B \in N, a \in T)$

s'appelle grammaire linéaire normale.

Nous venons de montrer que tout langage de Kleene ne contenant pas le mot vide peut être engendré par une grammaire linéaire droite. Il est normal de se poser le problème réciproque. Nous obtenons le résultat suivant

3.4.7 Théorème

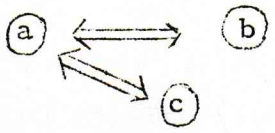
Soit un langage K sur T ; les trois assertions suivantes sont équivalentes

a) K est un langage de Kleene ne contenant pas le mot vide.

b) K est engendré par une grammaire linéaire gauche.

c) K est engendré par une grammaire linéaire droite.

Démonstration.



(a) $\Longrightarrow$ (b) : déjà prouvé (3.4.5 Remarque).

(a) $\Longrightarrow$ (c) : Si K est un langage de Kleene ne contenant pas de mot vide, $\tilde{K}$ vérifie la même hypothèse (2.7.2) et $\tilde{K}$ est engendré par une grammaire linéaire gauche, donc $K = \tilde{\tilde{K}}$ est engendré par une grammaire linéaire droite (3.4.2).

ⓒ $\Longrightarrow$ ⓐ : Soit un langage K engendré par la grammaire linéaire droite. $(T, N, ::=, X)$. Soit l'automate indéterministe $\mathcal{A}$ d'ensemble d'états $S = N \cup \{s_f\}$ (s_f n'appartenant pas à N), d'état initial $s_0 = X$, d'ensemble final $S' = \{s_f\}$ et dont l'ensemble des transitions P est défini par

$(A, a, B) \in P \Longleftrightarrow (A, B \in N, a \in T$ et $A ::= aB)$ ou $(A \in N, B = s_f, a \in T$ et $A ::= a)$

Appelons L le langage de Kleene engendré par $\mathcal{A}$ et montrons que K = L ; L ne contient pas le mot vide (car $X \neq s_f$) donc tout $\alpha \in L$ s'écrit $a_1 \ldots a_n$;

$$\alpha = a_1 \ldots a_n \in L$$

$$\Updownarrow$$

$(\exists B_1, \ldots, B_{n-1} \in N)$ (la suite $(B_i, a_i)_{1 \leq i \leq n-1} (s_f, a_n)$ est un calcul de $\mathcal{A}$)

$$\Updownarrow$$

$(\exists B_0, B_1, \ldots, B_{n-1} \in N)(B_0 = X$ et, pour $i = 1, \ldots, n-1, (B_{i-1}, a_i, B_i) \in P$ et $(B_{n-1}, a_n, s_f) \in P)$

$$\Updownarrow$$

$(\exists B_0, B_1, \ldots, B_n \in N)(B_0 = X$ et, pour $i = 1, \ldots, n-1, B_{i-1} ::= a_i B_i$ et $B_{n-1} ::= a_n)$

$$\Updownarrow$$

$$\alpha \in K$$

Ainsi c) entraine a).

ⓑ $\Longrightarrow$ ⓐ : Si K est un langage engendré par une grammaire linéaire gauche, $\tilde{K}$ est engendré par une grammaire linéaire droite (3. 4. 2) et $\tilde{K}$ est un langage de Kleene (ⓒ $\Longrightarrow$ ⓐ), par suite $K = \tilde{\tilde{K}}$ est un langage de Kleene.

Remarque.

Une conséquence immédiate du théorème 3. 4. 7 est que les langages de Kleene sont engendrés par des grammaires linéaires normales, mais la réciproque est fausse : en effet le langage $L = \{ a^n b^n \mid n > 0 \}$ qui n'est pas de Kleene est engendré par la grammaire linéaire normale :

$T = \{a,b\}$, $N = \{A,B\}$, $X = A$ et les règles :

$A ::= aB$

$B ::= Ab \mid b$

(Pour s'en assurer le lecteur pourra montrer que L satisfait au système associé à cette grammaire).

Pour compléter la proposition (3.4.1) qui étudie les propriétés ensemblistes des langages de Chomsky nous allons montrer que l'intersection de deux C-langages n'est pas toujours un C-langage ; il suffit d'exhiber un contrexemple. Pour cela envisageons un langage de structure plus complexe que la structure parenthétique :

3.4.8 Contrexemple des langages de Chomsky

Sur le vocabulaire $T_0 = \{a,b,c\}$ le langage $L_0 = \{a^n b^n c^n \mid n > 0\}$ n'est pas un langage de Chomsky.

Pour prouver cette assertion montrons d'abord le lemme suivant :

Lemme : Soit un langage de Chomsky infini L sur un vocabulaire T. Il existe une grammaire G engendrant L telle que pour aucun symbole non terminal A, l'ensemble L (A) des mots de T^* qui dérivent de A ne soit réduit à un mot.

Si G est une grammaire de L ne satisfaisant pas au lemme, soit α le seul mot qui dérive du non terminal A ; l'une des équations du système (S) associé à G est

$$L(A) = \{\alpha\}$$

Le système (S_1) obtenu en supprimant cette équation et en remplaçant dans les autres L(A) par $\{\alpha\}$ est évidemment équivalent à (S) pour les inconnues autres que A ; A n'est pas l'axiome de L car L est infini ; donc l'une des grammaires associées à (S_1) engendre L. Comme le vocabulaire auxiliaire N est fini, nous obtiendrons, après un nombre fini de transformations du même genre, un système (S_k) tel qu'une grammaire associée à (S_k) engendre L et vérifie le lemme.

Etudions le langage L_0. Pour montrer que L_0 n'est pas un langage de Chomsky il suffit de prouver que, pour tout langage de Chomsky L sur T_0 :

$$L_0 \subset L \quad \Longrightarrow \quad L_0 \neq L$$

L_0 et L sont infinis ; supposons donc L engendré par une grammaire $G = (N, T_0, ::=, X)$ satisfaisant au lemme. Tout mot $a^n b^n c^n$ dérive de X donc :

$$(\exists \gamma \in (N \cup T_0)^*) \qquad (X \overset{*}{\longrightarrow} \gamma \longrightarrow a^n b^n c^n)$$

c'est à dire :

$$(\exists \gamma, \lambda, \lambda' \in (N \cup T_0)^*, A \in N) \qquad (\gamma = \lambda A \lambda', A ::= \varphi, a^n b^n c^n = \lambda \varphi \lambda')$$

Le nombre de règles étant fini nous pouvons choisir n supérieur à la longueur du second membre de toute règle :

$$n > |\varphi|$$

par suite $\quad |\lambda| + |\lambda'| > 2n$

donc $\quad |\lambda| > n \quad$ ou $\quad |\lambda'| > n,$

étudions, par exemple, le cas $|\lambda| > n$:

$$(\exists \mu \in T^*) \qquad (\lambda = a^n b \mu)$$

ainsi λ est facteur gauche de tout mot dérivé de γ et n'est facteur gauche que d'un seul mot de L_0, donc de γ dérive un seul mot de L_0 ; or, par hypothèse, de A dérivent plusieurs mots de L, donc de γ également et L_0 est distinct de L.

