9 MARS 1968

LANGAGES

LES MODÈLES EN LINGUISTIQUE

DIDIER
4 et 6, rue de la Sorbonne
PARIS

LAROUSSE
13-21, rue du Montparnasse
PARIS

N. CHOMSKY ET M. P. SCHUTZENBERGER

THÉORIE ALGÉBRIQUE DES LANGAGES « CONTEXT-FREE » *

1. Motivation linguistique.

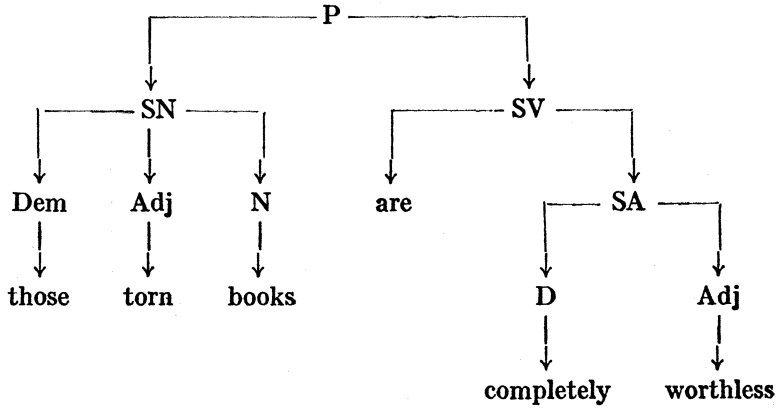
Nous nous intéresserons ici à plusieurs classes de processus générateurs de phrases qui sont liés de près, à beaucoup d'égards, aux grammaires des langues naturelles et de divers langages artificiels. Par *langage*, nous entendons simplement un ensemble de séquences sur un ensemble fini quelconque V de symboles, appelé *vocabulaire* du langage. Une *grammaire* sera un ensemble de règles énumérant récursivement les séquences appartenant au langage. Nous dirons que la grammaire *engendre* ces séquences. (En pensant aux langues naturelles nous appellerions les séquences engendrées, des *phrases;* en termes algébriques on les appellerait habituellement des *mots*, et le vocabulaire serait appelé un *alphabet;* en considérant la grammaire comme déterminant un langage de programmation, on appellerait les séquences des *programmes;* en général nous emploierons le terme neutre de *séquence.*)

Pour qu'une classe de grammaires soit linguistiquement intéressante, il doit exister un procédé qui fasse correspondre à n'importe quel couple (σ, G), où σ est une séquence et G une grammaire de cette classe, une *description structurale* satisfaisante de la séquence σ, par rapport à la grammaire G. En particulier la description structurale devrait indiquer, s'il en est ainsi, que la séquence σ est une phrase bien formée du langage $L(G)$ engendré par G. Dans ce cas, la description structurale doit contenir des renseignements grammaticaux servant de base à l'explication de la manière dont σ est interprétée par des « locuteurs » qui ont assimilé la grammaire G; sinon, il serait bon que la description structurale indique en quoi σ diffère d'une séquence bien formée.

Nous ne nous intéresserons ici qu'à un aspect de la description struc-

* Ce travail a été financé en partie par : U. S. Army Signal Corps, Air Force Office of Scientific Research, Office of Naval Research, et en partie par National Science Foundation et par un don du Commonwealth Fund. Il a été publié en anglais dans : *Computer Programming and Formal Systems*, P. Brafford et D. Hirschberg Eds, North Holland, Amsterdam, 1963.

turale d'une phrase, à savoir sa subdivision en syntagmes appartenant à différentes catégories. Ainsi, par exemple, une description structurale de la phrase anglaise « those torn books are completely worthless » devrait indiquer que *those* est un *Démonstratif*, *torn* et *worthless* des *Adjectifs*, *books* un *Nom*, *completely* un *Adverbe*, *those torn books* un *Syntagme Nominal*, *completely worthless* un *Syntagme Adjectival*, are *completely worthless* un *Syntagme Verbal*, que la séquence tout entière est une *Phrase*, ainsi que des détails supplémentaires quant à la sous-classification. Ces renseignements peuvent être représentés par un schéma du type (1) :



ou, d'une manière équivalente, par un parenthésage étiqueté de la séquence, comme dans (2) :

$$[_{P}[_{SN}[_{Dem}\text{those}][_{Adj}\text{torn}][_{N}\text{books}]][_{SV}\text{are}\ [_{SA}[_{D}\text{completely}][_{Adj}\text{worthless}]]]].$$

Une préoccupation essentielle de la théorie générale des langues naturelles est de définir la classe des séquences possibles (en fixant un alphabet phonétique universel); la classe des grammaires possibles; la classe des descriptions structurales possibles; un procédé permettant d'attribuer des descriptions structurales aux phrases, étant donnée une grammaire; et de faire tout ceci de telle façon que la description structurale attribuée à une phrase par la grammaire d'une langue naturelle soit une base pour expliquer comment une personne qui parle cette langue comprend cette phrase (en supposant l'absence de limitations dues à la mémoire, l'attention, etc.). Alors, la grammaire représentera certains aspects de la compétence linguistique de celui qui parle la langue.

Nous ne nous intéresserons pas ici à la question empirique de l'adéquation à la réalité des descriptions structurales ou des grammaires que nous étudierons. D'ailleurs les classes de grammaires que nous envisagerons et les genres de descriptions structurales qu'elles engendrent, sont sans aucun doute trop pauvres, pour être à la hauteur de la véritable

compétence linguistique de l'homme. Néanmoins, les systèmes que nous considérons (qui, effectivement, formalisent des notions traditionnelles de l'analyse logique et de l'analyse en constituants immédiats) sont à rapprocher des types de systèmes qui semblent empiriquement adéquats, et sont jusqu'à présent, trop complexes pour être soumis à une étude abstraite [1].

Dans la représentation (2), nous avons, en plus des parenthèses, deux sortes de symboles :

(i) des symboles de la séquence engendrée (i.e. les six symboles *those, torn, books, are, completely, worthless*) [2];

(ii) les symboles *P, SA, Dem., Adj, N, SV, SA, D*, représentant des catégories syntaxiques. Nous appellerons les symboles du type (i) terminaux, ceux du type (ii) non terminaux.

Nous supposerons, ci-après, qu'il existe une collection déterminée de symboles terminaux et non terminaux à partir desquels les grammaires de toutes les langues sont construites. On peut considérer que l'ensemble des terminaux constitue un vocabulaire potentiel commun à toutes les langues. Par analogie avec le langage parlé, on peut considérer que l'ensemble des terminaux est défini par un alphabet phonétique universel (en supposant, comme il est naturel de le faire que la longueur des morphèmes est bornée supérieurement). En examinant de nouveau le langage naturel, on peut considérer l'ensemble fixe des non-terminaux comme un ensemble universel de catégories dans lequel on prend les divers types de syntagmes du langage. Une question traditionnelle importante de la linguistique générale touche à la possibilité de donner une interprétation concrète des non-terminaux à l'aide desquels sont construites les grammaires. Est-il possible, autrement dit, de trouver une définition générale, indépendante de toute langue particulière, de catégories telles que Nom, Verbe, etc..., en termes de contenu sémantique ou de propriétés formelles des grammaires? Le problème d'une interprétation concrète à donner à l'ensemble des terminaux et non-terminaux est évidemment le même que le problème qui consiste à rendre empiriquement adéquates certaines catégories des grammaires, un point capital dans la science du langage; mais il va au-delà de nos préoccupations immédiates.

Nous pouvons engendrer la phrase : « those torn books are completely worthless » avec la description structurale (2), au moyen de l'ensemble des *règles de réécriture* :

1. Pour une discussion plus poussée de ces questions, voir Chomsky [10].

2. Dans une grammaire linguistiquement adéquate, on n'engendrerait pas ces, symboles, mais plutôt une représentation plus abstraite, utilisant les symboles *the, démonstratif, pluriel, tear, participe, book, pluriel, be, pluriel, complete, by, worth, less*, dans cet ordre. La représentation au moyen de ces symboles (appelés morphèmes) serait transformée en une représentation phonétique par un ensemble de *règles phonologiques* auxquelles nous ne nous intéresserons pas ici. Voir Chomsky et Miller [15]. Nous n'utiliserons des phrases véritables comme (2) que pour des exemples descriptifs, et par conséquent nous laisserons de côté des raffinements de ce genre.

(3)

$P \rightarrow SN\ SV$
$SN \rightarrow Det\ Adj\ N$
$Dem \rightarrow$ those
$Adj \rightarrow$ torn
$Adj \rightarrow$ worthless
$N \rightarrow$ books
$SV \rightarrow$ are SA
$SA \rightarrow D\ Adj$
$D \rightarrow$ completely

par une *dérivation* construite de la manière suivante.

On écrit d'abord le *symbole initial* P comme première ligne de la dérivation. On forme la $(n + 1)^e$ ligne de la dérivation en choisissant n'importe quelle occurrence d'un α non terminal dans la n^e ligne (où cette occurrence de α n'étiquette pas une parenthèse), et en la remplaçant par la séquence : $[_\alpha \varphi]$, où $\alpha \rightarrow \varphi$ est une des règles de (3). On continue jusqu'à ce que les seuls non-terminaux qui apparaissent soient ceux qui étiquètent des parenthèses; la dérivation est alors *terminée.* En supprimant les parenthèses d'une dérivation terminée avec leurs étiquettes, on a une séquence qui ne contient que des terminaux. Appelons ceci une *séquence terminale.* Quatre séquences terminales différentes peuvent être engendrées par la grammaire (3). Il est possible de construire une grammaire qui engendre une infinité de séquences terminales, chacune avec une description structurale, en autorisant des récurrences, par exemple en ajoutant à (3) les règles :

(4)

$SN \rightarrow$ that P
$SV \rightarrow$ is SA
$SA \rightarrow$ obvious

auquel cas on peut engendrer par exemple « that those torn books are completely worthless is obvious », etc. [3]. Chacune des phrases engendrées aura encore une description structurale du type voulu.

