LANGAGES "CONTEXT-SENSITIVES"

QUELQUES ASPECTS DE LEUR EXTENSION.

par J. FRIANT

SOMMAIRE.

Ce rapport rappelle les principales définitions concernant les grammaires de constituants, notamment les grammaires CS. Puis il introduit la notion de grammaires CS de reconnaissance, ce qui permet en particulier de démontrer simplement la stabilité de l'intersection relativement à la classe des langages CS. Plusieurs langages CS, de nature arithmétique, sont ensuite décrits; ainsi il est montré que l'ensemble des nombres premiers constitue un tel langage. En conclusion, l'énoncé de deux résultats permet d'inclure la classe des langages CF matricielles et celle des langages CFP dans la classe des langages CS.

§ 0. RAPPELS. NOTATIONS. [4]

0.1. Grammaires de constituants immédiats.

Ce sont des sytèmes formels dont le but est de décrire la construction des phrases de langages artificiels par modifications successives portant sur des suites de symboles. Formellement, une grammaire de constituants est la donnée d'un quadruplet:;

$$\mathcal{G} = (V, V_T, S, \mathcal{R})$$

- $\mathcal{V}$, appelé vocabulaire, est un ensemble fini de symboles;
- $\mathcal{V}_T$, vocabulaire terminal, est un sous-ensemble de $\mathcal{V}$, dont le complémentaire, $\mathcal{V}_N$, est appelé vocabulaire non-terminal;
- S est un symbole distingué de $\mathcal{V}_N$, désigné symbole initial.

Les symboles — éléments de $\mathcal{V}$ — seront représentés par des capitales latines habituellement indicées, et les suites finies de tels symboles, ou mots — éléments du monoïde libre $\mathcal{V}^*$ construit sur $\mathcal{V}$ — par des capitales latines surmontées d'un accent circonflexe.

A un couple ordonné de mots $(\hat{A},\hat{B})$ on associera le mot $\hat{C}$ obtenu en écrivant le mot $\hat{B}$ à droite du mot $\hat{A}$.

Formellement:

si $\hat{A} = A_1 \dots A_i \dots A_n$

$\hat{B} = B_1 \dots B_j \dots B_m$

alors $\hat{C} = \hat{A}\,\hat{B} = A_1 \dots A_n\, B_1 \dots B_m$.

On dira que $\hat{A}$, $\hat{B}$ et $\hat{C}$ sont des mots de longueur respectivement égale à n, m et n + m, et on écrira:

$|\hat{A}| = n$, $|\hat{B}| = m$, $|\hat{C}| = n + m$.

- $\mathcal{R}$ est un ensemble fini dont les éléments, appelés règles de production, sont des couples ordonnés de mots. Ainsi:

$r \in \mathcal{R}$: $r = (\hat{A},\hat{B})$ avec la restriction $\hat{A} \notin \mathcal{V}_T^*$.

0.2. Dérivation;

. On dira que $\hat{Y}$ dérive directement de $\hat{X}$, selon $\mathcal{G}$, par la règle de production $r = (\hat{A},\hat{B})$, s'il existe deux mots $\hat{G}$ et $\hat{D}$ tels que:

$\hat{X} = \hat{G}\,\hat{A}\,\hat{D}$, $\hat{Y} = \hat{G}\,\hat{B}\,\hat{D}$.

On écrira : $\hat{X} \xrightarrow[\mathcal{G}]{\hat{G},\, r,\, \hat{D}} \hat{Y}$

ou simplement: $\hat{X} \xrightarrow[\mathcal{G}]{} \hat{Y}$

On dira que $\hat{Y}$ dérive de X, selon $\mathcal{G}$, s'il existe une suite finie de mots $\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_n$ tels que :

$\hat{X}_0 = \hat{X}$, $\hat{X}_n = \hat{Y}$

et $\forall i,\ 1 \leq i \leq n \quad \exists r_i \in \mathcal{R}: \quad \hat{X}_{i-1} \xrightarrow[\mathcal{G}]{r_i} \hat{X}_i$

On écrira: $\hat{X} \xRightarrow[\mathcal{G}]{\hat{r}} \hat{Y}$ avec $\hat{r} = r_1 \ldots r_i \ldots r_n \in \mathcal{R}^*$

0.3. Langage.

Le langage engendré par la grammaire $\mathcal{G}$ est le sous-ensemble de $\mathcal{V}_T^*$ formé des mots dérivant de S, selon $\mathcal{G}$; on le notera $\mathcal{L}(\mathcal{G})$:

$$\mathcal{L}(\mathcal{G}) = \left\{ \hat{X} \mid \hat{X} \in \mathcal{V}_T^* ;\ \exists \hat{r} \in \mathcal{L}(\mathcal{G}) : S \xRightarrow[\mathcal{G}]{\hat{r}} \hat{X} \right\}$$

Deux grammaires seront dites équivalentes si elles engendrent le même langage.

0.4. Grammaires et langages context-sensitive.

Un sous-ensemble de $\mathcal{V}_T^*$ sera appelé langage context-sensitive s'il peut être engendré par une grammaire $\mathcal{G}$ dont toutes les règles sont telles que la longueur du premier membre est au plus égale à celle du second, i.e. :

$$\forall r \in \mathcal{R},\quad r = (\hat{A}, \hat{B}) : |\hat{A}| \leq |\hat{B}|$$

On remarquera que tous les langages context-free (CF) sont context-sensitive (CS).

On démontre ([7], [4]) qu'un tel langage $\mathcal{L}$ peut toujours être engendré par une grammaire

$$\mathcal{G} = (\mathcal{V}, \mathcal{V}_T, S, \mathcal{R})$$

satisfaisant aux conditions suivantes:

. Outre le symbole initial S, $\mathcal{V}_N$ contient deux symboles particuliers S' et W; on note: $\mathcal{V}' = \mathcal{V}_N - \{ S \}$.