Une conséquence immédiate est le résultat suivant :

3.4.9 Proposition

a) L'intersection de deux langages de Chomsky n'est pas toujours un langage de Chomsky.

b) Soit L un langage de Chomsky sur T ; le complément de L dans $T^* \setminus \{\Lambda\}$ n'est pas toujours un langage de Chomsky.

c) La différence de deux langages de Chomsky n'est pas toujours un langage de Chomsky.

a) Sur le vocabulaire $T_0 = \{a,b,c\}$ les langages $L_1 = \{a^n b^n c^p \mid n>0, p>0\}$ et $L_2 = \{a^p b^n c^n \mid n>0, p>0\}$ sont des langages de Chomsky ($L_1 = \{a^n b^n \mid n>0\}\{c\}$, et $\tilde{L}_2$, transcrit de L_1 est un langage de Chomsky, donc $L_2 = \tilde{\tilde{L}}_2$ l'est aussi). De plus :

$$L_0 = L_1 \cap L_2$$

b) Si le complément de L dans $T^* \setminus \{\Lambda\}$ était toujours un langage de Chomsky a) serait faux car

$$L_1 \cap L_2 = \complement(\complement L_1 \cup \complement L_2)$$

N. B. Naturellement le complément de L dans T^* n'est pas un langage de Chomsky car il contient Λ.

c) Soit L un langage de Chomsky et posons :

$$L' = (T^* \setminus \{\Lambda\}) \setminus L = \complement_{T^* \setminus \{\Lambda\}} L$$

$T^* \setminus \{\Lambda\}$ est un langage de Kleene ne contenant pas le mot vide ; donc L' est différence de deux langages de Chomsky et, d'après b), n'est pas toujours un langage de Chomsky.

3. 4. 10 Proposition

L'intersection d'un langage de Kleene et d'un langage de Chomsky est un langage de Chomsky.

Soit un langage de Chomsky engendré par une grammaire $G = (N, T, ::=, A_1)$ avec $N = \{A_1, \ldots, A_i, \ldots, A_n\}$. Posons :

$$L(A_i) = \{\alpha \in T^* \mid A_i \xrightarrow{*} \alpha\}$$

Les $L(A_i)$ sont solutions du système (S) associé à G (G est choisie de façon que (S) vérifie la première condition d'unicité(3.1.2)) ; donc, avec les notations habituelles :

pour $i = 1, \ldots, n$ $\quad L(A_i) = \bigcup_{j=1}^{p_i} L(B_{ij}^1) \ldots L(B_{ij}^{q_{ij}})$

Soit, d'autre part, un langage de Kleene K sur T défini comme image inverse d'une partie M' d'un monoïde fini M :

$K = \psi^{-1}(M')$ où ψ est un homomorphisme de monoïde.

Examinons les langages

$L(A_i) \cap \psi^{-1}(m)$ pour $i = 1,\dots,n$ et $m \in M$

notés $L(A_i,m)$. Leur nombre est fini et ils sont caractérisés de la façon suivante :

$$\alpha \in L(A_i,m)$$

$$\Updownarrow$$

$$\alpha \in L(A_i) \quad \text{et} \quad \psi(\alpha) = m$$

$$\Updownarrow$$

$$(\exists j \in [1,p_i])(\exists \beta_1,\dots,\beta_{q_{ij}})(\alpha = \beta_1 \dots \beta_{q_{ij}} \text{ et } \beta_k \in L(B_{ij}^k) \text{ pour } k=1,\dots,q_{ij}$$

$$\text{et } \psi(\beta_1)\dots\psi(\beta_{q_{ij}}) = m) \text{ (car } \psi \text{ est un homomorphisme)}$$

$$\Updownarrow$$

$$(\exists m_1,\dots,m_{q_{ij}}, m_k \in M \text{ pour } k=1,\dots,q_{ij})(\alpha \in L(B_{ij}^1, m_1)\dots L(B_{ij}^{q_{ij}}, m_{q_{ij}})$$

$$\text{et } m_1 \dots m_{q_{ij}} = m)$$

$$\Updownarrow$$

$$\alpha \in \bigcup_{j=1}^{p_i} \bigcup_{m \in \{m \in M \mid m_1 \dots m_{q_{ij}} = m\}} L(B_{ij}^1, m_1) \dots L(B_{ij}^{q_{ij}}, m_{q_{ij}})$$

Ainsi les $L(A_i,m)$ sont solutions d'un système d'équations. Ce nouveau système vérifie, comme (S), la première condition d'unicité ; les $L(A_i,m)$ sont donc des langages de Chomsky ; or :

$$K \cap L(A_1) = \bigcup_{m \in M'} L(A_1,m)$$

Par suite $K \cap L(A_1)$ est un langage de Chomsky.

3. 5. Décidabilité des langages de Chomsky.

3. 5. 1 Définition

Un langage est dit décidable s'il existe pour lui un algorithme de reconnaissance.

(Cette définition gagnera en clarté lorsque les "algorithmes" seront, eux-mêmes, dans un chapitre ultérieur, définis rigoureusement. Pour l'instant algorithme signifie intuitivement : méthode de calcul exécutable en un temps fini).

Nous avons montré (2. 1. 5) que les langages de Kleene sont décidables ; en est-il de même pour les langages de Chomsky ?

Soit L un langage de Chomsky engendré par G = (T, N, ::=, X) et soit $\alpha \in T^*$:

$$\alpha \in L \iff X \overset{*}{\succ\!\!-} \alpha$$

Envisageons le graphe infini, $\Gamma = ((T \cup N)^*, \succ\!\!-)$; α est mot de L si et seulement si il existe un chemin d'origine X et d'extrémité α dans Γ ; or tous les mots d'un tel chemin sont de longueur inférieure ou égale à $|\alpha|$ (3.2.4) ; par suite α est mot de L si et seulement si il existe un chemin d'origine X et d'extrémité α dans le sous graphe fini de Γ défini sur les mots de longueur inférieure ou égale à $|\alpha|$.

Or il existe des algorithmes généraux étudiant l'existence de chemins pour les graphes finis (c. f. cours de théorie des graphes). D'où l'énoncé :

3. 5. 2 Théorème

Les langages de Chomsky sont décidables.

Insuffisances de ce qui précède :

1) La démonstration que nous venons de donner ne conduit pas à des algorithmes rapides ("performants") ; elle n'est donc pas directement applicable.