Des grammaires du type (3)-(4) seront appelées « context-free » *((N. d. T.)* C-grammaires). Elles sont caractérisées par le fait qu'il apparaît exactement un symbole non terminal dans le membre gauche de chaque règle de réécriture.

Si cette restriction n'est pas exigée les systèmes obtenus ont alors des propriétés formelles entièrement différentes.

Il semble que les grammaires des langues naturelles doivent contenir

3. Dans ce cas, une infinité de phrases non anglaises seront également engendrées, par exemple « that those torn books is obvious are completely worthless », etc. Donc la grammaire ((3), (4)) est inacceptable. La difficulté qu'il y a à éviter de telles insuffisances empiriques peut être facilement sous-estimée. Soulignons à nouveau que c'est là le point déterminant, aussi bien pour la linguistique que pour la psychologie, bien qu'il ne soit pas abordé directement ici. Pour une discussion, voir Chomsky [13].

au moins quelques règles de réécriture de ce type plus général, et certaines règles qui ne sont pas du tout des règles de réécriture. Cf. Chomsky [8] [10] et [12], Chomsky et Miller [15] pour une discussion abstraite plus poussée de tels systèmes que nous n'examinerons pas plus en détail ici. Un ensemble de séquences terminales qui peut être engendré par une certaine C-grammaire sera appelé un *C-langage.*

Une C-grammaire peut engendrer une séquence terminale *(sans parenthèses)* φ avec plusieurs descriptions structurales différentes. Dans ce cas, si la grammaire est empiriquement adéquate, la séquence φ est structuralement ambiguë. Considérons par exemple la C-grammaire, ayant les règles :

(5)
$P \rightarrow SN\ SV$
$SN \rightarrow$ they; $SN \rightarrow Adj\ N$; $SN \rightarrow N$
$SV \rightarrow$ are SN; $SV \rightarrow$ Verbe SN
$Verbe \rightarrow$ are flying
$Adj \rightarrow$ flying
$N \rightarrow$ planes

Avec cette grammaire on peut engendrer aussi bien (6) que (7) :

(6) $[_P[_{SN}\text{they}]\ [_{SV}[_{Verbe}\text{are flying}]\ [_{SN}[_N\text{planes}]]]]$.

(7) $[_P[_{SN}\text{they}]\ [_{SV}\text{are}[_{SN}[_{Adj}\text{flying}]\ [_N\text{planes}]]]]$.

Corrélativement, la séquence terminale « they are flying planes » est structuralement ambiguë; elle peut vouloir dire : « my friends who are pilots are flying planes » : (mes amis, qui sont pilotes, sont en train de piloter des avions), ou : « those spots on the horizon are flying planes » (ces taches sur l'horizon sont des avions qui volent). L'étude de l'ambiguïté structurale est une des façons les plus instructives de déterminer si une grammaire est empiriquement adéquate ou non.

Nous verrons plus loin que l'ambiguïté de certains C-langages est inhérente, dans la mesure où toute C-grammaire qui les engendre attribue plusieurs descriptions structurales à certaines de leurs phrases.

De plus, nous verrons que le problème de savoir si une C-grammaire est ambiguë est récursivement indécidable [4], même pour des types de C-grammaires extrêmement simples.

Bien que les C-grammaires soient loin d'être suffisantes pour les langues naturelles, elles sont sans aucun doute satisfaisantes pour la description des langages artificiels usuels, et apparemment pour la description de certains (peut-être tous) langages de programmation. En particulier

4. En d'autres termes, il n'y a pas de procédure mécanique (d'algorithme) permettant de savoir si une C-grammaire quelconque attribue plus d'une description structurale à une séquence qu'elle engendre.

on peut donner une C-grammaire pour l'Algol [18] et chaque programme en Algol sera une des séquences terminales engendrées par cette grammaire.

Il est clair qu'un langage de programmation ne doit pas être ambigü. Par conséquent, il est important de pouvoir s'assurer qu'un langage de programmation particulier satisfait effectivement cette condition, ou encore que dans un certain ensemble infini de programmes, chacun d'eux est sans ambiguïté, certaines techniques utilisées pour les construire étant données (par exemple des techniques qu'on peut représenter comme des règles de construction de dérivations dans une C-grammaire). Comme on l'a signalé dans le précédent paragraphe, ces questions peuvent être assez difficiles.

Supposons que G_1 et G_2 soient des systèmes générateurs qui spécifient certaines techniques de construction de programmes de calculateurs; supposons, en fait, que G_1 et G_2 soient des grammaires engendrant les langages de programmation L_1 et L_2, consistant chacun en un nombre infini de séquences, chaque séquence étant un programme possible. Il est souvent intéressant d'étudier les puissances relatives des langages de programmation.

Nous verrons que si G_1 et G_2 sont des C-grammaires (comme par exemple pour l'Algol), la plupart des problèmes touchant aux rapports entre L_1 et L_2 sont récursivement indécidables, ceci est vrai en particulier pour le problème de savoir si L_1 et L_2 ont une intersection vide, ou infinie, ou si L_1 est contenu dans L_2 [2], ou s'il existe un transducteur fini (un compilateur) qui applique L_1 sur L_2 (Ginsburg et Rose, communication personnelle).

Par conséquent, il est possible que des questions générales sur les propriétés formelles des C-systèmes et de leurs relations formelles aient une interprétation concrète dans l'étude des systèmes de traitement de l'information aussi bien que dans celle des langues naturelles. Cette possibilité a été montrée en particulier par Ginsburg et Rice [18], Ginsburg et Rose [19].

Lorsqu'on considère une grammaire comme un processus générateur, on peut s'intéresser au langage (à l'ensemble des séquences terminales) qu'elle engendre, ou bien à l'ensemble des descriptions structurales qu'elle engendre (N. B. : chaque description structurale détermine une séquence terminale de manière unique, comme dans (2)). Cette dernière question est évidemment de loin la plus intéressante. De la même façon, dans l'étude de la capacité génératrice d'une classe de grammaires (ou de la capacité relative de plusieurs de ces classes, comme dans l'évaluation de théories linguistiques (soumises à un choix), on peut s'intéresser ou bien à l'ensemble des langages qu'on peut engendrer, ou bien à l'ensemble des systèmes de descriptions structurales qu'on peut engendrer. Ici encore, le dernier aspect est plus intéressant, mais beaucoup plus difficile. Ces questions ont toutes été abordées très récemment et on s'est limité presque

exclusivement à engendrer des langages plutôt que des systèmes de descriptions structurales.

Nous envisagerons la génération d'un point de vue intermédiaire entre les deux précédents. Nous considérerons une représentation d'un langage qui ne sera ni un ensemble de séquences ni non plus un ensemble de descriptions structurales, mais un ensemble de couples (σ, n), où σ est une séquence, et n son degré d'ambiguïté, c'est-à-dire le nombre des descriptions structurales différentes attribuées à σ par la grammaire G qui engendre le langage auquel σ appartient.

2. Les grammaires en tant que générateurs de séries formelles de puissances.

2.1. Supposons donné un vocabulaire fini V, où les ensembles V_T (= vocabulaire terminal) et V_N (= vocabulaire non terminal) forment une partition.

Considérons maintenant des langages sur le vocabulaire V_T, et des grammaires ayant leurs symboles non terminaux dans V_N. Soit $F(V_T)$ le monoïde libre engendré par V_T, i. e. l'ensemble de toutes les séquences sur le vocabulaire V_T. Un langage est alors un sous-ensemble de $F(V_T)$.

Considérons une application r qui fasse correspondre à chaque séquence $f \varepsilon F(V_T)$ un certain entier $< r, f >$. Une telle application peut être représentée par une *série formelle de puissances* (notée également r) sur les variables x non-commutatives de V_T. Ainsi :

$$(8) \qquad r = \sum_i < r, f_i > f_i = < r, f_1 > f_1 + < r, f_2 > f_2 + \ldots,$$

où $f_1, f_2, \ldots$ est une énumération de toutes les séquences de V_T. Le support de r (= *Supp.* (r)) est défini comme l'ensemble des séquences à coefficients non nuls dans r. Ainsi :

$$(9) \qquad \textit{Supp.}\ (r) = \{ f_i \varepsilon F(V_T) \mid < r, f_i > \neq 0 \}.$$

Il n'est pas exigé que les coefficients $< r, f_i >$ de la série formelle de puissance r soient positifs. S'ils le sont (pour tout i, $< r, f_i > \geqslant 0$), nous dirons que r est une série formelle de puissances POSITIVE. Si pour chaque $f_i \varepsilon F(V_T)$, le coefficient est 0 ou 1, nous dirons que r est la série formelle de puissances *caractéristique* de son support.