. Les règles se répartissent en trois groupes:

- les règles initiales: (S,SW) et (S,S')
- l'ensemble $\mathcal{R}'$ des règles binaires, de la forme: $(A_i\ A_j\ ,\ A_k\ A_l)$ aucun de ces symboles n'étant terminal,
- et l'ensemble $\mathcal{C}$ des règles terminales, de la forme:

$(A,T)\ :\ A \in \mathcal{V}'\ ,\ T \in \mathcal{V}_T$

Soit $\hat{X} \in \mathcal{V}_T^*\ ,\ |\hat{X}| = n$; il résulte des conditions ci-dessus que $\hat{X}$ est une phrase de $\mathcal{L}$ si et seulement si on peut dériver $\hat{X}$ de $S'\ W^{n-1}$ à l'aide des règles de $\mathcal{R}'$ et de $\mathcal{C}$; plus précisément: $\hat{X} \in \mathcal{L}$ si et seulement si il existe $\hat{A} \in \mathcal{V}'^n$ tel que $\hat{A}$ dérive de $S'\ W^{n-1}$ par $\mathcal{R}'$ et $\hat{X}$ dérive de $\hat{A}$ par $\mathcal{C}$.

§ 1. GRAMMAIRE DE RECONNAISSANCE ASSOCIÉE À UN LANGAGE CS.

1.1. Soit $\mathcal{L}$ un langage CS engendré par une grammaire $\mathcal{G}$ du type rappelé ci-dessus. Nous allons définir une grammaire $\mathcal{G}_r$ équivalente à $\mathcal{G}$:

$\mathcal{G}_r = (\mathcal{V}_r\ ,\ \mathcal{V}_T\ ,\ S\ ,\ \mathcal{R}_r)$.

- Le symbole initial et les symboles terminaux sont ceux de $\mathcal{G}$. On trouve de plus les éléments de

$\mathcal{V}_r' = \mathcal{V}' \times \mathcal{V}_T$

représentés par: $\left[\frac{A}{T}\right]\ A \in \mathcal{V}'\ ,\ T \in \mathcal{V}_T$

- Les règles de $\mathcal{G}_r$ sont déduites de celles de $\mathcal{G}$:

. règles initiales:

$\forall\ T \in \mathcal{V}_T\ :\ (S,\ S\left[\frac{W}{T}\right])$ et $(S,\ \left[\frac{S'}{T}\right])$

. règles binaires, $\mathcal{R}'_r$:

$$\forall T_1, T_2 \in \mathcal{V}_T : \left(\left[\frac{A_i}{T_1}\right]\left[\frac{A_j}{T_2}\right], \left[\frac{A_k}{T_1}\right]\left[\frac{A_l}{T_2}\right]\right) \in \mathcal{R}'_r$$

si et seulement si $(A_i A_j , A_k A_l) \in \mathcal{R}'$

. règles terminales, $\mathcal{C}_r$:

$$\forall T \in \mathcal{V}_T : \left(\left[\frac{A}{T}\right], T\right) \in \mathcal{C}_r \text{ si et seulement si } (A,T) \in \mathcal{C}.$$

Par extension si $\hat{A} = A_1 \dots A_i \dots A_n \in \mathcal{V}'^n$

et $\hat{X} = T_1 \dots T_i \dots T_n \in \mathcal{V}_T^n$

on écrira: $$\left[\frac{\hat{A}}{\hat{X}}\right] = \left[\frac{A_1}{T_1}\right] \dots \left[\frac{A_i}{T_i}\right] \dots \left[\frac{A_n}{T_n}\right]$$

Comme pour la grammaire $\mathcal{G}$, il résulte du choix des règles de $\mathcal{G}_r$ que $\hat{X} \in \mathcal{L}(\mathcal{G}_r)$ si et seulement si on peut dériver $\hat{X}$ de $\left[\frac{S' W^{n-1}}{\hat{X}}\right]$ à l'aide des règles de $\mathcal{R}'_r$ et de $\mathcal{C}_r$, plus précisément $\hat{X} \in \mathcal{L}(\mathcal{G}_r)$ si et seulement si il existe $\hat{A} \in \mathcal{V}'^n$ tel que $\left[\frac{\hat{A}}{\hat{X}}\right]$ dérive de $\left[\frac{S' W^{n-1}}{\hat{X}}\right]$ par $\mathcal{R}'_r$ et $\hat{X}$ dérive de $\left[\frac{\hat{A}}{\hat{X}}\right]$ par $\mathcal{C}_r$. Ainsi à toute S-dérivation de $\hat{X}$, selon $\mathcal{G}$, décomposable en (cf. ci-dessus) :

$$S \xrightarrow[\mathcal{G}]{} S' W^{n-1} \xrightarrow[\mathcal{G}]{(\mathcal{R}')} \hat{A} \xrightarrow[\mathcal{G}]{(\mathcal{C})} \hat{X}$$

correspond la S-dérivation de $\hat{X}$ selon $\mathcal{G}_r$:

$$S \xrightarrow[\mathcal{G}_r]{} \left[\frac{S' W^{n-1}}{\hat{X}}\right] \xrightarrow[\mathcal{G}_r]{(\mathcal{R}'_r)} \left[\frac{\hat{A}}{\hat{X}}\right] \xrightarrow[(\mathcal{G}_r)]{(\mathcal{C}_r)} \hat{X}$$

et réciproquement.

On a donc bien le résultat annoncé:

$$\mathcal{L}(\mathcal{G}_r) = \mathcal{L}(\mathcal{G}) = \mathcal{L}.$$

La grammaire $\mathcal{G}_r$ est dite de reconnaissance car si $\hat{X}$ est un mot terminal ($\hat{X} \in \mathcal{V}^*_T$, et $|\hat{X}| = n$) nous pouvons toujours envisager

une S-dérivation, selon $\mathcal{G}_r$, telle que l'application des règles initiales aboutisse au mot $\left[\frac{S' W^{n-1}}{\hat{X}} \right]$; mais cette dérivation ne sera fructueuse que si $\hat{X}$ est une phrase du langage $\mathcal{L}$, c'est-à-dire si la grammaire $\mathcal{G}_r$, par ses règles $\mathcal{R}'_r$ et $\mathcal{T}_r$, le "reconnaît" telle. On dira indifféremment que $\mathcal{G}$ engendre ou accepte $\hat{X}$.

N. B. : Comme dans le cas des grammaires CS ordinaires ([8],,[4]) on peut aussi envisager des grammaires CS de reconnaissance avec marquant d'extrémités de phrases, noté #. L'introduction d'un tel marquant ne modifie pas la capacité générative de nos grammaires.

1.2. Application: Intersection de langages CS:

En utilisant la notion de grammaire de reconnaissance nous allons prouver le résultat suivant ([8], [4]) :

. L'intersection de deux langages CS est CS.

Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages CS sur le même vocabulaire terminal $\mathcal{V}_T$. On suppose donné pour $\mathcal{L}_i$ (i = 1, 2) une grammaire $\mathcal{G}_i$ du type indiqué en 0.4. ; on note S_i le symbole initial, S'_i , W_i les symboles spéciaux à $\mathcal{G}_i$. D'après le paragraphe 1.1. on peut définir une grammaire de reconnaissance $\mathcal{G}_{r_i}$ équivalente à $\mathcal{G}_i$ dont l'ensemble des symboles non-terminaux, mis à part S_i , est noté $\mathcal{V}'_{r_i}$ et l'ensemble des règles binaires (resp. terminales) $\mathcal{R}'_{r_i}$ (resp. $\mathcal{T}_{r_i}$).

On va construire une grammaire CS de reconnaissance, avec marquant

$$\mathcal{G} = (\mathcal{V} , \mathcal{V}_T , S , \# , \mathcal{R})$$

qui engendre $\mathcal{L}_1 \cap \mathcal{L}_2 = \{ \hat{X} \mid \hat{X} \in \mathcal{L}_1 \text{ et } \hat{X} \in \mathcal{L}_2 \}$.

- Vocabulaire: Le vocabulaire terminal de $\mathcal{G}$ est $\mathcal{V}_T$; le vocabulaire non-terminal $\mathcal{V}_N$ comporte en plus des symboles distingués S et #, les symboles de $\mathcal{V}'_{r_i}$ (i = 1, 2), i.e.:

$$\mathcal{V}_N = \{ S, \# \} \cup \mathcal{V}'_{r_1} \cup \mathcal{V}'_{r_2} .$$

- Principe du choix des règles de $\mathcal{G}$

Nous voulons définir une grammaire de reconnaissance qui n'accepte que les phrases appartenant simultanément à $\mathcal{L}_1$ et $\mathcal{L}_2$. Etant donné un mot terminal $\hat{X}$, nous vérifierons donc d'abord, à l'aide des règles de $\mathcal{R}'_{r_1}$, que $\hat{X}$ est une phrase de $\mathcal{L}_1$; si c'est le cas, et alors seulement, par des règles intermédiaires nous passerons à l'étude de la reconnaissance de $\hat{X}$ par la grammaire $\mathcal{G}_{r_2}$, à l'aide des règles de $\mathcal{R}'_{r_2}$. Et les règles de $\mathcal{T}_{r_2}$ nous permettront de passer au niveau terminal X, si et seulement si $\hat{X}$ appartient à l'intersection $\mathcal{L}_1 \cap \mathcal{L}_2$.

- Règles de $\mathcal{G}$:

. règles initiales

$\forall T \in \mathcal{V}_T$: $\left(S, S \left[\frac{W_1}{T} \right] \right.$ et $\left. \left(S, \left[\frac{S_1}{T} \right] \right) \right.$

. règles de "reconnaissance" de $\mathcal{G}_{r_1}$: $\mathcal{R}'_{r_1}$.

. règles intermédiaires

$\forall T, T' \in \mathcal{V}_T$.

$$\left.\begin{array}{l} \cdot \left(\left[\frac{A}{T} \right] \#, \left[\frac{W_2}{T} \right] \# \right) \\ \cdot \left(\left[\frac{A}{T} \right] \left[\frac{W_2}{T'} \right], \left[\frac{W_2}{T} \right] \left[\frac{W_2}{T'} \right] \right) \\ \cdot \left(\# \left[\frac{W_2}{T} \right], \# \left[\frac{S_2}{T} \right] \right) \end{array}\right\} \text{ si et seulement si } \left(\left[\frac{A}{T} \right], T \right) \in \mathcal{T}_{r_1}$$

. règles $\mathcal{R}'_{r_2}$ et $\mathcal{T}_{r_2}$ de la grammaire $\mathcal{G}_{r_2}$.

Le lecteur vérifiera facilement que le principe signalé plus haut se trouve respecté par le choix de ces règles. $\mathcal{G}$ accepte les phrases $\hat{X}$

appartenant simultanément à $\mathcal{L}_1$ et $\mathcal{L}_2$, et elles seules, ce qui démontre le résultat énoncé.

Nous allons présenter maintenant quelques exemples de langages CS.

§ 2. LANGAGES CS ARTIFICIELS.

Les langages étudiés dans ce paragraphe, sont définis sur le vocabulaire terminal $\mathcal{V}_T = \{ | \}$, réduit à un seul symbole. Une phrase, I^k, d'un tel langage, est entièrement caractérisée par sa longueur k, aussi, par abus de langage, dirons-nous parfois "le nombre k", pour le "mot I^k ". On sait qu'un tel langage infini n'est CF que s'il contient un sous-ensemble "formant" une progression géométrique. J. P. Benzécri ([1] , [2]) a généralisé la notion de langage CF en faisant intervenir des symboles et des mots à plusieurs insertions, et diverses opérations sur ces mots, en plus du produit de juxtaposition on parle de langage CFP. R. Guedj a montré [6] que tout langage CFP infini, sur le vocabulaire terminal $\{ | \}$, contient une progression arithmétique ou une progression de la forme: $\{ | ^{w + u\, v^k} \mid k \in \mathbb{N} \}$. Ainsi le lecteur pourra vérifier facilement qu'aucun des langages étudiés dans ce paragraphe n'est CFP, a fortiori CF.

2.1. 1^er exemple:

$$\mathcal{L}(i,j) = \{ | ^{i^n + j^n} \mid , n \in \mathbb{N} \}$$

i et j étant deux entiers distincts supérieurs à 1..

Soit $\mathcal{G}(i,j)$ la grammaire CS avec marquant

$$\mathcal{G}(i,j) = (\mathcal{V}(i,j) , \mathcal{V}_T , S , \# , \mathcal{R}(i,j))$$

où: - $\mathcal{V}(i,j) = \{ S, \#, A_1 , A_2 , B_1 , B_2 , | \}$

- $\mathcal{R}(i,j)$ comporte :

. la règle initiale (S, A_1B_1)

. les règles de production

$(\# A_1 , \# A_2)$ (début)

$(A_2A_1 , A_1^i A_2)$; $(A_2B_1 , A_1^i B_2)$; $(B_2B_1 , B_1^j B_2)$

$(B_2 \# , B_1^j \#)$ (fin) .

. et les règles terminales $(A_1 , |)$; $(B_1 , |)$.

Le lecteur vérifiera facilement que

$$\mathcal{L}(\mathcal{G}(i,j)) = \mathcal{L}(i,j)$$

Ce résultat serait évidemment susceptible de généralisation en considérant plus de deux indices (i,j).