2) Parmi les chemins de X à α , c'est à dire les dérivations de X à α , nous allons montrer sur les exemples suivant que certains sont essentiellement différents et d'autres non. Or l'étude faite ne permet pas de les distinguer :

a) Soit $L = \{ a^n b^n c^p \mid n \geq 0 , p \geq 0 \}$ muni de la grammaire $G = (N, T, ::= X)$ avec

$X ::= Ac \mid A \mid c$

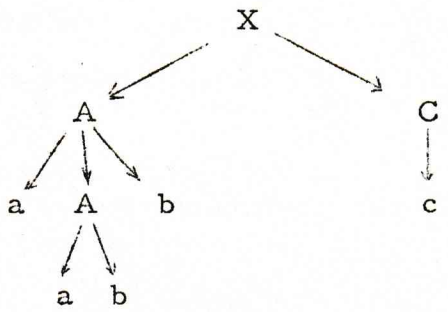
$A ::= aAb \mid ab$

$C ::= cC \mid c$

Le mot $a^2 b^2 c$ peut être engendré par les dérivations suivantes :

X	X	X
A C	A C	A C
aAb C	aAb C	A c
aabb C	aAb c	aAb c
aabb c	aabb c	aabb c

qui sont toutes liées au même "schéma arborescent" :



il est naturel de considérer que ces dérivations ne sont pas essentiellement différentes.

b) Envisageons la phrase française :

La pompiste regarde la fille de la voisine qui achète une voiture .

Elle peut être engendrée pour la grammaire qui suit (les symboles non terminaux sont soulignés ; leur signification est la même qu'en (3.2.3)) :

P ::= GN GV

GN ::= A N | GN de GN | GN SR

A ::= la | une

N ::= pompiste | fille | voisine | voiture

SR ::= R GV

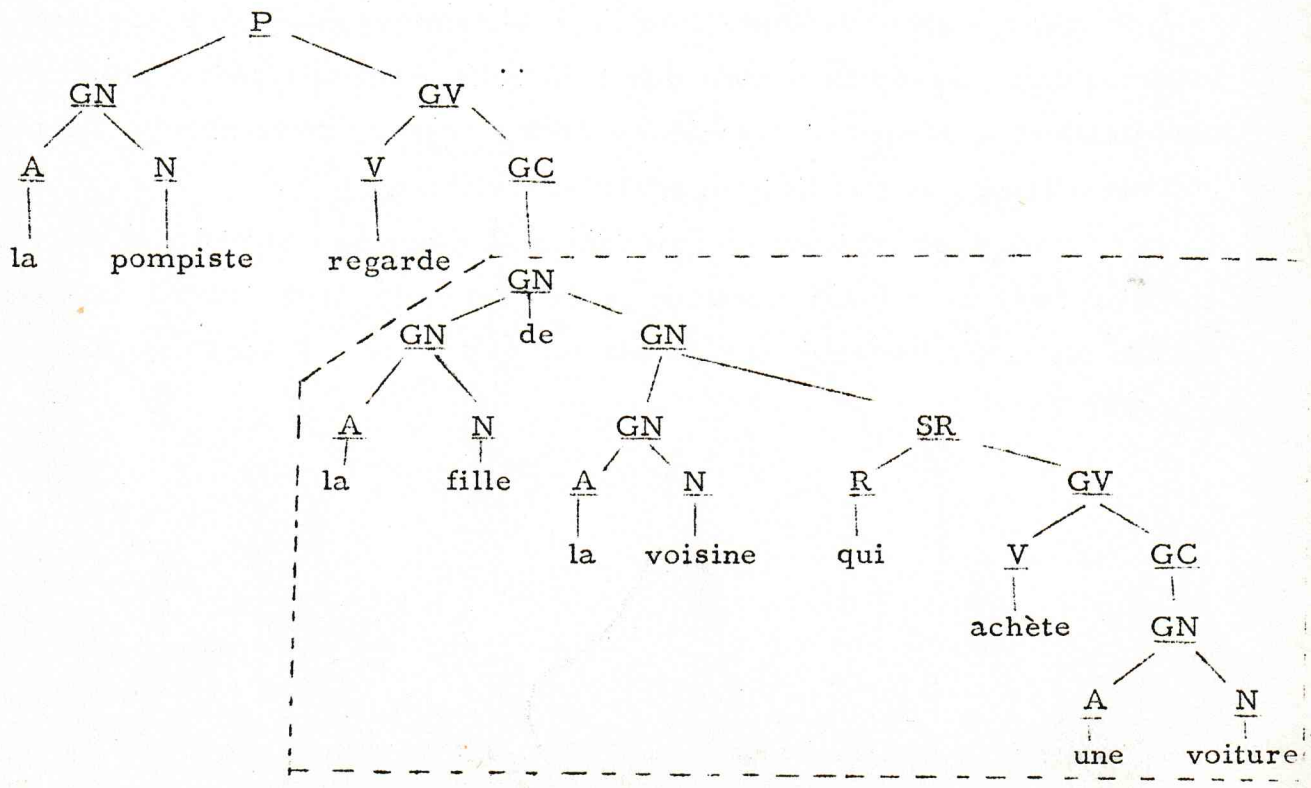
R ::= qui

GV ::= V GC

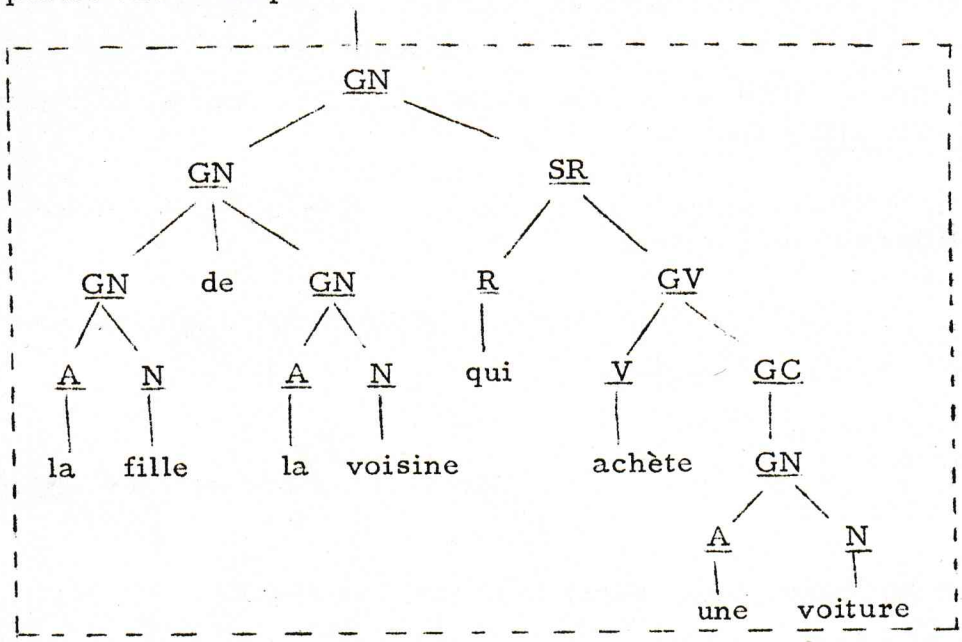
V ::= regarde | achète

GC ::= GN

Une première façon d'engendrer la phrase est d'utiliser les dérivations qui conduisent au schéma arborescent :



Mais en utilisant autrement les règles , nous pouvons remplacer la partie encadrée par :



Il est naturel de considérer les dérivations conduisant aux deux schémas comme essentiellement différentes. La phrase possède d'ailleurs deux significations différentes, associées à deux analyses différentes, selon que c'est la voisine , ou sa fille, qui achète une voiture.