2.2. Si r est une série formelle de puissances et n un entier, le produit nr est par définition la série formelle de puissances de coefficients $< nr, f > = n < r, f, >$ où $< r, f >$ est le coefficient de f dans r. Si r et r' sont des séries formelles de puissances, $r + r'$ est par définition, la série formelle de puissances de coefficients : $< r + r', f > = < r, f > + < r', f >$ où $< r, f >$ et $< r', f >$ sont respectivement les coefficients de f dans r et r'. rr' sera définie comme la série formelle de puissances de

coefficients $< rr', f > = \sum_{i,j} < r, f_i > < r', f_j >$, avec $f_i f_j = f$. Ainsi l'ensemble des séries formelles de puissances est un anneau fermé pour les opérations : multiplication par un entier, addition, multiplication.

Remarquons que si r et r' sont des séries formelles de puissances le support de $r + r'$ est exactement la réunion des supports de r et r', et le support de rr' est exactement l'ensemble produit des supports de r et r' (i. e., l'ensemble de toutes les séquences $f_i f_j$ telles que f_i soit dans le support de r et f_i dans le support de r'). Nous discuterons ci-dessous l'interprétation d'autres opérations théoriques simples sur ces ensembles.

On dira que deux séries formelles de puissances r et r' sont équivalentes mod. degré n (i. e., $r \equiv r'$ (mod. *deg.* n)) si $< r, f > = < r', f >$ pour chaque séquence f de longueur (« degré ») $\leqslant n$. Supposons alors que nous ayons une suite infinie de séries formelles de puissances $r_1, r_2 \ldots$, telles que pour chaque n et chaque $n' > n$, $r_n \equiv r_{n'}$ (mod. *deg.* n). Dans ce cas la limite r de la suite $r_1, r_2, \ldots$, est bien définie comme étant :

$$r = \lim_{n \to \infty} \pi_n r_n$$

où, pour chaque n, $\pi_n r_n$ est le polynôme formé à partir de r_n en remplaçant tous les coefficients des séquences de longueur $> n$ par zéro. L'anneau des séries formelles de puissances devient donc ultramétrique, donc topologique.

Ces notions étant définies, abordons le problème du lien entre la représentation des langages en termes de série formelle de puissances et la représentation des langages par des processus générateurs tels que les C-grammaires.

2.3. Soit G un proccessus générateur engendrant le langage $L(G)$. A chaque séquence $f \varepsilon F(V_T)$, G associe un certain nombre $N(G, f)$ de descriptions structurales; $N(G, f) > 0$ seulement si $f \varepsilon L(G)$. $N(G, f)$ exprime le degré d'ambiguïté structurale de f par rapport à G. Il est naturel d'associer à G la série formelle de puissances $r(G)$ telles que $< r(G), f > = N(G, f)$, $< r(G), f >$ étant le coefficient de f dans $r(G)$. Ainsi $r(G)$ exprime l'ambiguïté de toutes les séquences terminales vis-à-vis de la grammaire G. Le coefficient $< r(G), f >$ n'est égal à *zéro* que si f n'est pas engendrée par G; il n'est égal à *1* que si f est engendrée sans ambiguïté (d'une manière et d'une seule) par G; il est égal à *2* s'il existe deux descriptions structurales différentes pour f, en termes de G; etc...

Un $r(G)$ associé à une grammaire G sera évidemment toujours positif; et son support *Supp.* $(r(G))$ sera exactement le langage $L(G)$ engendré par G. Nous pouvons considérer qu'une série formelle de puissances r qui a des coefficients positifs et négatifs est associée à *deux* processus générateurs G_1 et G_2. On peut prendre pour coefficient $< r, f >$ de f dans r la différence entre les nombres de fois où f est engendrée par G_1 et G_2; c'est-à-dire, dans ce cas,

$$< r, f > = N(G_1, f) - N(G_2, f).$$

Supposons que G soit une C-grammaire d'éléments non terminaux $\alpha_1, \alpha_2, \ldots \alpha_n$, où α_1 est le symbole initial désigné (i. e., $\alpha_1 = P$ dans l'exemple (1), ci-dessus).

Nous pouvons construire la série formelle de puissances $r(G)$ associée à G par une méthode d'itération directe. Pour cela nous procéderons de la façon suivante :

Remarquons d'abord que G peut s'écrire sous la forme d'un système d'équations par rapport aux variables $\alpha_1, \ldots, \alpha_n$. Soient $\varphi_{i,1} \ldots, \varphi_{i,m_i}$ les séquences telles que $\alpha_i \to \varphi_{i,j}$ $(1 \leqslant j \leqslant m_i)$ soient des règles de G. Associons alors à α_i l'expression polynômiale σ_i,

$$\sigma_i = \varphi_{i,1} + \varphi_{i,2} + \ldots + \varphi_{i,m_i}. \tag{11}$$

Associons maintenant à la grammaire G le système d'équations :

$$\alpha_1 = \sigma_1; \ldots \alpha_n = \sigma_n. \tag{12}$$

Supposons que la grammaire G ne contienne pas de règles de la forme :

$$\begin{aligned} &\alpha_i \to e \\ &\alpha_i \to \alpha_j. \end{aligned} \tag{13}$$

Il est clair que ces suppositions n'affectent en rien la capacité génératrice [2]. C'est-à-dire que pour chaque C-grammaire comportant de telles règles, il en existe une autre sans aucune règle de cette sorte et qui engendre le même langage. Nous exigerons explicitement encore, que si G est une C-grammaire, α un non-terminal de G, il existe des séquences terminales dérivables de α — i. e. si G' contient les règles de G et possède α comme symbole initial, le langage engendré par G' doit être non vide. Il est évident que cette exigeance n'affecte pas non plus la capacité génératrice.

Revenons maintenant au problème de la construction de la série de puissances associée à G_1 et qui représente le degré d'ambiguïté que G attribue à chaque séquence; remarquons que chaque équation $\alpha_i = \sigma_i$ de (12) peut être considérée comme définissant une application ψ_i qui applique un n-uple $(r_1, \ldots, r_n)$ de séries de puissances sur la série de puissances obtenue en remplaçant α_j par r_j dans σ_i. Ceci est légitime du fait des propriétés de clôture de l'anneau des séries de puissances, indiquées ci-dessus au § 2.2.

Ainsi l'ensemble d'équations (12) définit une application ψ,

$$\psi(r_1, \ldots, r_n) = (r'_1, \ldots, r'_n) \text{ où } r'_i = \psi_i(r_1, \ldots, r_n). \tag{14}$$

Considérons maintenant la suite infinie de n-uples de séries de puissances $\rho_0, \rho_1, \ldots$, où :

$$\begin{aligned} \rho_0 &= (r_{0,1}, \ldots, r_{0,n}) = (0, \ldots, 0) \\ \rho_1 &= (r_{1,1}, \ldots, r_{1,n}) \\ \rho_2 &= (r_{2,1}, \ldots, r_{2,n}) \end{aligned}$$

et où pour chaque i, j $(j > 0)$

$$r_{j,i} = \psi_i (r_{j-1,1}, \ldots, r_{j-1,n}), \tag{16}$$

et où 0 est la série de puissances qui a tous ses coefficients nuls. Chaque $r_{j,i}$ de (15) n'a qu'un nombre fini de coefficients non nuls; autrement dit, c'est un polynôme. De plus, on peut montrer que pour tout i, j, j' tels que $j' > j > 0$, $1 \leqslant i \leqslant n$, on a bien :

$$r_{j,i} \equiv r_{j',i} \text{ (mod. } deg.\ j). \tag{17}$$

Par la suite, comme on l'a vu au § 2.2, la limite $r_{\infty,i}$ de la suite infinie $r_{1,i}, r_{2,i}, \ldots$, est bien définie pour tout i (ce n'est évidemment pas en général un polynôme). Nous appellerons le n – uple $(r_{\infty,1}, \ldots, r_{,\infty n})$ ainsi défini, la *solution* du système d'équations (12). En effet, le n – uple $(r_{\infty'1}, \ldots, r_{\infty'n})$ est le seul n – uple à satisfaire, dans notre cadre, le système d'équations (12). Pour cette raison, nous dirons qu'une série de puissances est *algébrique* [42] si elle est un des termes d'une solution d'un système d'équations tels que (12), sans restriction sur le signe des coefficients numériques. Nous dirons qu'une série de puissances est « context-free » si les coefficients des équations de définitions sont tous positifs.

En particulier $r_{\infty,1}$, que nous appellerons désormais, la *série de puissances engendrée* par la grammaire G de (12) de symbole initial α_1, est la série de puissances associée à G, de la manière décrite au début du § 2.3. Son support est le langage $L(G)$ engendré par G, et le coefficient $< r_{\infty,1}, f >$ d'une séquence $f \varepsilon F(V_T)$ détermine l'ambiguïté de ce f par rapport à G, de la manière décrite ci-dessus.

Remarquons que si une série algébrique de puissances est « context-free », elle est positive, mais la réciproque n'est pas nécessairement vraie. C'est ainsi qu'une série de puissances pourra être un terme d'une solution d'un système d'équations, et n'avoir que des coefficients positifs, mais ne pas être un terme d'une solution de n'importe quel système d'équations à coefficients tous positifs[5].