2.2. 2ème exemple:

$$\mathcal{L}(!) = \{ |^{n!} \mid n \in \mathbb{N} \}$$

Considérons la grammaire CS avec marquant

$$\mathcal{G}(!) = (\mathcal{V}(!) , \mathcal{V}_T , S , \# , \mathcal{R}(!)) .$$

- $\mathcal{V}(!)$ comprend les deux symboles distingués S et #, les éléments de $\mathcal{V}_1$:

$$\mathcal{V}_1 = \{ A , A_1 , A_2 , B , C , C_1 , C_2 , D \}$$

et ceus de

$$\mathcal{V}_1' = \{ X' \mid X \in \mathcal{V}_1 \} , \quad \mathcal{V}_1'' = \{ X'' \mid X \in \mathcal{V}_1 \} , \quad \mathcal{V}_1''' = \{ X''' \mid X \in \mathcal{V}_1 \}$$

et le symbole terminal | .

Nous allons décrire les règles de la grammaire $\mathcal{G}(!)$ en indiquant les dérivations qu'elles permettent. L'idée directrice est la suivante: on passe de $|^{n!}$ à $|^{(n+1)!}$ en "copiant" (n + 1) fois le première phrase $|^{n!}$:

$$|^{(n+1)!} = \underbrace{|^{n!} \, |^{n!} \ldots |^{n!}}_{(n+1) \text{ fois}}$$

Ainsi au niveau non terminal le passage de $|^{3!}$ à $|^{4!}$ correspondra à la dérivation:

$$\# AAAB^3 \# \xrightarrow[\varphi(!)]{} \# A^4 B^{20} \#$$

le nombre de symboles A permet de savoir à quelle étape on est rendu. On montrera comment les règles décrites permettent de réaliser la dérivation précédente.

I. Règle initiale (S, AA)

II. Règles permettant les dérivations du type:

$$\# A^3 B^3 \# \xrightarrow[\varphi(!)]{} \# DAAB^3 (C_1)^6 \#$$

. (# A , # DA_1) -(début)

. (A_1A , AA_1C) ; (A_1B , BA_1C) : règles de "recopiage"

. (CA , AC) ; (CB , BC)
  (C # , C_1 #) ; (CC_1 , C_1C_1) } mise en place des nouveaux symboles

. (A_1C_1 , C_1C_1)
  $\forall X \in \mathcal{V}_1$, $\forall Y \in \mathcal{V}_1 - \{ A_2 , D \}$
  (YX' , Y'X) } passage à l'étape suivante

Par la suite les symboles X et Y désigneront un symbole quelconque de $\mathcal{V}_1$, les restrictions faites seront seules indiquées.

III. Règles permettant les dérivations du type

$$\# DAAB^3(C_1)^6 \# \xrightarrow[\varphi(!)]{} \# DAA_2 B^3 (C_1C_2C_2)^6 \#$$

. (DA' , DA''_2) ou (A_2A' , AA''_2) : début de la nouvelle étape

. (X''Y , XY'') $X \neq C_1$

. (C''_1Y , C_1C_2Y'') , (C''_1 # , C_1C_2 #) : règles de "recopiage"

. (C''_2 # , C'_2 #)

D'une façon générale s'il s'agit du passage de $|^{n!}$ à $|^{(n+1)!}$ cette étape recommence (n - 1) fois.

IV. Règles permettant les dérivations du type

$$\# DAA_2B^3 (C_1C_2C_2)^6 \# \xrightarrow[\mathcal{G}(1)]{} \# DA_3B^{20} \#$$

. (A_2C_1' , $A''A'''$) ou (A_2B' , $A''A'''$) : un nouveau symbole A apparaît en vue d'une nouvelle étape éventuelle

. ($X'''Y$, XY''') $X \neq C_1$ et $X \neq C_2$

. ($C_1''Y$, BY''')

. ($C_2''Y$, BY''') ou (C_2'' # , B #).

V. Enfin les règles

. (XA''' , $X'''A$)

. (# D''' , # A)

achèvent un développement complet (dans l'exemple choisi on aboutira à: # A^4B^{20} #), et l'on pourra recommencer un nouveau développemnt (on passerait à: # A^5B^{115} #) ou appliquer les :

VI. Règles terminales:

. (S , |)

. (# A , # |) (|A , | |)

. (|B , | |)

On vérifiera que :

$$\mathcal{L}(\mathcal{G}(1)) = \mathcal{L}(1) = \{ |^{n!} \mid n \in \mathbb{N} \}$$

2.3. 3ème exemple

$\forall$ i i ≥ 2 : $\mathcal{L}^{(i)} = \{ |^{n^i} \mid n \in \mathbb{N} \}$.

En s'inspirant de la grammaire $\mathcal{G}(1)$ on peut trouver une grammaire CS engendrant $\mathcal{L}^{(i)}$ [4]. Cela nous permet de donner dans le cadre des langages CS un énoncé du "théorème" de Fermat énonçant que les équations

$$x^n + y^n = z^n \quad ,$$

pour n entier supérieur ou égal à 3, n'admettent pas de solutions à valeurs entières. Il est facile de vérifier que cela revient à poser:

$$\forall i , i \in \mathbb{N}, i \geq 3 : \left(\mathcal{L}^{(i)} . \mathcal{L}^{(i)}\right) \cap \mathcal{L}^{(i)} = \emptyset$$

Le produit $\left(\mathcal{L}_1 . \mathcal{L}_2 = \left\{ \hat{X} = \hat{X}_1 . \hat{X}_2 \mid \hat{X}_1 \in \mathcal{L}_1 \text{ et } \hat{X}_2 \in \mathcal{L}_2 \right\}\right)$ et l'intersection (cf. § 1.2.) étant stables relativement à la classe des langages CS, on peut définir une grammaire CS engendrant le langage du premier membre de l'égalité précédente. Malheureusement le problème général, de savoir si le langage engendré par une grammaire CS est vide, est indécidable.

Nous avons aussi démontré [5] que le langage

$$\mathcal{L}(p) = \left\{ |^p \mid p \in \mathbb{N}, p \text{ premier} \right\}$$

est CS.

Mais ici nous allons présenter une grammaire CS engendrant les nombres premiers écrits sous forme décimale.

§ 3. NOMBRES PREMIERS DECIMAUX ET LANGAGES CS.