Donc pour analyser un mot dans un langage de Chomsky, il n'est pas suffisant de trouver les dérivations permettant de le construire ; il faut aussi étudier les schémas arborescents liés aux dérivations. C'est l'objet du chapitre 5.

CHAPITRE 4 :
GENERALISATIONS des LANGAGES de CHOMSKY

Pour généraliser les langages de Chomsky on peut élargir la définition des grammaires (*) ; cependant certaines extensions de la notion de grammaire ne modifient pas l'ensemble des langages engendrés ; ainsi en faisant les hypothèses suivantes sur une grammaire G on engendre seulement les langages de Chomsky :

- La grammaire G admet des axiomes qui forment un K-langage (ou un C-langage).

- Les α tels qu'il existe A, $A ::= \alpha$ forment un K-langage ne contenant pas Λ (ou même un C-langage).

en Algol, par exemple, la règle :

(1) <identificateur> ::= <lettre> | <identificateur> <lettre> | <identificateur> <chiffre>

peut être remplacée par :

(2) <identificateur> ::= <lettre> (<lettre> | <chiffre>)*

ce qui signifie qu'à droite on peut avoir n'importe quelle suite de lettres ou de chiffres commençant par une lettre ; cette présentation à l'avantage de rendre mieux compte de la réalité sémantique (Si TRUC est un identificateur ALGOL il n'est jamais considéré, en pratique, comme l'identificateur TRU suivi de C ainsi que le suggère (1)).

Nous allons voir d'autres extensions de la relation de production (toutes les autres définitions demeureront semblables) qui conduisent à des extensions véritables des langages engendrés :

4. 1. Le second membre d'une règle peut être vide.

Soit une grammaire $G = (T, N, ::=, X)$ où les couples tels que $A ::= \varphi$ vérifient $A \in N$ et $\varphi \in V^*$.

Le langage L (G) engendré par G est un langage de Chomsky ou sa réunion avec le mot vide.

(*) Pour éviter toute confusion nous appellerons, dans ce chapitre, "grammaires de Chomsky" les grammaires définies en (3. 2. 1)

Nous allons construire une grammaire de Chomsky G', associée à G et telle que $L(G) = L(G')$ ou $L(G) = L(G') \cup \{\Lambda\}$. Soient $N_0 = \{ A \in N \mid A \overset{*}{\underset{G}{\succ}} \Lambda \}$, $G' = (T, N, ::=', X)$ où la relation $::='$ est définie par :

$$(\forall A \in N,\ \forall \beta_1, \ldots, \beta_q \in V \cup \{\Lambda\}\ )\ \ A ::=' \beta_1 \ldots \beta_q \Leftrightarrow (\exists B_1, \ldots, B_q \in V)$$

$$((\beta_i = B_i \text{ ou } (\beta_i = \Lambda \text{ et } B_i \in N_0) \text{ pour } i = 1, \ldots, q) \text{ et } (\exists i, 1 \leqslant i \leqslant q \ \ \beta_i = B_i)).$$

Ainsi Λ n'est jamais second membre de $::='$ et G' est bien une grammaire de Chomsky. D'après la définition :

$$A ::=' \gamma \Rightarrow A \overset{*}{\underset{G}{\succ}} \gamma$$

à fortiori $\alpha \overset{*}{\underset{G'}{\succ}} \gamma \Rightarrow \alpha \overset{*}{\underset{G}{\succ}} \gamma$ (1)

car $\overset{*}{\underset{G'}{\succ}}$ est le plus petit préordre compatible avec la concaténation, qui contient la relation $::='$.

Réciproquement montrons que :

$$A \overset{*}{\underset{G}{\succ}} \gamma \text{ et } \gamma \neq \Lambda \Longrightarrow A \overset{*}{\underset{G'}{\succ}} \gamma \qquad (2)$$

Procédons par récurrence sur la longueur n de la dérivation ; pour n = 0 (2) est trivialement vrai. Supposons (2) pour des dérivations de longueur inférieure à n et soit γ non vide dérivant de A par une dérivation de longueur n : $(\exists B_1, \ldots, B_q \in V \cup \{\Lambda\}\ )\ (A ::= B_1 \ldots B_q \overset{*}{\underset{G}{\succ}} \gamma)$.

Par suite γ peut s'écrire $\gamma_1 \ldots \gamma_q$ de manière que γ_i dérive de B_i par une dérivation de longueur inférieure ou égale à celle de la dérivation de $B_1 \ldots B_q$ à γ (3. 2. 5) ; donc

$$B_i \overset{*}{\underset{G}{\succ}} \gamma_i \qquad \text{pour } i = 1, \ldots, q$$

et, par hypothèse de récurrence : si $\gamma_i \neq \Lambda$ alors $B_i \overset{*}{\underset{G'}{\succ}} \gamma_i$

si $\gamma_i = \Lambda$, $B_i \in N_0$

et $A ::=' \beta_1 \ldots \beta_q$ avec $\beta_i = B_i$ si $\gamma_i \neq \Lambda$ et $\beta_i = \Lambda$ sinon ; ~~de plus,~~

il existe un γ_i non vide car γ est non vide de sorte que :

$A ::= \beta_1 \ldots \beta_q$

et pour tout β_i non vide $\quad \beta_i \overset{*}{\underset{G'}{\succ}} \gamma_i$

finalement $\quad A \overset{*}{\underset{G'}{\succ}} \gamma \quad$ ce qui prouve (2)

Par conséquent si $X \notin N_0 \qquad L(G) = \{\gamma \mid X \overset{*}{\underset{G}{\succ}} \gamma\}$

$$= \{\gamma \mid X \overset{*}{\underset{G'}{\succ}} \gamma\} = L(G') \quad (1) \text{ et } (2)$$

si $X \in N_0 \qquad L(G) = \{\Lambda\} \cup \{\gamma \neq \Lambda \mid X \overset{*}{\underset{G}{\succ}} \gamma\}$

$$= \{\Lambda\} \cup \{\gamma \mid X \overset{*}{\underset{G'}{\succ}} \gamma\} \qquad (1) \text{ et } (2)$$

$$= \{\Lambda\} \cup L(G')$$

Ce qui achève la démonstration.

4. 2. Règles générales de production (grammaires semi-thueiennes)

Une généralisation beaucoup plus importante consiste à prendre pour couples en relation ::= une famille finie quelconque de $(V^* \setminus \{\Lambda\}) \times V^*$.
On démontre qu'on engendre ainsi des langages indécidables ce qui diminue l'intérêt pratique de cette généralisation. Cependant si les règles sont du type

$\alpha ::= \beta \qquad \alpha \in V^* \setminus \{\Lambda\} \quad , \beta \in V^* \quad$ et $\quad |\alpha| \leqslant |\beta|$

les langages engendrés sont décidables (la démonstration de 3. 5. 2 subsiste) : ce sont les langages contextuels engendrés aussi par les grammaires contextuelles :

Définition

Une grammaire contextuelle (context sensitive grammar) $G = (N, T, ::=, X)$ est une grammaire dont les règles sont du type :

$\alpha' A \alpha'' ::= \alpha' \beta \alpha'' \qquad \alpha', \alpha'', \beta \in V^*, A \in N, \beta \neq \Lambda$

Le nom de ces grammaires rappelle que lorsqu'une règle $\alpha' A \alpha'' ::= \alpha' \beta \alpha''$ est appliquée, A n'est remplacé par β que s'il est environné par le contexte (α', α'').