2.4. Comme exemple du procédé ci-dessus, considérons les deux grammaires (18) et (19) :

$$P \to bPP;\ P \to a \tag{18}$$

$$P \to PbP;\ P \to a. \tag{19}$$

Chacune de ces grammaires n'a qu'un non-terminal, donc le système d'équations correspondant, consistera, dans les deux cas, en une équation unique. A (18) correspond (20) et à (19), (21) :

$$P = bPP + a \tag{20}$$

5. Par exemple, en utilisant les notions qui seront définies plus bas § 3.1, le carré d'Hadamard $s \odot s$, pour $s \varepsilon \lambda_0$ n'a que des coefficients positifs (et a le même support que s), mais en général, n'est pas engendré par un ensemble d'équations à coefficients seulement positifs.

(21) $$P = PbP + a.$$

Les équations (19) et (20) correspondent à (12) ci-dessus, avec $n = 1$. Elles remplissent toutes deux la condition (13).

Envisageons d'abord la grammaire (18) représentée sous la forme (20). En procédant comme dans l'alinéa précédent, on considère que (20) définit une application ψ telle que $\psi(r) = a + brr$, où r est une série de puissances.

On forme alors (comme en (15)) la suite infinie $\rho_0, \rho_1, \rho_2, \ldots$, de la façon suivante :

(22)
$$\begin{aligned}
&\rho_0 = r_0 = 0\\
&\rho_1 = r_1 = a + br_0r_0 = a + b00 = a\\
&\rho_2 = r_2 = a + br_1r_1 = a + baa\\
&\rho_3 = r_3 = a + br_2r_2 = a + b(a + baa)\,(a + baa)\\
&\qquad\qquad = a + baa + babaa + bbaaa + bbaabaa\\
&\rho_4 = r_4 = a + br_3r_3\\
&\ldots\quad\ldots\\
&\ldots\quad\ldots
\end{aligned}$$

Il est clair que pour tout j, j', tels que $j' > j > 0$, on a $r_j \equiv r_{j'}$ (mod. *deg. j*). Par suite la limite r_∞ est bien définie. Cette série de puissances est la solution de l'équation (20), et son support est le langage engendré par la C-grammaire (18). Remarquons que la série de puissances r_∞ est, dans ce cas, caractéristique, et que son support, est l'ensemble des *formules bien formées* du « *calcul des implications* » à une variable, dans la notation sans parenthèses (notation polonaise) (le symbole a jouant le rôle de variable propositionnelle et b le rôle de l'opérateur « conditionnel »).

Considérons maintenant la grammaire (19) représentée sous la forme (21). Nous regarderons (21), comme définissant une application ψ telle que $\psi(r) = a + rbr$, où r est une série de puissances. Formons la suite infinie $\rho_0, \rho_1, \rho_2 \ldots$:

(23)
$$\begin{aligned}
&\rho = r_0 = 0\\
&\rho = r_1 = a + r_0br_0 = a + 0b0 = a\\
&\rho = r_2 = a + r_1br_1 = a + aba\\
&\rho = r_3 = a + r_2br_2 = a + (a + aba)b(a + aba)\\
&\qquad\qquad = a + aba + 2ababa + abababa\\
&\rho = r_4 = a + r_3br_3\\
&\qquad = a + aba + 2(ab)^2\,a + 5(ab)^3\,a + 6(ab)^4\,a + 6(ab)^5\,a +\\
&\qquad\quad 4(ab)^6\,a + (ab)^7\,a\\
&\ldots\quad\ldots
\end{aligned}$$

Ici encore pour tout j, j' tels que $j' > j > 0$, on a $r_j \equiv r_{j'}$ (mod. deg j) et la limite r_∞ est définie comme étant la série des puissances :

(24) $r_\infty = \sum_n \begin{bmatrix} 2n \\ n \end{bmatrix} \frac{1}{n+1} (ab)^n a = a + aba + 2(ab)^2a + 5(ab)^3a + 14(ab)^4a$

$+ 42(ab)^5a + \ldots\ldots$

où $$\begin{bmatrix} 2n \\ n \end{bmatrix} = \frac{2n \times 2n-1 \times \ldots \times n+1}{1 \times 2 \times \ldots \times n}.$$

La série de puissances r_∞ de (24) est la solution de l'équation (21) et son support est le langage engendré par la grammaire (19). Ce n'est pas, dans ce cas, une série de puissances caractéristique. En prenant encore le symbole *a* comme variable propositionnelle et *b* comme signe pour « conditionnel », la grammaire (19) est l'ensemble de règles qui engendre les *formules bien formées* du *calcul des implications* à une variable, en notations ordinaires mais sans parenthèses. Les descriptions structurales engendrées par (19), de la manière décrite dans le premier alinéa (cf. (3)) sont évidemment sans ambiguïté, puisqu'on conserve les parenthèses, mais les séquences terminales formées en supprimant les parenthèses sont ambiguës et le degré d'ambiguïté de chaque séquence terminale engendrée est exactement son coefficient dans r_∞; ainsi on peut interpréter *ababa* de deux façons, soit comme (*ab*(*aba*)), soit comme ((*aba*)*ba*), etc. Un cas plus général a été traité par Raney [38] au moyen de la formule d'inversion de Lagrange.

Dans (20) et (21) tous les coefficients sont positifs et la solution est donc une série de puissances positive. Considérons, toutefois, le système d'équations constitué par l'équation unique :

(25) $$S = a - SbS.$$

Nous avons alors la suite :

(26)
$$\begin{aligned}
\rho_0 &= r_0 = 0 \\
\rho_1 &= r_1 = a - r_0 b r_0 = a - 0b0 = a \\
\rho_2 &= r_2 = a - r_1 b r_1 = a - aba \\
\rho_3 &= r_3 = a - r_2 b r_2 = a - (a - aba)\, b(a - aba) \\
&\qquad = a - aba + 2\, ababa - abababa.
\end{aligned}$$

Les coefficients de ρ_i dans (26) sont précisément les mêmes que ceux de ρ_i dans (23) sauf pour le signe —, le coefficient de *f* dans ρ_i de (26) n'est positif que si *f* comporte un nombre pair de *b*.

La série de puissances r_∞, solution de (25), n'est pas positive et donc n'est pas « context-free » (bien qu'il se trouve que son support soit ici un langage « context-free »; c'est d'ailleurs un langage dont (19) est une grammaire). Nous pouvons cependant considérer r_∞ comme la différence entre les deux séries de puissances « context-free » r^+ et r_∞^-; et corrélativement nous pouvons considérer que son support est l'ensemble des séquences qui ne sont pas engendrées le même nombre de

fois par un couple de C-grammaires $G+$ et G^- qui engendrent $r_\infty+$ et $r_\infty-$ respectivement. Posons $S = S+ - S-$ de telle façon que (25) devienne :

$$\begin{aligned}(27)\quad S+ - S- &= a - (S+ - S-)\, b\, (S+ - S-)\\ &= a - (S+bS+ - S+bS- - S-bS+ + S-bS-)\\ &= a + S+bS+ + S-bS+ - (S+bS+ + S-bS-).\end{aligned}$$

Considérons alors l'ensemble des équations :

$$\begin{aligned}(28)\quad &(\text{i}) \qquad S+ = a + S+bS- + S-bS+\\ &(\text{ii}) \qquad S- = S+bS+ + S-bS-.\end{aligned}$$

C'est un ensemble d'équations positives à deux variables $S+$ et $S-$ admettant comme solution $(r_\infty+, r_\infty-)$, où $r_\infty+$ est la limite de la suite $r_0+, r_1- \ldots$, et $r_\infty-$ la limite de la suite $r_0+, r_1-, \ldots$ de (29) :

$$\begin{aligned}(29)\qquad \rho_0 &= (r_0+, r_0-) = (0, 0)\\ \rho_1 &= (r_1+, r_1-) = (a, 0)\\ \rho_2 &= (r_2+, r_2-) = (a, aba).\end{aligned}$$

Il est clair que lorsque r_∞ est la solution de (25), $r_\infty = r_\infty{}^+ - r_\infty{}^-$. Mais de plus, $r_\infty{}^+$ est la série de puissances engendrée par la C-grammaire G^+ de symbole initial S^+ et de grammaire (28 i) et $r_\infty{}^-$ est la série de puissances engendrée par la C-grammaire G^- de symbole initial S^- et de grammaire (28 ii).

D'une manière analogue toute série algébrique de puissances peut être représentée (d'une infinité de façons différentes) par la différence de deux séries de puissances « context-free », et son support peut donc être considéré comme l'ensemble des séquences qui ne sont pas engendrées le même nombre de fois par deux C-grammaires. C'est ce que nous pouvons donner de plus concret comme interprétation d'une série algébrique de puissances.

Plus généralement, la même construction pourrait se faire pour un anneau quelconque de coefficients, au lieu de l'anneau des nombres naturels utilisés ci-dessus. Ce domaine reste inexploré. Par exemple si les coefficients sont pris modulo un nombre premier (i. e. si on considère comme « non produites » les séquences produites un nombre de fois égal à un multiple de p), la série formelle de puissances $\Sigma n > o^{z^{p^n}}$ à un seul terminal z est algébrique [27] alors que son support ne peut être aucune des séries de puissances introduites plus haut.

3. Autres opérations sur les séries formelles de puissances.

3.1. Au § 2.2 nous avons vu qu'un ensemble de séries de puissances est fermé pour les opérations d'addition, produit, et multiplication par un entier. Nous avons signalé que le support de $r + r'$ est la réunion des

supports de r et r' et que le support de rr' est l'ensemble produit des supports de r et r', pourvu que les coefficients ne soient pas négatifs.