L'ensemble des nombres premiers, écrits sous forme décimale usuelle, est un langage CS.

DEMONSTRATION:

Considérons la grammaire CS avec marquant:

$$\mathcal{G} = \left\{ \mathcal{V}, \mathcal{V}_T , S , \# , \mathcal{R} \right\}$$

où: - le vocabulaire terminal, $\mathcal{V}_T$, est l'ensemble des chiffres,

$$\mathcal{V}_T = \left\{ 0, 1, 2, \ldots, 8, 9, \right\}$$

- le vocabulaire non terminal, $\mathcal{V}_N$, comporte en plus des deux symboles distingués S et # , tous les sumboles du type: $\left(\begin{array}{c} c_3 \\ \hline c_2 \\ \hline c_1 \end{array}\right)$ où

c_1 , c_2 , c_3 sont des chiffres arbitraires l'un quelconque d'entre eux pouvant éventuellemnt être indicé par « ' » ou « " » , plus précisément:

$$\mathcal{V}_N = \left\{ S,\ \#,\ \left[\begin{matrix} c_3 \\ c_2 \\ c_1 \end{matrix}\right],\ \left[\begin{matrix} c_3 \\ c_2 \\ c_1' \end{matrix}\right],\ \left[\begin{matrix} c_3 \\ c_2 \\ c_1'' \end{matrix}\right],\ \left[\begin{matrix} c_3 \\ c_2' \\ c_1 \end{matrix}\right],\ \left[\begin{matrix} c_3 \\ c_2'' \\ c_1 \end{matrix}\right],\ \left[\begin{matrix} c_3' \\ c_2 \\ c_1 \end{matrix}\right],\ \left[\begin{matrix} c_3'' \\ c_2 \\ c_1 \end{matrix}\right] \;\middle|\; c_1, c_2, c_3 \in \mathcal{V}_T \right\};$$

- les règles de la grammaire $\mathcal{G}$ vont être divisées en trois groupes.

I. Règles initiales:

. $\# S \longrightarrow \# SS$

. $\# S \longrightarrow \# \left[\begin{matrix} c \\ 0 \\ c \end{matrix}\right] S$ $\qquad c \in \mathcal{V}_T - \{0\}$.

. $\left[\begin{matrix} c_1 \\ 0 \\ c_1 \end{matrix}\right] SS \longrightarrow \left[\begin{matrix} c_1 \\ 0 \\ c_1 \end{matrix}\right]\left[\begin{matrix} c_2 \\ 0 \\ c_2 \end{matrix}\right] S$

$\left[\begin{matrix} c_1 \\ 0 \\ c_1 \end{matrix}\right] S\# \longrightarrow \left[\begin{matrix} c_1 \\ 0 \\ c_1 \end{matrix}\right]\left[\begin{matrix} c_2 \\ 2 \\ c_2 \end{matrix}\right] \#$ $\qquad c_1 , c_2 \in \mathcal{V}_T$

. $\# S \# \longrightarrow \# \left[\begin{matrix} u \\ 2 \\ u \end{matrix}\right] \#$ $\qquad u \in \mathcal{V}_T - \{0 , 1\}$.

Faisons ici l'hypothèse que le symbole S n'intervient pas dans les autres règles et que ces dernières n'agissent que sur les symboles du type $\left[\begin{matrix} c_3 \\ c_2 \\ c_1 \end{matrix}\right]$ indicés ou placés à l'extrémité gauche: il en résulte que toute #S#-dérivation commencera par une séquence:

$$\#S\# \longrightarrow \#SS\# \longrightarrow \#S^{n-1}\# \longrightarrow \# \left[\begin{matrix} c_n \\ 0 \\ c_n \end{matrix}\right] S^{n-1} \# \longrightarrow \ldots$$

$$\ldots \longrightarrow \# \left[\begin{matrix} c_n \\ 0 \\ c_n \end{matrix}\right] \ldots \left[\begin{matrix} c_p \\ 0 \\ c_p \end{matrix}\right] S^{n-p+1} \# \longrightarrow \ldots \longrightarrow \# \left[\begin{matrix} c_n \\ 0 \\ c_n \end{matrix}\right] \ldots \left[\begin{matrix} c_2 \\ 0 \\ c_2 \end{matrix}\right]\left[\begin{matrix} c_1 \\ 2 \\ c_1 \end{matrix}\right] \#, \quad c_n \neq 0$$

- ou simplement $\#S\# \longrightarrow \#\left[\begin{matrix}u\\ \hline 2\\ \hline u\end{matrix}\right]\#$ $u \in \sigma_T - \{0, 1\}$

Ces dérivations faites, les règles initiales ne pourront plus être appliquées.

II. Règles de transformation:

Elles seront décrites formellement ci-dessous; signalons ici qu'il s'agit de règles conservant la longueur et que leur but est de déterminer si, oui ou non, le nombre N,

$$N = c_n c_{n-1} \cdots c_2 c_1$$

est premier. Dans ce but N va être "divisé" par les nombres D, compris entre 2 et N:

$$D = d_1 d_{1-1} \cdots d_2 d_1$$

Ainsi, en cours de dérivation on passe par le mot non terminal:

$$\left[\begin{matrix}c_n\\ \hline 0\\ \hline c_n\end{matrix}\right] \cdots \left[\begin{matrix}c_1+1\\ \hline 0\\ \hline c_1+1\end{matrix}\right]\left[\begin{matrix}c_1\\ \hline d_1\\ \hline c_1\end{matrix}\right] \cdots \left[\begin{matrix}c_2\\ \hline d_2\\ \hline c_2\end{matrix}\right]\left[\begin{matrix}c_1\\ \hline d_1\\ \hline c_1\end{matrix}\right] ,$$

que l'on représenter schématiquement par $\left[\begin{matrix}N\\ \hline D\\ \hline N\end{matrix}\right]$, $2 \leq D \leq N$.

- Si N est un multiple de D on obtient le mot $\left[\begin{matrix}0\\ \hline D\\ \hline N\end{matrix}\right]$, i.e. :

$$\left[\begin{matrix}0\\ \hline 0\\ \hline c_n\end{matrix}\right] \cdots \left[\begin{matrix}0\\ \hline d_1\\ \hline c_1\end{matrix}\right] \cdots \left[\begin{matrix}0\\ \hline d_2\\ \hline c_2\end{matrix}\right]\left[\begin{matrix}0\\ \hline d_1\\ \hline c_1\end{matrix}\right] ,$$

et l'on bloque la dérivation, N n'étant pas premier.