Les grammaires de Chomsky sont des grammaires contextuelles particulières.

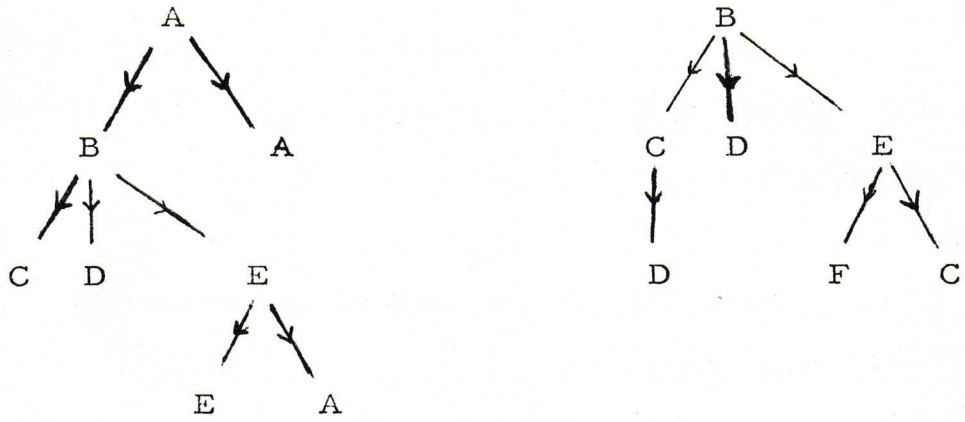
On peut démontrer que le langage $\{a^n b^n c^n\}$ est un langage contextuel ; il en résulte que l'ensemble des langages de Chomsky est strictement inclus dans l'ensemble des langages contextuels.

CHAPITRE 5

RAMIFICATIONS

5. 1. Introduction

Dans de nombreux domaines du traitement de l'information, on emploie des "structures arborescentes", avec éventuellement plusieurs racines, souvent orientées "de gauche à droite" et dont chaque noeud est étiqueté par un nom appartenant à un certain alphabet (figure 1). En particulier, la théorie des langages conduit à considérer des ensembles de telles structures comme de véritables "langages à deux dimensions". L'étude qui suit présente une formalisation permettant de traiter algébriquement ces objets, nommés ramifications.



Ramification

Figure 1

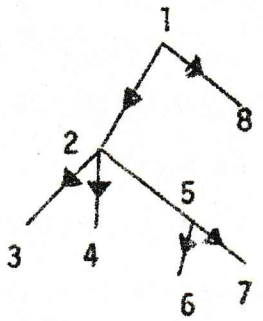
5.1 Définition des ramifications sur un ensemble V :

5.1.1 Arborescence (BERGE, 1958). On appelle arborescence un graphe fini sans circuit tel que :

a) il existe un point a qui n'est extrémité d'aucun arc ;

b) tout point $x \neq a$ est l'extrémité d'un arc unique.

a s'appelle la racine de l'arborescence.



Arborescence de racine 1

Figure 2

5.1.2 Orientation d'une arborescence : Une orientation d'une arborescence (E, Γ)[1] est un ordre partiel 0 dans l'ensemble E tel que :

a) les restrictions de 0 à chacun des ensembles $\Gamma(x)$ $(x \in E)$ sont des ordres totaux ;

b) si $y \in \Gamma(x)$ et $z \notin \Gamma(x)$, y et z ne sont pas comparables par la relation 0.

Un triplet $(E, \Gamma, 0)$ où (E, Γ) est une arborescence et 0 une de ses orientations s'appelle arborescence orientée.

---

[1] Rappelons qu'un graphe est un couple (E, Γ) formé d'un ensemble E (ensemble des points du graphe) et d'une relation binaire Γ dans E, on note $\Gamma(x)$ l'ensemble des $y \in E$ tels que $x \Gamma y$.

Soient deux arborescences orientées $(E, \Gamma, 0)$ et $(E', \Gamma', 0')$; un isomorphisme de la première sur la deuxième est une bijection h de E sur E' telle que

$$(\forall x,y \in E)\ \left[(x\,\Gamma y \Longleftrightarrow h(x)\ \Gamma'\ h(y))\ \text{et}\ (x0y \Longleftrightarrow h(x)\ 0'\ h(y))\right].$$

5.1.3. Pseudo-arborescence sur un ensemble V. Etiqueter les points d'une arborescence par des éléments d'un ensemble V revient à définir une application de l'ensemble des points de l'arborescence dans V. Mais il est souhaitable que deux arborescences orientées isomorphes, dont les points qui se correspondent dans l'isomorphisme ont même étiquette, déterminent la même ramification. Aussi sommes-nous conduits à la définition suivante :

Soit un ensemble V. Dans l'ensemble des couples (I,f) formés par une arborescence orientée I dont les points sont des entiers, et une application f de l'ensemble des points de I dans V, introduisons la relation d'équivalence :

$(I,f) \sim (I',f') \Longleftrightarrow$ il existe un isomorphisme h de I sur I' tel que $f = f' \circ h$.

Une pseudo-arborescence sur V est une classe de cette équivalence.

Elle pourra être représentée par l'un de ses couples (I,f). On appelle ordre de cette pseudo-arborescence le nombre des points de l'arborescence I ; pour chaque $A \in V$, on appelle nombre d'occurences de A dans la pseudo-arborescence le nombre de points x de I tels que $f(x) = A$: l'ordre est la somme des nombres d'occurences des divers éléments de V.

Une pseudo-arborescence d'ordre 1 sera identifiée à l'unique élément de V qui y possède une occurence. Ainsi l'ensemble V est contenu dans l'ensemble des pseudo-arborescences sur lui-même.

L'ensemble des pseudo-arborescences sur V sera noté $\mathcal{A}(V)$.

5.1.4. Ramifications sur un ensemble V : Nous appellerons ramification sur V toute suite finie de pseudo-arborescences sur V. L'ensemble des ramifications sur V, c'est-à-dire le monoïde libre déduit de l'ensemble des pseudo-arborescences sur V, sera noté $\hat{V}$. La loi de composition de ce monoïde sera appelée produit et notée $\longrightarrow$; l'élément neutre de cette loi sera nommée ramification vide et noté $\wedge$

La figure 3 schématise deux ramifications r' et r" et leur produit r' $\longrightarrow$ r".