Considérons maintenant deux autres opérations pour lesquelles l'ensemble des séries de puissances est fermé, et considérons l'interprétation correspondante en théorie des ensembles pour leurs supports.

Il est classique de dire que r est quasi régulier si $< r, e > = 0$. Alors $r^{n'} \not\equiv 0 \pmod{\deg n}$ pour $0 < n < n'$ et l'élément $r^* = \lim_{n\to\infty} \Sigma_{0<n'<n} r^{n'}$ est bien défini. De plus r^* vérifie l'identité

$$r + r^*r = r + rr^* = r^* \tag{30}$$

qui le détermine de façon unique. C'est pourquoi on appelle habituellement r^* le quasi-inverse de r. Cette notion est en rapport direct avec la notion plus familière d'inverse grâce à la remarque suivante : si $r' = e - r$ et $r'' = e + r^*$, $r'r'' = (e - r)(e + r^*) = e - r + r^* - rr^* = e = r''r'$, c'est-à-dire $r'' = r'^{-1}$.
Inversement, étant donné r' tel que $< r', e > = 1$, on peut l'écrire : $r' = e - r$, où r est quasi régulier, si bien que $e + r^*$ est l'inverse de r''.

Remarquons que, par définition même de r^*, cette série de puissances n'a que des coefficients non négatifs si c'est le cas pour r, et que $supp.\ r^* = (supp.\ r)^*$ où au deuxième membre, l'étoile représente l'opération étoile de Kleene [21].

En particulier, si V est un ensemble arbitraire de lettres et si la série de puissances ν est définie par $<\nu, x> = 1$ si $x \in V$, $<\nu, x> = 0$ si $x \notin V$ (i. e. si ν est la fonction caractéristique de V), $e + V^*$ (au sens de Kleene) est l'ensemble de tous les mots engendrés par les lettres de V et $e + \nu^* = (e - \nu)^{-1}$ est la fonction caractéristique de cet ensemble. Ceci résulte du fait que tout mot $f \in V^*$ apparaît une fois et une seule dans la somme infinie $\Sigma_{n>0} V^n$. Par la suite, quand on connaît la fonction caractéristique r d'un ensemble de séquences, on peut aussi écrire la fonction caractéristique $(1 - V_T)^{-1} - r$ de son complément.

Il est inutile de remarquer que dans ce cas, cette dernière a des coefficients non négatifs et, bien qu'algébrique au sens défini plus haut, elle n'est pas pour autant obligatoirement « context-free ».

La deuxième opération que nous définissons est le produit d'Hadamard, ce qui généralise de l'une des manières possibles, la notion habituelle de l'analyse classique. La définition que nous donnons diffère des extensions diverses au cas de plusieurs variables que l'on trouve dans la littérature, mais paraît être l'extension la plus naturelle pour les séries de puissances non commutatives.

Si r et r' sont deux séries de puissances, leur produit d'Hadamard $r \odot r'$ sera la série de puissances dont les coefficients sont :

$$<r \odot r', f> = <r, f> <r', f> \tag{31}$$

identiquement pour toutes les séquences f.

D'où *supp.* $(r \odot r') = (supp.\ r) \cap (supp.\ r')$ et $r \odot r'$ est une fonction caractéristique si r et r' le sont.

Enfin, nous introduisons la notation suivante : étant donnée une séquence $x_{i_1} x_{i_2} \ldots x_{i_{n-1}} x_{i_n} = f(x_{i_j} \varepsilon V)$ nous définissons $\tilde{f}$ (l'image miroir de f) comme la séquence :

$$\tilde{f} = x_{i_n} x_{i_{n-1}} \ldots\ldots\ldots x_{i_2} x_{i_1}. \tag{32}$$

Il est clair que $\tilde{\tilde{f}} = f$ et la relation $ff' = f''$ implique $\tilde{f}'' = \tilde{f}' \tilde{f}$. Formellement, cette application est un *antiautomorphisme involutif* de l'anneau des séries de puissances, et on peut démontrer qu'il est caractérisé de façon unique par cette propriété (à une permutation près des éléments de V).

3.2. La notation qui vient d'être introduite sera utilisée plus loin pour simplifier la description des grammaires de la manière suivante. Supposons que la grammaire G contienne les règles :

$$\begin{aligned} \alpha_1 &= \pi_1 \alpha_2 \pi_2 + \pi_1 \pi_2 + \pi_3 \\ \alpha_2 &= \alpha_2 \pi_4 + \pi_4 \end{aligned} \tag{33}$$

où les π_j sont des expressions polynômiales sur $V - \{ \alpha_1 \}$. Alors la seconde règle entraîne :

$$\alpha_2 = \left(\sum_{n>0} \pi_4^n \right) \tag{34}$$

et on peut remplacer les règles (33) par la règle plus simple :

$$\alpha_1 = \pi_1 (1 - \pi_4)^{-1} \pi_2 + \pi_3. \tag{35}$$

On peut, d'ailleurs, donner une interprétation linguistique de cette forme simplifiée de description. Ainsi, par exemple, un couple de règles de la forme $\alpha_1 \to f_1 \alpha_2 f_2$, $\alpha_2 \to \alpha_2 \alpha_2$ c'est-à-dire un couple que l'on peut désormais écrire : $\alpha_1 \to f_1 (1 - \alpha_2)^{-1} f_2$ \ peut être considéré comme constituant en fait un schéma de règle : $\alpha_1 \to f_1 \alpha_2^n f_2 (n = (1, 2, \ldots))$. Avec cette nouvelle interprétation, la grammaire, bien qu'encore déterminée par un nombre fini de schémas de règle, contient une infinité de règles. Rappelons alors la façon dont une description structurale (un parenthésage étiqueté) est attribuée à une séquence terminale engendrée par une C-grammaire (voir ci-dessus, § 1). Une grammaire déterminée par le schéma de règle ci-dessus peut engendrer une description structurale de la forme :

$$\ldots [_{\alpha_1} f_1 [_{\alpha_2} p_1] [_{\alpha_2} p_2] \ldots [_{\alpha_2} p_n] f_2] \ldots \tag{36}$$

pour chaque n, où chaque p_k est dérivé de α_2. Dans la phrase (séquence terminale) ayant cette description, chaque $\bar{p}_k$ est un syntagme du type α_2, où $\bar{p}_k$ est formé par « déparenthésage » de p_k. Les syntagmes successifs $\bar{p}_1, \ldots, \bar{p}_n$, constituent une « coordination », qui, prise avec les séquences

finales provenant de f_1 et f_2, est une construction de type α_1. Ceci est la manière naturelle d'étendre les C-grammaires pour rendre compte de la véritable coordination, comme par exemple lorsqu'une séquence d'adjectifs de longueur arbitraire et sans structure interne peut apparaître en position d'attribut. Cf. Chomsky [10].

3.3. Essayons de rattacher ce qui a été fait jusqu'à présent, à l'analyse classique en écrivant $\varphi f = \varphi f'$, pour deux séquences quelconques f et f', si elles contiennent exactement le même nombre de chacune des lettres (qu'elles soient terminales ou non).

Il est clair que φ se prolonge en une application de nos séries de puissances non commutatives sur l'anneau des séries *formelles* de puissances ordinaires (commutatives) à coefficients entiers et il est facile de voir que φ est un homomorphisme. Par exemple, si $\alpha = a + b\alpha\alpha$, on a : $\varphi\alpha = \varphi a + \varphi b\varphi\alpha\varphi\alpha$, et $\varphi\alpha$ est la série de puissances ordinaire

$$\varphi\alpha = \sum_n (\varphi a)^{n+1} (\varphi b)^n \begin{bmatrix} 2n \\ n \end{bmatrix} \frac{1}{n+1} \tag{37}$$

par rapport aux variables ordinaires φa, φb. (Ici, avec $\alpha' = a + \alpha' b\alpha'$, on aurait aussi $\varphi\alpha' = \varphi\alpha$.)

De plus, on peut montrer directement, à partir de la façon dont on obtient nos séries de puissances que les coefficients n'augmentent pas plus vite qu'une fonction exponentielle du degré (longueur) des séquences. Ainsi l'image φ de n'importe laquelle de nos séries de puissances est en fait une série de Taylor convergente ordinaire, développement d'une fonction algébrique.

Réciproquement, si on se donne des variables (ordinaires) $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, une fonction algébrique (ordinaire) de cette quantité $\bar{y}$ est définie par un polynôme en $\bar{y}$ et les $\bar{x}$; et si $\bar{y}$ admet un développement en série de Taylor (à coefficients entiers) au voisinage de zéro, par rapport aux $\bar{x}_i$, on peut lui associer une infinité de séries formelles de puissances β telles que $\varphi\beta = \bar{y}$ et β est définie par des équations formelles. Par exemple en partant de la fonction algébrique $\bar{y}$ de $\bar{a}$ et $\bar{b}$ définie par $\bar{y}^2\bar{b} - \bar{y} + \bar{a} = 0$ on obtient les deux exemples donnés plus haut, ainsi que la série formelle de puissances :

$$\alpha = a + b\alpha\alpha + \pi\alpha - \alpha\pi \tag{38}$$

où π est un polynôme quelconque en a et b. Ainsi par exemple, pour $\pi = b$:

$$\begin{aligned} &\alpha_0 = a \\ &\alpha_1 = a + baa + ba - ab \\ &\ldots\ldots\ldots \\ &\text{etc.} \end{aligned} \tag{39}$$

3.4. Concluons en donnant quelques liens entre nos réflexions et la théorie de Lyndon des équations dans un groupe libre (Lyndon, 1960).