- Si N n'est pas divisible par D on passe à la "division" par (D + 1), c'est-à-dire au mot non terminal $\left[\begin{matrix}N\\ \hline D+1\\ \hline N\end{matrix}\right]$

Il en résulte que si et seulemnt si N est premier on aboutit au mot non terminal $\left[\begin{matrix}N\\ \hline N\\ \hline N\end{matrix}\right]$, i.e. :

$$\left[\begin{matrix}c_n\\ c_n\\ c_n\end{matrix}\right]\left[\begin{matrix}c_{n-1}\\ c_{n-1}\\ c_{n-1}\end{matrix}\right] \cdots \left[\begin{matrix}c_2\\ c_2\\ c_2\end{matrix}\right]\left[\begin{matrix}c_1\\ c_1\\ c_1\end{matrix}\right]$$

dont on dérivera le nombre premier N par les règles terminales.

III. Règles terminales. c, c_1 , c_2 sont des chiffres quelconques:

$$\left[\begin{matrix}c\\ c\\ c\end{matrix}\right] \# \longrightarrow c \#$$

$$\left[\begin{matrix}c_1\\ c_1\\ c_1\end{matrix}\right] c_2 \longrightarrow c_1 c_2$$

Il s'agit maintenant de préciser les règles de transformation et de décrire les dérivations qu'elles permettent. On l'a dit ci-dessus, leur but est de "diviser" N par D ($2 \leq D \leq N$). Cette division va consister en une succession de soustractions.

Soit: $N = kD + R \quad 0 \leq R < D$, (N n'étant multiple d'aucun nombre inférieur à D, autre que 1).

la dérivation passera par les mots non terminaux $\left[\begin{matrix}N - mD\\ D\\ N\end{matrix}\right] \quad 0 \leq m \leq k.$

- si $R = 0$, la dérivation sera bloquée, pour $m = k$, à $\left[\begin{matrix}0\\ D\\ N\end{matrix}\right]$;

- si $R \neq 0$, après la $(k + 1)$-ème soustraction de D, la présence de la retenue ($N - (k + 1)D < 0$) nous fera passer au mot non terminal $\left[\begin{matrix}N\\ D + 1\\ N\end{matrix}\right]$

Distinguons donc les différentes étapes et les différents cas suivants:

II. 1. Début de soustraction.

Supposons que nous ayons dérivé de S le mot $\begin{bmatrix} N - mD \\ \hline D \\ \hline N \end{bmatrix}$,

$0 \leq m \leq k$, avec:

$N - mD = r_i \ \ldots \ r_1$.

A. $r_1 \neq 0$ (donc à fortiori $(N - mD) > 0$)

$$\cdot \begin{bmatrix} r_1 \\ \hline d_1 \\ \hline c_1 \end{bmatrix} \# \longrightarrow \begin{bmatrix} r_1 \\ \hline d'_1 \\ \hline c_1 \end{bmatrix} \#$$

Les chiffres r_1 , d_1 , $_1$, sont quelconques, seule existe la restriction signalée, $r_1 \neq 0$. Cette remarque vaut pour toutes les autres règles.

B. $r_1 = 0$: il s'agit de vérifier si $(N - mD)$ n'est pas nul.

$$(1) \cdot \begin{bmatrix} 0 \\ \hline d_1 \\ \hline c_1 \end{bmatrix} \# \longrightarrow \begin{bmatrix} 0' \\ \hline d_1 \\ \hline c_1 \end{bmatrix} \#$$

Nous nous intéressons uniquement aux chiffres de la ligne supérieure, aussi les règles précédentes peuvent, sans aucune confusion, se mettre sous la forme: $\lceil \underline{0} \rceil \# \longrightarrow \lceil \underline{0'} \rceil \#$

étant bien entendu que les deux chiffres inférieurs sont quelconques (à remarquer toutefois qu'ils ne peuvent être indicés) et ne sont pas modifiés par ces règles. Cette remarque nous permettra de simplifier l'écriture de certaines règles

$$\cdot \lceil \underline{0} \rceil \lceil \underline{0'} \rceil \longrightarrow \lceil \underline{0'} \rceil \lceil \underline{0} \rceil$$

Ainsi dans le cas où $N = mD$ on se trouvera arrêté à:

$$\# \begin{bmatrix} 0' \\ \hline 0 \\ \hline c_n \end{bmatrix} \begin{bmatrix} 0 \\ \hline 0 \\ \hline c_{n-1} \end{bmatrix} \ \ldots \ \begin{bmatrix} 0 \\ \hline d_1 \\ \hline c_1 \end{bmatrix} \ \ldots \ \begin{bmatrix} 0 \\ \hline d_1 \\ \hline c_1 \end{bmatrix} \#$$

(Tout au plus pourrait-on ajouter de nouveaux indices « ' » à la ligne supérieure).

. Si (N - mD) est positif il existe j, $j \leq i$, tel que r_j est distinct de 0, et il faut alors continuer notre division par D par une nouvelle soustraction:

. $r \neq 0$ $\left[\underline{r}\right]\left[\underline{0'}\right] \longrightarrow \left[\underline{r}\right]\left[\underline{0''}\right]$

. $\left[\underline{0''}\right]\left[\underline{0}\right] \longrightarrow \left[\underline{0}\right]\left[\underline{0''}\right]$

$\left[\begin{matrix}0'' \\ \underline{d_1} \\ c_1\end{matrix}\right] \#\# \longrightarrow \left[\begin{matrix}0 \\ \underline{d'_1} \\ c_1\end{matrix}\right] \#$ ou plus simplement $\left[\underline{\begin{matrix}0'' \\ d_1\end{matrix}}\right] \# \longrightarrow \left[\underline{\begin{matrix}0 \\ d'_1\end{matrix}}\right] \#$

II. 2. Règles de soustraction.

Seules nous intéressent les deux lignes supérieures; la ligne inférieure, n'étant pas modifiée, conserve l'information du nombre N étudié. Nous avons quatre cas à considérer:

A. Sans retenue (sur le chiffre précédent):

. $r_2 \geqslant d_2$: $\left[\underline{\begin{matrix}r_1 \\ d_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 \\ d'_2\end{matrix}}\right] \longrightarrow \left[\underline{\begin{matrix}r_1 \\ d'_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 - d_2 \\ d_2\end{matrix}}\right]$

. $r_2 < d_2$: $\left[\underline{\begin{matrix}r_1 \\ d_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 \\ d'_2\end{matrix}}\right] \longrightarrow \left[\underline{\begin{matrix}r_1 \\ d''_1\end{matrix}}\right]\left[\underline{\begin{matrix}(10 + r_2) - d_2 \\ d_2\end{matrix}}\right]$

on introduit la retenue (évidemment égale à 1) en utilisant l'indice « " » .