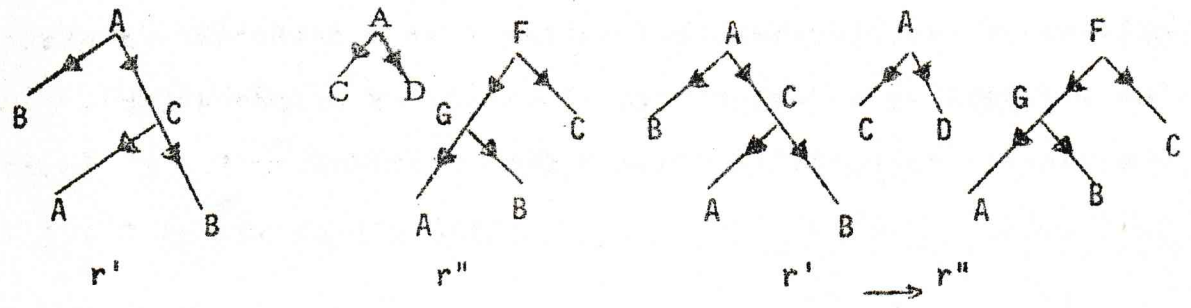


figure 3.

Par définition :

- la longueur d'une ramification est le nombre des pseudo-arborescences qui la constituent ;
- l'ordre d'une ramification est la somme des ordres des pseudo-arborescences qui la constituent ;
- le nombre d'occurences de $A \in V$ dans une ramification est la somme des nombres d'occurences de A dans les pseudo-arborescences qui la constituent.

De même qu'on appelait langage sur V toute partie de V^*, on appellera bilangage sur V toute partie de $\hat{V}$. Comme V est inclus dans $\mathcal{O}(V)$, V^* est inclus dans $\hat{V}$ et tout langage est un bilangage ; en général, nous noterons aussi $\longrightarrow$ la concaténation dans V^*.

La définition des ramifications sur V est relativement complexe. La théorie algébrique qui suit permettra d'engendrer les ramifications de $\hat{V}$ à partir de V grâce au produit et à une loi de composition externe, et ensuite de ne plus guère employer explicitement la définition.

5.2 Etude algébrique de $\hat{V}$:

5.2.1. Enracinement : Ce sera une loi de composition externe à opérateurs dans l'ensemble V, notée $\uparrow$ avec opérateurs à gauche : intuitivement, la ramification $A \uparrow r$ est la pseudo-arborescence obtenue en adjoignant à r une racine d'étiquette A. La figure 4 présente $B \uparrow r'$ et $A \uparrow r''$, r' et r'' étant prises sur la figure 3.

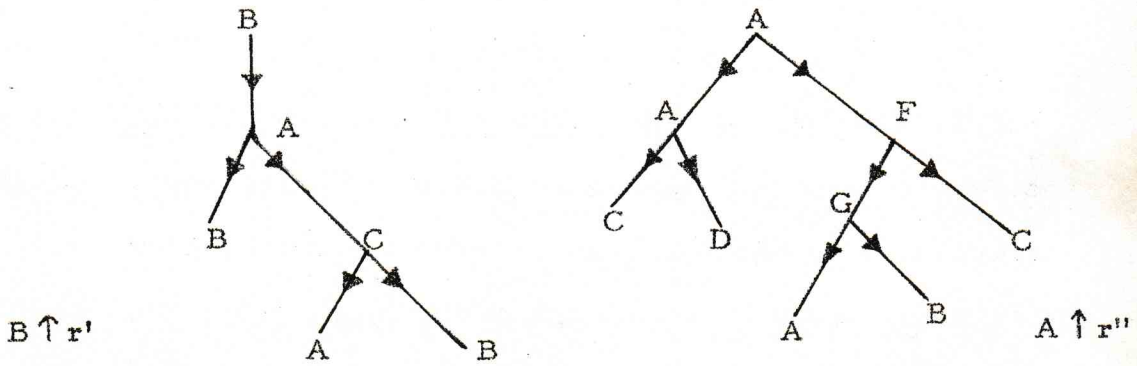


Figure 4

Soit une ramification r sur V, et un élément A de V.

Si r est la ramification vide, $A \uparrow r = A$.

Supposons que r est une suite de pseudo-arborescences $(E_i, \Gamma_i, 0_i, f_i)$, pour $i = 1, 2, \ldots, n$; on peut toujours supposer les ensembles E_i deux à deux disjoints ; désignons par a_i la racine de (E_i, Γ_i). $A \uparrow r$ est la pseudo-arborescence $(E, \Gamma, 0, f)$ définie par :

- E est la réunion des E_i et d'un élément a n'appartenant à aucun E_i ;
- $x \Gamma y \Longleftrightarrow (\exists\, i)\ (x \Gamma_i y)$ ou $(x = a$ et $(\exists i)\ (y = a_i))$;
- $x\ 0\ y \Longleftrightarrow (\exists i)\ (x\ 0_i\ y)$ ou $((\exists\, i, j)\ (i \leqslant j$ et $x = a_i$ et $y = a_j))$
- $f(x) = f_i(x)$ si $x \in E_i$, $f(a) = A$.

On voit aisément que $A \uparrow r$ est bien une pseudo-arborescence et que réciproquement :

5.2.2. Proposition : Pour toute pseudo-arborescence s sur V, il existe un élément A de V et un seul, une ramification r sur V et une seule, tels que $s = A \uparrow r$.

5.2.3. Binoïde sur un ensemble V : On appelle binoïde sur un ensemble V un ensemble muni d'une loi de composition interne associative, admettant un élément neutre et d'une loi de composition externe à opérateurs dans V. $\widehat{V}$ est donc un binoïde sur V.

De même que pour les structures algébriques classiques, on définit un sous-binoïde de $\mathcal{B}$ comme une partie $\mathcal{B}_0$ de $\mathcal{B}$ qui contient e et qui est stable pour $\longrightarrow$ et $\uparrow$ (si $r \in \mathcal{B}_0$, $s \in \mathcal{B}_0$, $A \in V$, alors $r \longrightarrow s \in \mathcal{B}_0$ et $A \uparrow r \in \mathcal{B}_0$).

Dans la suite les deux lois d'un binoïde sur V seront, en général, notées $\longrightarrow$ et $\uparrow$; l'élément neutre sera noté e, sauf pour $\hat{V}$ dont l'élément neutre est $\wedge$.

Si $\mathfrak{B}$ et $\mathfrak{B}'$ sont deux binoïdes sur le même ensemble V, un homomorphisme de binoïdes (ou, simplement, homomorphisme) de $\mathfrak{B}$ dans $\mathfrak{B}'$ est une application ψ de $\mathfrak{B}$ dans $\mathfrak{B}'$, telle que, pour r et s dans $\mathfrak{B}$ et A dans V :

$$\psi(r \longrightarrow s) = \psi(r) \longrightarrow \psi(s) \; ; \; \psi(e) = e \; ; \; \psi(A \uparrow r) = A \uparrow \psi(r).$$

5.3 Propriétés fondamentales de $\hat{V}$: De la définition des ramifications comme suites de pseudo-arborescences et de la proposition 2.2.2, il résulte immédiatement :

2.3.1 Proposition : Pour tout $r \in \hat{V}$, non vide, il existe $A \in V$, $r' \in \hat{V}$, $r'' \in \hat{V}$ uniques tels que

$$r = r' \longrightarrow (A \uparrow r'')$$

On en déduit un principe de récurrence dans $\hat{V}$:

5.3.2 Principe de récurrence : Soit $\mathcal{P}$ un prédicat tel que

a) $\mathcal{P}(\wedge)$ soit vrai

b) $(\forall r \in \hat{V},\ a \in \mathcal{A}(V))\ (\mathcal{P}(r)$ et $\mathcal{P}(a) \Longrightarrow \mathcal{P}(r \longrightarrow a))$

c) $(\forall r \in \hat{V})\ (\forall A \in V)\ (\mathcal{P}(r) \Longrightarrow \mathcal{P}(A \uparrow r))$;

alors, $\mathcal{P}(r)$ est vrai pour tout $r \in \hat{V}$.