Soit $\{ x_i \}$ $(1 \leqslant i \leqslant n)$ un vocabulaire terminal, ξ une lettre non terminale, et w un produit de termes de la forme $1 - x_i, (1 - x_i)^{-1}, 1 - \xi, (1 - \xi)^{-1}$. On définit $deg\ (w) = d+ - d-$ où $d+$ et $d-$ sont les nombres de facteurs $1 - \xi$ et $(1 - \xi)^{-1}$ dans w. Ainsi par exemple pour $w = (1 - x_2)(1 - x_1)(1 - \xi)(1 - x_i)^{-1}(1 - \xi\xi_i)^{-1}(1 - x_2)^{-1}$, on a $deg(w) = 1 - 1 = 0$.

Il est bien connu que les éléments $1 - x_i$ engendrent (par multiplication) un groupe libre G. La relation $w = 1$ peut être considérée comme une équation d'inconnue ξ. Avec notre terminologie, une solution de $w = 1$ serait une série de puissances ξ_0 en les x_i telle que $w = 1$ identiquement, quand ξ_0 est substituée à ξ dans w; ξ_0 sera une solution de groupe si de plus, $1 - \xi_0 \in G$; i. e., si $1 - \xi_0$ est exprimable comme produit de termes $(i - x_i)^{\pm 1}$. R. C. Lyndon a démontré le résultat très remarquable, que l'on peut obtenir par algorithme la *totalité* des *solutions de groupes.*

Rattachons une partie de cette question à nos remarques du § 2.3. Pour cela introduisons les nouveaux symboles ξ_i $(1 < i < n)$, et les équations :

$$\xi_i = x_i + \xi_i x_i\ ;\ \mu = \xi^2 + \xi\mu \tag{1}$$

de telle façon que $(1 - x_1)^{-1} = 1 + \xi_i$ et $(1 - \xi)^{-1} = 1 + \xi + \xi^2 + \xi\mu$

En substituant ces expressions dans $w = 1$, et après simplification, on obtient une relation :

$$(deg(w))\xi = \rho \tag{2}$$

où ρ est un polynôme en les variables x_i, ξ_i, μ, sans terme de degré inférieur à 2.

Par suite si $deg(w) \neq 0$ le système (1), (2) a une solution et une seule en série de puissances (le fait que les coefficients puissent être éventuellement rationnels plutôt qu'entiers ne change rien à la démonstration du § 2.3) et puisque les solutions de groupe forment un sous-ensemble des séries de puissances solutions, nous avons vérifié directement que si $deg(w) \neq 0$ l'équation de groupe libre $w = 1$ a au plus une solution.

Au contraire, si $deg(w) = 0$ (comme par exemple pour l'équation $w = (1 - \xi)(1 - x_i)(1 - \xi)^{-1}(1 - x^i)^{-1} = 1$ notre méthode n'opère plus du tout et ne dit même rien quant aux solutions quelconques (sans restriction) de $w = 1$.

Par exemple $(1 - x_i)(1 - \xi)(1 - x_i)^{\varepsilon}(1 - \xi)^{-1} = 1$ n'a pas de solution pour $\varepsilon \neq -1$ et a une infinité de solutions de *groupe* si $\varepsilon = -1$, viz. $1 - \xi = (1 - x_i)^{\pm n}$ $(n > 0)$. En effet, l'équation peut alors aussi bien s'écrire : $\xi x_1 = x_1 \xi$ (qui a comme solutions, dans notre théorie, toutes les séries de puissance en x_1).

Évidemment, le cas $deg(w) = 0$ est justement celui dans lequel l'inconnue $1 - \xi$ disparaît, quand on prend l'image commutative comme au § 3.3 et c'est le cas non trivial du point de vue de la théorie des groupes.

4. Types de C-grammaires et leurs propriétés générales.

4.1. En termes de conditions sur les règles qui les constituent, on peut définir plusieurs catégories de C-grammaires particulièrement intéressantes. Dans la suite $,\alpha,\beta \ldots$ seront des symboles non terminaux; $f, g, \ldots$ des séquences terminales (pouvant être nulles); et φ, ψ des séquences quelconques. Rappelons que nous avons exclu la possibilité des règles de la forme $\alpha \to e$ ou $\alpha \to \beta$ en remarquant que cette restriction ne modifie pas la capacité génératrice. Nous décrirons les C-grammaires en termes de règles ou d'équations, selon ce qui sera le plus commode.

Si la grammaire G ne contient pas de non-terminal α duquel on peut dériver à la fois une séquence f' et une séquence $f\alpha g$, le langage terminal $L(G)$ engendré par G sera fini. Dans ce cas G sera appelée une *grammaire polynômiale.*

Considérons maintenant des règles grammaticales des types suivants :

(40) (i) $\alpha \to f\beta$ (linéaire à droite)
(ii) $\alpha \to \beta f$ (linéaire à gauche)
(iii) $\alpha \to f\beta g$ (linéaire)
(iv) $\alpha \to fg$ (terminantes)

Une grammaire ne contenant que des règles linéaires à droite et terminantes ou des règles linéaires à gauche et terminantes sera appelée une grammaire *linéaire unilatère.*

Une grammaire ne contenant que des règles du type (40) sera dite *linéaire.* Supposons que G ne contienne que des règles du type (40) et du type $\alpha_1 \to \varphi$ où α_1 est le symbole initial de G; et que, de plus, elle ne contienne aucune règle $\beta \to \varphi\alpha_1\psi$. Ainsi, l'équation de définition de α_1 sera $\alpha_1 = \pi_1$ où π_1 est un polynôme qui ne fait pas intervenir α_1. Une telle grammaire sera dite *méta-linéaire.*

Étant donnée une grammaire G (i. e. un ensemble d'équations positives) polynômiale, linéaire unilatère, linéaire, méta-linéaire ou « context-free », on dira que la série de puissances r, terme principal de sa solution (i. e. qu'elle *engendre,* au sens du § 2.3) et le langage *Supp.* (r) qu'elle engendre, sont respectivement polynômiaux, linéaires unilatères, linéaires, méta-linéaires, ou « context-free ». Ces familles de séries de puissances seront désignées respectivement par $P+, L_o^+, L+, L_m^+, C+$; et pour chaque famille F la famille des supports de F sera notée *Supp.* (F).

Remarquons que *Supp.* ($P+$) est simplement la famille des ensembles finis, et que *Supp.* (L_o+) est la famille des événements réguliers, au sens de Kleene [21] (cf. Chomsky [7]. Remarquons que la classe des événements réguliers est fermée par réflexion).

Considérons maintenant quelques propriétés élémentaires de ces familles de langages.

Il est d'abord immédiat que l'on a les relations d'inclusion suivantes pour ces familles :

(41) *Supp.* $(P+) \subset$ *Supp.* $(L_0+) \subset$ *Supp.* $(L+) \subset$ *Supp.* $(L_m+) \subset$ *Supp.* $(C+)$. De plus, dans chacun de ces cas l'inclusion est stricte. D'où :

PROPRIÉTÉ 1.

$$\textit{Supp.}\,(P+) \subsetneq \textit{Supp.}\,(L_0+) \subsetneq \textit{Supp.}\,(L+) \subsetneq \textit{Supp.}\,(L_m+) \subsetneq \textit{Supp.}\,(C+).$$

L'exemple le plus simple d'un langage de *Supp.* $(L+)$ mais non de *Supp.* (L_0+) est l'ensemble de toutes les séquences $\{a^n b a^n\}$ $(a,b \in V_T)$. Il est engendré par la grammaire : $\alpha = a\alpha a + b$, et il est facile de montrer que ce n'est pas un événement régulier. Le produit des langages dans *Supp.* (L^+) est toujours dans *Supp.* (L_m+), mais n'est généralement pas dans *Supp.* (L^+). Le langage L_{IC} de notre exemple (18) ci-dessus, avec la grammaire

$$\alpha = a + b\alpha\alpha \tag{42}$$

qui est formé des formules du calcul des implications à une variable libre en notation polonaise, est dans *Supp.* (C^+) mais non dans *Supp.* (L_m+). Ceci résulte du fait que L_{IC} contient toutes les séquences de la forme :

$$b^{m_1} a^{m_1} b^{m_2} a^{m_2} \ldots b^{m_k} a^{m_k} a, \tag{43}$$

pour tout $k \geqslant 1$, $m_i \geqslant 1$. Mais chaque séquence de L_{IC} contient n occurrences de b et $n + 1$ occurrences de a, pour un $n \geqslant 1$. Par suite un entier k étant fixé pour engendrer toutes les séquences de la forme (43), on doit pouvoir dériver du symbole initial de la grammaire de L_{IC} une séquence φ contenant k occurrences de non-terminaux. Par suite cette grammaire ne peut être métalinéaire.

Dans une interprétation empirique des C-grammaires, la relation entre *Supp.* (C^+) et *Supp.* (L_0+) est particulièrement importante, du fait qu'un procédé fini, où sont incorporées les instructions d'une C-grammaire G engendrant $L(G)$ comme représentation de sa compétence intrinsèque, ne pourra interpréter que les phrases d'un sous-ensemble fixe $R \in$ *Supp.* (L_0+) de $L(G) \in$ *Supp.* (C^+) au moyen de mécanismes supplémentaires fixés). Cette relation peut justement se décrire grâce à certains traits formels des descriptions structurales (parenthésages étiquetés) engendrées par les C-grammaires. Cf. § 1.