B. Avec retenue:

. $r_2 > d_2$: $\left[\underline{\begin{matrix}r_1 \\ d_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 \\ d''_2\end{matrix}}\right] \longrightarrow \left[\underline{\begin{matrix}r_1 \\ d'_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 - d_2 - 1 \\ d_2\end{matrix}}\right]$

. $r_2 \leq d_2$: $\left[\underline{\begin{matrix}r_1 \\ d_1\end{matrix}}\right]\left[\underline{\begin{matrix}r_2 \\ d''_2\end{matrix}}\right] \longrightarrow \left[\underline{\begin{matrix}r_1 \\ d''_1\end{matrix}}\right]\left[\underline{\begin{matrix}(9 + r_2) - d_2 \\ d_2\end{matrix}}\right]$

(introduction de la retenue)

II. 3. Fin de soustraction.

Il y a essentiellemnt deux cas à envisager:

A. $N - (m + 1)D \geqslant 0 \qquad (m < k)$

. sans la retenue:

$$r \geqslant d \ : \ \# \left| \frac{r}{d'} \right| \longrightarrow \# \left| \frac{r - d}{d} \right|$$

. avec la retenue:

$$r > d \ : \ \# \left| \frac{r}{d''} \right| \longrightarrow \# \left| \frac{r - d - 1}{d} \right|$$

B. $N - (m + 1)D < 0 \qquad (m = k)$

. sans la retenue:

$$r < d \ : \ \# \left[\begin{matrix} r \\ d' \\ c \end{matrix} \right] \longrightarrow \# \left[\begin{matrix} c \\ d \\ c' \end{matrix} \right]$$

. avec la retenue:

$$r \leqslant d \ : \ \# \left[\begin{matrix} r \\ d'' \\ c \end{matrix} \right] \longrightarrow \# \left[\begin{matrix} c \\ d \\ c' \end{matrix} \right]$$

L'indice « ' » qui apparaît à la ligne inférieure, indique que N n'est pas divisible par D; il va permettre de passer à la division par (D + 1) grâce aux règles suivantes.

II. 4. Passage à la division par (D + 1); c'est-à-dire passage au mot non terminal: $\left[\begin{matrix} N \\ \hline D + 1 \\ \hline N \end{matrix} \right]$

$$\left[\begin{matrix} c_1 \\ d_1 \\ c'_1 \end{matrix} \right] \left[\begin{matrix} r \\ d_2 \\ c_2 \end{matrix} \right] \longrightarrow \left[\begin{matrix} c_1 \\ d_1 \\ c_1 \end{matrix} \right] \left[\begin{matrix} c_2 \\ d_2 \\ c'_2 \end{matrix} \right]$$

A. $d_1 \neq 9$. $\left[\begin{matrix} c_1 \\ d_1 \\ c'_1 \end{matrix} \right] \# \longrightarrow \left[\begin{matrix} c_1 \\ d_1 + 1 \\ c_1 \end{matrix} \right] \#$

B. $d_1 = 9$. $\begin{bmatrix} c_1 \\ \hline 9 \\ \hline c'_1 \end{bmatrix} \# \longrightarrow \begin{bmatrix} c_1 \\ \hline 9 \\ \hline c''_1 \end{bmatrix} \#$ la présence de l'indice « " » à la ligne inférieure indique que le passage de D à D + 1 fait intervenir la retenue (D se terminant par un 9).

$$\cdot \begin{bmatrix} c_1 \\ \hline d \\ \hline c_1 \end{bmatrix} \begin{bmatrix} c_2 \\ \hline 9 \\ \hline c''_2 \end{bmatrix} \longrightarrow \begin{bmatrix} c_1 \\ \hline d \\ \hline c''_1 \end{bmatrix} \begin{bmatrix} c_2 \\ \hline 0 \\ \hline c_2 \end{bmatrix}$$

$$\cdot \; d \neq 9 \quad \begin{bmatrix} c \\ \hline d \\ \hline c'' \end{bmatrix} \longrightarrow \begin{bmatrix} c \\ \hline d+1 \\ \hline c \end{bmatrix}$$

Il résulte de la description faite que tout nombre premier N dérive de S, selon la grammaire $\mathcal{G}$. Il resterait à voir que la grammaire $\mathcal{G}$ n'engendre pas, ou plutôt "n'accepte pas" d'autres nombres. On remarquera que les règles doivent être appliquées dans l'ordre de leur présentation si l'on veut que la dérivation aboutisse à une phrase. Pour cela on considérera l'apparition et la disparition des indices qui commandent, en quelque sorte, l'enchaînement des étapes.

Ainsi s'achève la démonstration du résultat énoncé:

$$\mathcal{L}(\mathcal{G}) = \{ N \mid N \text{ nombre premier décimal} \} .$$

REMARQUES

1°)- A ce résultat nous pouvons ajouter le suivant:

Le complémentaire du langage $\mathcal{L}(\mathcal{G})$, c'est-à-dire le langage: $\mathcal{L}' = \{ N' \mid N' \text{ entier naturel décimal non premier} \}$ est un langage CS.

Il suffit de considérer la grammaire $\mathcal{G}$ et d'en remplacer les règles terminales III par les suivantes:

$$\cdot \; d < c \quad \# \begin{bmatrix} 0 \\ \hline d \\ \hline c \end{bmatrix} \longrightarrow \# \; c$$

$$\cdot\ c_1 \left[\begin{array}{c} 0 \\ \hline d_2 \\ \hline c_2 \end{array}\right] \longrightarrow c_1 c_2$$

en ajoutant # S # ⟶ # 0 # et # S # ⟶ # 1 # .