On montre que $\mathcal{P}(r)$ est vrai en raisonnant par récurrence sur l'ordre de la ramification r, grâce à la proposition 2.3.1.

Soit $\mathfrak{B}$ un sous-binoïde de $\hat{V}$; en utilisant le principe de récurrence, on voit que toute ramification sur V appartient à $\mathfrak{B}$, autrement dit que $\mathfrak{B} = \hat{V}$: $\hat{V}$ ne contient aucun sous-binoïde autre que lui-même. En particulier, l'ensemble des combinaisons

finies, par $\longrightarrow$ et $\uparrow$, d'éléments de V, qui est un sous-binoïde de $\hat{V}$, est égal à $\hat{V}$: <u>toute ramification sur V est une combinaison finie par $\longrightarrow$ et $\uparrow$, d'éléments de V</u>.

<u>Exemples</u> : Pour les ramifications de la figure 3 :

$r' = A \uparrow (B \longrightarrow (C \uparrow (A \longrightarrow B)))$

$r'' = (A \uparrow (C \longrightarrow D)) \longrightarrow (F \uparrow ((G \uparrow (A \longrightarrow B)) \longrightarrow C))$.

5.3.3. Les applications de $\hat{V}$ dans un ensemble E seront commodément définies par récurrence. Plus précisément :

<u>Théorème</u> : Soient un ensemble E, un élément e de E, deux applications f_1, de $\hat{V} \times \mathcal{A}(V) \times E^2$ dans E et f_2 de $V \times \hat{V} \times E$ dans E. Il existe une application λ et une seule, de $\hat{V}$ dans E, telle que :

a) $\lambda(\wedge) = e$

b) $(\forall r \in \hat{V})\ (\forall a \in \mathcal{A}(V))\ [\lambda(r \longrightarrow a) = f_1(r, a, \lambda(r), \lambda(a))]$

c) $(\forall A \in V)\ (\forall r \in \hat{V})\ [\lambda(A \uparrow r) = f_1(A, r, \lambda(r))]$.

On en déduit aisément :

5.3.4. <u>Théorème</u> : Pour tout binoïde $\mathcal{B}$ sur V, il existe un homomorphisme de binoïdes et un seul de $\hat{V}$ dans $\mathcal{B}$.

Ce résultat conduit à nommer $\hat{V}$ <u>binoïde libre sur V</u>.

Nous définirons maintenant par récurrence quelques applications simples.

5.3.5. <u>Mot des racines d'une ramification</u> : Appelons <u>mot des racines de $r \in \hat{V}$</u> le mot $\rho(r)$ sur le vocabulaire V défini par :

$\rho(\wedge) = \wedge$; $\rho(r \longrightarrow s) = \rho(r) \longrightarrow \rho(s)$; $\rho(A \uparrow r) = A$

Si le mot des racines de r est $A_1 \ldots A_n$, A_i est appelée la <u>$i^{ème}$ racine de r</u>.

Par exemple, le mot des racines de la ramification $r' \longrightarrow r''$, prise dans la figure 3, est AAF.

5.3.6. Proposition : Soient α' et α'' deux mots sur V et une ramification r dont le mot des racines est $\alpha' \longrightarrow \alpha''$ il existe deux ramifications r' et r", chacune unique, telles que $\rho(r') = \alpha'$, $\rho(r'') = \alpha''$ et $r = r' \longrightarrow r''$.
(Ce résultat se démontre par récurrence sur la longueur de α'').

2.3.7. Mot des feuilles d'une ramification : Appelons mot des feuilles de r le mot $\varphi(r)$ sur V tel que :

$$\varphi(\wedge) = \wedge ; \varphi(r \longrightarrow s) = \varphi(r) \longrightarrow \varphi(s) ;$$

$$\varphi(A \uparrow r) = \text{Si } \varphi(r) \neq \Lambda \text{ alors } \varphi(r) \text{ sinon } A.$$

Si le mot des feuilles de r est $A_1 \ldots A_n$, on dit que A_i est la $i^{ème}$ feuille de r.

Sur la figure 3, par exemple : $\varphi(r' \longrightarrow r'') = BABCDABC$.

5.3.8. Familles d'une ramification : Pour la ramification $r' \longrightarrow r''$ de la figure 3 nous dirons que le mot CD est une famille de prédécesseur A, que le mot vide est une famille de prédécesseur D.

Soit A un élément de V et F_A l'application de $\hat{V}$ dans l'ensemble $\mathcal{F}(V^*)$ des parties de V^* définie par :

$$F_A(\wedge) = \emptyset ; F_A(r \longrightarrow s) = F_A(r) \cup F_A(s) ;$$

$$F_A(B \uparrow r) = \text{Si } B = A \text{ alors } F_A(r) \cup \{\rho(r)\} \text{ sinon } F_A(r).$$

Par définition $F_A(r)$ est l'ensemble des familles de prédécesseur A dans r.

Sur la figure 3, par exemple, $F_A(r' \longrightarrow r'') = \{B \longrightarrow C, C \longrightarrow D, \wedge\}$.

φ, ρ et F_A garderons la même signification dans tout ce qui suit .

5. 5. Application à l'analyse syntaxique

5. 5. 1 Définition

Soit G = (T, N, ::=, X) une grammaire ; une ramification r est engendrée au sens large par la grammaire G si :

a) Chacune de ses familles non vides α de prédecesseur A vérifie A ::= α.

b) $\varphi(r) \in T^*$

Une ramification r de l'ensemble $\mathcal{L}(G)$ des ramifications engendrées au sens large par G est dite engendrée au sens strict (ou, simplement, engendrée) si $\rho(r)$ est X. L'ensemble des ramifications engendrées au sens strict par G est noté $\mathcal{S}(G)$.

Les ramifications de $\mathcal{S}(G)$ sont donc des pseudo-arborescences.

5. 5. 2 Théorème

Le langage engendré par une grammaire G est l'ensemble des mots des feuilles des pseudo-arborescences engendrées par G.

Nous allons montrer, en fait, la propriété suivante, qui se prête mieux au raisonnement par récurrence, et dont découle immédiatement le théorème :

$$((\forall \alpha \in V^*,\ \beta \in T^*)\ \alpha \overset{*}{\longrightarrow} \beta) \Longleftrightarrow (\exists\, r \in \mathcal{L}(G))\ (\rho(r) = \alpha \text{ et } \varphi(r) = \beta)$$

Supposons qu'il existe une dérivation de α à β ; soit n sa longueur.