Disons que G est une grammaire *auto-imbriquée* si elle engendre une description structurale de la forme :

$$[\alpha\varphi[\alpha\,\psi]\chi]\ldots, \tag{44}$$

où φ et χ contiennent des terminaux non nuls et où ψ est une expression bien parenthésée. Nous avons le résultat suivant :

THÉORÈME 1*a*.

$L \not\subseteq L_0^+$ *si et seulement si toute C-grammaire engendrant* L *est auto-imbriquée.* Chomsky [9].

Ce résultat peut s'étendre de la manière suivante. Définissons le *degré d'auto-imbrication* d'une description structurale D comme le plus grand N telle que D contienne une sous-configuration :

$$[\alpha\varphi_1[\alpha\varphi_2[\cdots[\alpha\varphi_{N+1}]\varphi_{N+2}]\cdots]\varphi_{2N+1}]$$

où chaque φ contient des terminaux non-nuls. Alors il existe une bijection effective Φ de $\{(G, n); G$ est une C-grammaire, $n \geqslant 1\}$ dans l'ensemble des grammaires linéaires unilatères et une bijection effective Ψ de l'ensemble Δ des descriptions structurales dans Δ telles que :

THÉORÈME 1*b*.

Pour tout L ε *Supp.* (C^+), *il y a une C-grammaire qui engendre* L, *telle que pour chaque* N, $\Phi(G, N)$ *engendre* f *avec la description structurale* D *si et seulement si* G *engendre la séquence terminale* f *avec la description structurale* (ΨD) *où* (ΨD) *a un degré d'auto-imbrication* $\leqslant N$. Chomsky [8].

Ainsi, intuitivement, on peut, étant donnée G, construire un procédé fini $\Phi(G, N)$ qui reconnaisse la structure d'une séquence f engendrée par G pourvu que le degré d'auto-imbrication d'une description structurale particulière de f ne dépasse pas N. Ceci suggère plusieurs conséquences d'ordre empirique. Pour une discussion, cf. Chomsky [10], Miller et Chomsky [29].

4.2. Considérons maintenant diverses propriétés de clôture de ces familles de langages.

On peut caractériser algébriquement les familles de séries de puissances ci-dessus de la manière suivante : P^+ est un demi-anneau [6], L_0^+ est le plus petit demi-anneau contenant P^+ et fermé par quasi-inversion des éléments quasi-réguliers.

L^+ est un module, et L_m^+ est le plus petit demi-anneau le contenant. L'ensemble C^+ est un demi-anneau fermé par quasi-inversion des éléments quasi-réguliers.

Corrélativement, on a les propriétés suivantes des supports : *Supp.* (P) est fermé pour la réunion ensembliste, et le produit ensembliste; *Supp.* (L_0^+) est le plus petit ensemble contenant les ensembles finis et fermé pour les opérations : union ensembliste, produit ensembliste, et l'opération étoile décrite au § 3.1 [21]; *Supp.* (L^+) est fermé pour l'union ensembliste, mais non pour le produit ensembliste, *Supp.* (L_m^+) est le

6. La notion de demi-anneau généralise celle d'anneau par le fait que la structure d'addition n'est qu'une structure de monoïde (et non nécessairement de groupe). Un demi-anneau typique, est « l'anneau Booléen » à deux éléments *0* et *1* et avec les règles ($0 = 0 + 0 = 00 = 01 = 10$; $1 = 0 + 1 = 1 + 0 = 1 + 1 = 11$).

plus petit ensemble contenant les ensembles de *Supp.* (L^+) et fermé pour le produit ensembliste (c'est évidemment la raison de la construction de $L_m{}^+$); l'ensemble *Supp.* (C^+) est fermé pour l'union, le produit, et l'opération étoile.

Ces propriétés sont immédiates et il est naturel d'examiner la clôture par rapport aux autres opérations ensemblistes élémentaires, à savoir l'intersection et la complémentation.

Il est évident que *Supp.* (P^+) est fermé pour l'intersection, et il est bien connu que la classe *Supp.* ($L_0{}^+$) des événements réguliers est fermée pour l'intersection et la complémentation.

Pour les autres familles, on a les résultats suivants : la famille *Supp.* (C^+) de tous les C-langages n'est pas fermée pour l'intersection et donc (puisqu'elle est fermée pour la réunion) n'est pas fermée pour la complémentation [40], [2]. L'exemple donné dans chacune de ces références est un couple de langages métalinéaires dont l'intersection n'est pas « context-free ». Il s'ensuit donc que *Supp.* ($L_m{}^+$) n'est pas non plus fermé pour l'intersection, donc pour la complémentation. On peut obtenir un résultat plus fort qui couvre les grammaires linéaires, et d'ailleurs (pour l'intersection) même les grammaires linéaires à un seul non-terminal.

Pour cela, considérons les grammaires G_1 et G_2 définies respectivement par (45) et (46) :

$$\alpha = aa\alpha c + b\alpha c + bc \tag{45}$$

$$\alpha = a\alpha cc + a\alpha b + ab. \tag{46}$$

G_1 et G_2 sont chacune linéaires à un seul non-terminal, mais l'intersection des langages qu'elles engendrent est l'ensemble des séquences $\{a^{2n}b^na^{2n}\}$ qui n'est pas « context-free ». Cet exemple (plus le fait que ces familles, sont fermées pour la réunion) établit la /

PROPRIÉTÉ 2.

Les familles *Supp.* (L^+), *Supp.* ($L_m{}^+$), *Supp.* (C^+) ne sont fermées ni pour l'intersection, ni pour la complémentation; l'intersection de deux ensembles dans une de ces familles peut n'être même pas dans *Supp.* (C^+), même quand les ensembles en question sont engendrés par des grammaires à un seul non-terminal.

On peut penser que le complément d'un langage de *Supp.* (L^+) ou de *Supp.* ($L_m{}^+$) n'est pas un C-langage (i. e. n'est pas un élément de *Supp.* ($\bar{C}^+$)). Cependant, nous n'avons pas d'exemple qui indique ceci.

Donc, parmi les classes de langages considérés ci-dessus, seuls les événements réguliers (et les ensemble finis) sont fermés par formation d'intersections. Cependant, l'intersection d'un événement régulier et d'un C-langage est encore un C-langage. Nous avons en fait le résultat suivant plus fort et qui étend un théorème d'analyse classique dû à R. Jungen [20].

THÉORÈME 2.

Supposons que $r_1 \varepsilon \lambda_0{}^+$. Soit U^+ une des familles P^+, $\lambda_0{}^+$, λ^+, $\lambda_m{}^+$, C^+. Soit $r_1 \odot r_2$ le produit d'Hadamard de r_1, r_2 (cf. § 3.1). Alors $r_1 \odot r_2 \varepsilon U^+$, pour tout $r_2 \varepsilon U^+$. De plus si r_2, $r_3 \varepsilon \lambda_0{}^+$, alors $r_2 \odot r_3 \varepsilon \lambda_0{}^+$.

Cf. Schützenberger [46]. Il s'ensuit que l'intersection d'un langage de *Supp.* (U^+) avec un événement régulier est dans *Supp.* (U^+) pour tout U^+. La démonstration de ce résultat, lié à un résultat analogue sur la clôture par transduction, sera esquissée au § 8, ci-dessous.

4.3. La catégorie des grammaires linéaires est particulièrement intéressante, comme nous le verrons directement, et nous allons maintenant faire quelques observations préliminaires à son sujet.

Remarquons que si L est un langage engendré par une grammaire linéaire, on peut trouver un vocabulaire V', disjoint de V_T, deux homomorphismes α, α' de $F(V')$ dans $F(V_T)$, un événement régulier R dans V' et un ensemble fini $C \subset F(V_T)$ tel que L se compose exactement des séquences $f = \alpha(g)c\alpha'(\tilde{g})$, où $g \varepsilon R$, $\tilde{g}$ est la réflexion de g et $c \varepsilon C$. Ainsi un processus fini appliqué à une collection de couples de séquences ou un couple de processus finis coordonnés peut, en général, être rattaché à une grammaire linéaire et étudié de cette façon.

D'une manière équivalente, on peut caractériser un langage linéaire de la façon suivante légèrement différente. Soit $V' = V^+ \cup V^- (V^+ = \{ \nu_i : 0 \leqslant i \leqslant n \}; V^- = \{ \nu_i : -n \leqslant i \leqslant -1 \}$. Si $f \varepsilon F(V^+)$, définissons $\tilde{f}$ comme le résultat de la substitution de ν_{-i} à ν_i dans f, partout. Alors un langage linéaire L est déterminé par le choix d'un homomorphisme β de $F(V')$ dans $F(V_T)$, un événement régulier R dans V^+, et un ensemble fini $C \subset F(V_T)$. L est maintenant l'ensemble de séquences $\beta(f)c\beta(\tilde{f})$, où $f \varepsilon R$ et $c \varepsilon C$. Nous utiliserons cette deuxième caractérisation ci-dessous.