Le lecteur vérifiera ce résultat facilement en remarquant que N n'est pas premier si et seulement s'il existe D, D < N (on peut préciser $D \leq \frac{N}{2}$) tel que N = kD. Dans une dérivation selon la grammaire $\mathcal{G}$, essayant d'engendrer N, on aboutit au mot $\left[\begin{array}{c} 0 \\ \hline D \\ \hline N \end{array}\right]$. En posant:

$$N = c_n \ldots c_1 \qquad \text{et} \qquad D = d_1 \ldots d_1$$

on est sûr que l'on a $d_n < c_n$ (car N ⩾ 2D) d'où la restriction imposée ci-dessus à la première règle terminale applicable. Elle était nécessaire, car si N est premier, la grammaire $\mathcal{G}$ nous permet d'aboutir au mot $\left[\begin{array}{c} N \\ \hline N \\ \hline N \end{array}\right]$ et en continuant d'appliquer les règles transformation, au mot $\left[\begin{array}{c} 0 \\ \hline N \\ \hline N \end{array}\right]$, mais alors $d_n = c_n$ et notre nouvelle grammaire "n'accepte pas" N.

2°) Nous allons donner une nouvelle grammaire CS engendrant le langage:

$$\mathcal{L}(p) = \left\{ |^p \mid p \in \mathbb{N}, \text{ p premier} \right\} ,$$

Auparavant, donnons un ensemble $\mathcal{R}_1$ de règles CS permettant de passer de l'écriture décimale d'un entier naturel p à son écriture "en bâtons".

. Règles CS de passage de # p # à # $|^p$ #.

Soit $p = c_n c_{n-1} \cdots c_2 c_1$.

Nous allons distinguer 3 cas suivant la valeur du chiffre des unités.

- 1er cas: c > 1 . c # ⟶ (c - 1) | #

. c | ⟶ (c - 1) ||

- 2ème cas C = 1 . d 1 # ⟶ d 0 | # } d est un chiffre
. d 1 | ⟶ d 0 || } quelconque.

règle de fin de transformation . # 1 | ⟶ # ||

cas particulier . # 1 # ⟶ # | #

- 3ème cas: C = 0 . 0 # ⟶ 0' #

. 0 | ⟶ 0' |

. 0 0' ⟶ 0' 0

d > 1 . d 0' ⟶ (d - 1)9 |'

d = 1 { . u 1 0' ⟶ u 0 9 |' u est un chiffre
{ . # 1 0' ⟶ # 9 |' quelconque

. |' 0 ⟶ 9 |'

. |' # ⟶ | #

. |' | ⟶ ||

On remarquera que la dérivation

$$\# \; p \; \# \quad \Longrightarrow \quad \# \; |^{p} \; \#$$

est définie d'une façon unique; on passera par les étapes:

$$\# \; p \; \# \Longrightarrow \# \; (p-1) \; | \; \# \Longrightarrow \ldots \# \; q \; |^{p-q} \; \# \longrightarrow \ldots \longrightarrow \# \; |^{p} \; \# \; ,$$

le passage de q à (q - 1) ($1 < q \leq p$) exige, dans le cas où le chiffre des unités de q est nul, l'intervention d'une retenue.

Nous sommes maintenant en mesure de définir une grammaire engendrant $\mathcal{L}(p)$:

$\mathcal{G}' = (\mathcal{V}', \{ | \}, S, \#, \mathcal{R}')$

- $\mathcal{V}' = \mathcal{V} \cup \{ 0', | ' \}$, $\mathcal{V}$ étant le vocabulaire terminal et non terminal de la grammaire $\mathcal{G}$ étudiée ci-dessus.

- $\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_1$, $\mathcal{R}$ étant l'ensemble des règles de la grammaire $\mathcal{G}$ et $\mathcal{R}_1$ l'ensemble des règles de passage décrites ci-dessus.

On a bien: $\mathcal{L}(\mathcal{G}') = \mathcal{L}(p)$

En conclusion nous allons énoncer des résultats permettant de comparer la classe des langages CS à deux autres classes qui généralisent celle des langages CF.

- Langages CF matricielles.

Une grammaire matricielle est un couple $[\mathcal{G}, \mathcal{M}]$ où $\mathcal{G}$ est une grammaire de constituants.

$\mathcal{G} = (V, V_T, S, \mathcal{R})$

et $\mathcal{M}$ une partie finie du monoïde libre $\mathcal{R}^*$. $[\mathcal{G}, \mathcal{M}]$ sera dite grammaire matricielle CF, CS, etc., si $\mathcal{G}$ est respectivement une grammaire CF, CS, etc... Le langage engendré par la grammaire matricielle $[\mathcal{G}, \mathcal{M}]$ est l'ensemble:

$$\mathcal{L}[\mathcal{G}, \mathcal{M}] = \{ \hat{x} \mid \hat{x} \in V_T^* ; \exists \hat{p} \in \mathcal{M}^* : S \xRightarrow[\mathcal{G}]{\hat{p}} \hat{x} \}$$

des mots sur le vocabulaire V_T qui dérivent de S par une suite de règles de $\mathcal{R}$ qui est un produit de juxtaposition de mots de $\mathcal{M}$. (Autrement dit on applique d'abord dans leur ordre les règles d'un premier mot $\hat{p}_1 \in \mathcal{M}$; puis on passe à un second mot $\hat{p}_2$, dont on doit pouvoir appliquer la première "lettre", qui est une règle de $\mathcal{R}$, puis la seconde etc...).

J. P. Benzécri a prouvé [2] que :

. tout langage engendré par une grammaire CF matricielle est CS...

On a plus: tout langage CS matriciel est CS.

- Langages CFP (cf. § 2 au début)

Nous avons prouvé (travail à paraître) que:

. tout langage CFP est CS.

B I B L I O G R A P H I E

- o - O - o -

[1] J. P. BENZECRI (Rennes, février 1965) Structures algèbriques et constituants non connexes dans les grammaires.

[2] J. P. BENZECRI (Paris, 1966) Grammaires matricielles

[3] N. CHOMSKY, Formal Properties of Grammars, dans Handbook of Mathematical Psychology, vol. II (Ed. by D. Luce, E. Busch, E. Galanter) 1963, pp. 323-418.

[4] J. FRIANT (Thèse de 3° cycle, Paris, 1966) Les langages CS.

[5] J. FRIANT (Paris 1966) Nombres premiers et langages CS.

[6] R. GUEDJ (Thèse de 3° cycle, Paris, 1966) Grammaires de Constituants Généraux.

[7] S. Y. KURODA (Octobre 1963), Classes of Languages and Linear bounded autamata.

[8] P. S. LANDWEBER, Three theorems on phrase structure grammars of type 1, Information and Control, t. 6, 1963, p. 137-146.