Pour n = 0, si $\alpha = \beta = \Lambda$ alors $r = \Lambda$ répond à la question, si $\alpha = \beta = B_1 \ldots B_q$ $(B_i \in T)$ il suffit de choisir $r = B_1 \rightarrow \ldots \rightarrow B_q$.

Admettons la propriété pour toute dérivation de longueur n - 1. Une dérivation de longueur n de α à β s'écrit :

$\alpha = \lambda A \lambda' \longrightarrow \lambda \gamma \lambda' \overset{*}{\longrightarrow} \beta$ où $A ::= \gamma$ et où une dérivation de longueur n - 1 fait passer de $\lambda \gamma \lambda'$ à β.

Par hypothèse de récurrence il existe une ramification r telle que

$$\rho(r) = \lambda\gamma\lambda'$$
$$\varphi(r) = \beta$$

d'après la proposition 5.3,6, r s'écrit :

$r = r_1 \longrightarrow r_2 \longrightarrow r_3$ avec $\rho(r_1) = \lambda$, $\rho(r_2) = \gamma$ $\rho(r_3) = \lambda'$.

Soit $r' = r_1 \longrightarrow (A\uparrow r_2) \longrightarrow r_3$

Alors $\rho(r') = \rho(r_1)\ \rho(A\uparrow r_2)\ \rho(r_3) = \lambda A \lambda' = \alpha$

$$\varphi(r') = \varphi(r_1)\ \varphi(r_2)\ \varphi(r_3) = \varphi(r) = \beta$$

de plus r' est engendrée au sens large par G car :

a) les familles de r' sont celles de r et $\rho(r_2) = \gamma$, famille de prédécesseur A qui vérifie $A ::= \gamma$.

b) $\varphi(r') = \beta$ est élément de T^*.

Envisageons la réciproque en appliquant le principe de récurrence sur pour $r = \Lambda$ alors $\rho(r) = \varphi(r) = \Lambda$ et la propriété est vraie.

Pour $r = r' \longrightarrow r''$: $\alpha = \rho(r) = \rho(r')\ \rho(r'')$

$$\beta = \varphi(r) = \varphi(r')\ \varphi(r'')$$

par hypothèse de récurrence : $\rho(r') \overset{*}{\rightarrowtail} \varphi(r')$ et $\rho(r'') \overset{*}{\rightarrowtail} \varphi(r'')$

donc, par concaténation $\alpha \overset{*}{\rightarrowtail} \beta$.

Pour $r = A\uparrow r'$, si r' est vide $\varphi(r) = \rho(r)$, donc $\rho(r) \overset{*}{\rightarrowtail} \varphi(r)$

si r' n'est pas vide $\varphi(r) = \varphi(r')$; par hypothèse :

$\rho(r') \overset{*}{\rightarrowtail} \varphi(r')$ et $r \in \mathcal{L}(G)$ donc $A ::= \rho(r')$, par suite :

$A \overset{*}{\rightarrowtail} \varphi(r')$ ce qui achève la démonstration.

5. 5. 3 Le problème de l'analyse syntaxique

Soit une grammaire G et T son vocabulaire terminal. Effectuer l'analyse syntaxique d'un mot α sur T pour la grammaire G, c'est trouver l'ensemble des ramifications de $\mathcal{S}(G)$ dont α est le mot des feuilles, autrement dit l'ensemble $\varphi^{-1}(\alpha) \cap \mathcal{S}(G)$.

Cet ensemble est non vide si, et seulement si, α appartient au langage L (G) engendré par G. S'il existe un mot α pour lequel cet ensemble contient plus d'un élément, on dit que la grammaire G est ambiguë, et que α est ambigu pour G.

L'ambiguité d'un mot α dépend donc de la grammaire qui l'engendre ; toutefois il existe des langages dont toute grammaire est ambiguë ; c'est le cas, par exemple, du langage de Chomsky

$$L = \left\{ a^n b^n a^p \mid n>0, p>0 \right\} \cup \left\{ a^p b^n a^n \mid n>0, p>0 \right\} \quad \text{sur } V = \left\{ a,b \right\}.$$

On montre, en effet, que, quelle que soit la grammaire G engendrant les mots $a^n b^n c^n$ sont ambigus pour G.

ERRATA

PAGE	LIGNE	TEXTE	CORRECTION
4	22	α	a
19	8	(2.4.5)	(2.5.5)
22	6	miroir	réflechi
23	11-13-14	V	$\forall$
24	7	$\int_B^{M'}$	$\int_M^{M'}$
28	3	(2.8.4)	(2.8.5)
32	10	$= \{((s, a), \ldots$	$\tau = \{((s, a), \ldots$
37	19	Enfin les inégalités :	Enfin i et α peuvent être choisis de façon que : ordre $[\Phi(\mathfrak{E})\Delta\Phi(\mathfrak{P})]$ = ordre $[E'_i \Delta F'_i] = \lvert\alpha\rvert \ldots$
41	10		supprimer "et elles seules"
45	2		ajouter et $\alpha = \beta_1\beta_2\ldots\beta_{qj}$
51	20	$A::\tilde{=}\varphi \quad A::=\tilde{\varphi}$	$A::\tilde{=}\varphi \iff A ::= \tilde{\varphi}$
62	8	$n > 0$	$n \geqslant 0$
	10	X::= Ac |A| c	X::= AC |A| c
66	5		ajouter : et $(A::=B_1 \ldots B_q)$
74-75			inverser l'ordre de la dernière phrase p. 74 et de la première p. 75.
75	18	$\Rightarrow (r \rightarrow a)$	$= \mathfrak{F}(r \rightarrow a)$
78	4	φ (r) T^*	φ (r) ϵ T^*

ERRATA

PAGE	LIGNE	
4	22	$\alpha \longrightarrow a$
19	8	(2.4.5) $\longrightarrow$ (2.5.5)
22	6	miroir $\longrightarrow$ réfléchi.
23	10.11-14	$V \longrightarrow \forall$
24	8	$\begin{cases} M' \\ B \end{cases} \longrightarrow \begin{cases} M' \\ M \end{cases}$
25	avt dernière	$D_{t(K)}(\alpha) \longrightarrow D_{t(K)}(\alpha')$
28	3	(2.8.4) $\longrightarrow$ (2.8.5)
32		endessous de F= .... c'est : $\gamma = \{((o,a), \ldots$
45	2	rajouter : et $\alpha = \beta_1 \beta_2 \ldots \beta_q$;
51	19	$A ::\tilde{=} \varphi \quad A ::= \tilde{\varphi} \longrightarrow A ::\tilde{=} \varphi \iff A ::= \tilde{\varphi}$
54	6 et 8	s'a $\longrightarrow$ s'. a
62	8	$n>0 \quad p>0 \longrightarrow n \geqslant 0 \quad p \geqslant 0$
	9	$X ::= Ac \mid A \mid c \longrightarrow X ::= AC \mid A \mid c$
66	5	rajouter : et ($A ::= B_1 \ldots B_q$)