Nous pouvons maintenant déterminer des classes spéciales de langages linéaires en imposant des conditions supplémentaires à l'événement régulier R correspondant, aux applications α, α', et à la classe C. En particulier dans les applications ci-dessous nous nous intéresserons au cas où R, est simplement un monoïde libre (un événement régulier défini par un automate à un seul état) et où C contient seulement $c \varepsilon V_T$, où $\alpha(f) \neq \varphi c \psi \neq \alpha'(f)$. Nous appellerons les grammaires définies par cette condition des *grammaires linéaires minimales.*

Une grammaire linéaire minimale contient un seul symbole non terminal S et une seule règle terminale $S \to c$, et aucune règle non terminale $S \to \varphi c \psi$. Ainsi toute séquence du langage qu'elle engendre a le « marqueur central » désigné c. Ceci est l'ensemble de langages le plus simple de notre cadre, après les événements réguliers et nous verrons qu'ils diffèrent nettement des événements réguliers par beaucoup de propriétés formelles.

Donnons tout de suite un résultat sur les grammaires linéaires mini-

males qui sera utilisé par la suite. Prenons V', $V_T = W \cup \{ c \}$ $(c \notin W)$, α et α' comme ci-dessus. Soit G la grammaire linéaire minimale définie par α, α' et engendrant $L(G)$. Ainsi G a l'équation de définition :

$$\beta = c + \Sigma \{ \alpha(v)\beta\alpha'(v) : v \in V' \} \tag{47}$$

où α, α' sont des applications de $F(V')$ dans $F(W)$. Alors on a :

THÉORÈME 3.

Si α est un monomorphisme (isomorphisme dans), le complémentaire $F(V_T) \setminus L(G)$ *de* $L(G)$ *par rapport à* $F(V_T)$ *est engendré par une grammaire linéaire non-ambiguë.*

Démonstration : Soit $A = \alpha(V')$, $F(A) = \alpha F(V')$, et pour tout ensemble $F \subset F(W)$ soit $F^+ = \{ f \in F : f \neq e \}$.

Il est clair qu'il y a une partition : $F(V_T) \setminus L(G) = L \cup L'$, telle que

$$L = fcf' : f \in F^+(A),\ f' \in F(W),\ fcf' \notin L(G); \tag{48}$$

$L' = F(W) \cup cF(W) \cup ((F(W) \setminus F(A))cf(W) \cup F(V_T)cF(V_T)cF(V_T)$. Mais L' est un événement régulier. Il suffit donc de montrer que L est engendré par une grammaire linéaire non ambiguë.

Puisque α est un monomorphisme, il existe un isomorphisme $\bar{\alpha}$: $F(A) \to F(V')$. On étend α' à $F^+(A)$ en définissant $\alpha' a = \alpha'(\bar{\alpha}\, a)$, pour $a \in F^+(A)$.

Supposons que $acf' \in L$. Ainsi $a \in F^+(A)$, $f' \in F(W)$ et $f' \neq \alpha' a$. Par définition il n'y a que trois possibilités pour acf' et elles s'excluent mutuellement.

(49) (i) $f' \in F^+(W)\alpha' a$
(ii) $\alpha' a \in F^+(W)f'$
(iii) $a = a_1 a_2 a_3$ et $f' = hwg\alpha' a_1$ (où $a_1, a_3 \in F(A)$; $a_2 \in A$; $w \in W$; $h, g \in F(W)$; $\alpha' a_2 \in F^+(W)g$; $\alpha' a_3 \in F(W)wg$).

(49i) est le cas où f' a $\alpha' a$ comme facteur propre à droite.

(49ii) est le cas où $\alpha' a$ a f' comme facteur propre à droite.

(49iii) est le cas où $\alpha' a$ et f' ont le même facteur maximal à droite la séquence $g\alpha' a_1$, qui est une sous-séquence propre de $\alpha' a$ et f' à la fois.

Les trois cas s'excluent donc deux à deux et sont exhaustifs; on a une partition de L en trois sous-ensembles L_1, L_2, L_3, constitués respectivement des séquences qui vérifient (i), (ii), (iii). Ce qu'il nous reste à montrer, c'est que chacun des langages L_1, L_2, L_3 est engendré par une grammaire linéaire non ambiguë.

Pour L_1 et L_2 c'est évident. Soit $\overline{A} = \Sigma \{ a : a \in A \}$ et $\overline{W} = \Sigma \{ w :$

$w \varepsilon W \}$. Alors L_1 est engendré par la grammaire (50) et L_2 par la grammaire (51) (cf. § 3.2).

(50) $$\beta = \Sigma \{ a\beta\alpha' a : a \varepsilon A \} + c(1 - \overline{W})^{-1}$$

(51) $$\beta = \Sigma \{ a\beta\alpha' a : a \varepsilon A \} + (1 - \overline{A})^{-1} c.$$

Envisageons maintenant le cas de L_3. Pour tout $a \varepsilon A$, soit $B(a)$ l'ensemble de toutes les séquences wg ($w \varepsilon W$, $g \varepsilon F(W)$) telles que $\alpha' a \varepsilon F^+(W)$ g et $\alpha' a \notin (W)$ wg. Il est clair que $B(a)$ est toujours un ensemble fini, puisque g est plus court que $\alpha' a$. On peut alors engendrer L_3 par grammaire linéaire non-ambiguë d'équations :

(52) $$\beta_1 = \Sigma \{ a\beta_1\alpha' a : a \varepsilon A \} + \Sigma \{ a\beta_2 b : a \varepsilon A, b \varepsilon B(a) \}$$
$$\beta_2 = c + c(1 - \overline{W})^{-1} + (1 - \overline{A})^{-1} c + \Sigma \{ a\beta_2 w : a \varepsilon A, w \varepsilon W \}.$$

La vérification est sans problème. Mais alors $F(V_T) \setminus L(G)$ se trouve être exprimé comme la réunion de quatre ensembles disjoints L_1, L_2, L_3, L', qui ont chacun une grammaire linéaire non-ambiguë. Il en résulte que $F(V_T) \setminus L(G)$ a lui-même une grammaire linéaire non ambiguë, ce qu'on voulait démontrer.

Remarquons que si on avait pris pour α au départ « une transduction sans perte d'information » [43] au lieu d'un monomorphisme, on aurait pu démontrer un résultat où la seule différence avec le théorème 3 serait que la grammaire linéaire construite aurait une ambiguïté bornée au lieu d'être non ambiguë.

4.4. Nous avons considéré plusieurs sous-familles de la classe des C-grammaires, en les classant d'après les propriétés structurales des règles de définition. On peut envisager d'autres principes de classification. Ainsi par exemple, il serait peut-être utile d'isoler la classe de grammaires (langages) étoile caractérisée de la façon suivante : G est une grammaire étoile si à chaque non-terminal α_i de G sont associés un ensemble Σ_i de non-terminaux et trois séquences terminales f_i, f'_i, f''_i, et G contient toutes les règles suivantes à l'exclusion de toute autre : $\alpha_i \to f''_i$, $\alpha_i \to f_i \alpha_j f'_i$ ($\alpha_j \varepsilon \Sigma_i$), $\alpha_j \to \alpha_k \alpha_l$ (α_j, α_k, $\alpha_l \varepsilon \Sigma_i$). Ce sont, en un sens, les C-grammaires les « moins structurées ». L'intérêt de ces langages vient du fait que les équations définissant les séries de puissances associées sont exprimables en n'utilisant de manière essentielle, que le quasi-inverse et l'addition, comme nous l'avons remarqué au § 3.2. Remarquons en particulier, que le langage non-métalinéaire L_{IC} défini par (42) est un langage étoile. Nous avons suggéré une interprétation linguistique de la notion de langage étoile au § 3.2.

Un autre principe de classification pourrait résider dans le nombre des non-terminaux de la grammaire définissante minimale d'une certaine série de puissances. Cependant, il ne paraît pas probable que les propriétés intéressantes du langage puissent être reliées à une mesure aussi peu sensible aux traits structuraux des grammaires (sauf dans le cas particu-

lier des langages définis par des grammaires à un seul non-terminal), car pour les monoïdes différents des groupes, il n'y a pas de relation intéressante entre les paramètres numériques grossiers, et la structure fine. Remarquons, d'ailleurs, que pour tout N fini, on peut construire un événement régulier qui ne peut être engendré par une C-grammaire à moins de N symboles non-terminaux.

La considération des inter-dépendances entre les différentes parties d'une grammaire suggère un autre principe de classification. Disons qu'une C-grammaire est *irréductible*, si aucun sous-ensemble propre de l'ensemble des équations de définition ne constitue une C-grammaire (rappelons que des séquences terminales doivent être dérivables de chaque symbole non terminal non-initial d'une C-grammaire); dans le cas contraire la grammaire sera dite *réductible*. Si une C-grammaire est réductible dans ce sens, il doit y avoir des sous-ensembles propres Σ_1 de ses règles et Σ_2 de ses non-terminaux, tels que seules des règles de Σ_1 interviennent dans la transformation des dérivations en dérivations terminales aux points où des symboles de Σ_2 apparaissent dans les lignes de dérivations.

Un cas extrêmement particulier de la réductibilité a été étudié par Ginsburg et Rice [18]. Comme eux, disons qu'une C-grammaire G est séquentielle si ses non-terminaux peuvent être ordonnés en une suite $\alpha_1, \ldots, \alpha_n$ (où α_1 est le symbole initial) de telle façon qu'il n'y ait aucune règle $\alpha_i \to \varphi\alpha_j\psi$ pour $j < i$. La solution d'une grammaire séquentielle est particulièrement facile à déterminer par le processus itératif décrit au § 2.3, par élimination successive des variables.

Pour la famille S^+ des grammaires séquentielles et pour la famille *Supp.* (S^+) de leurs supports, Ginsburg et Rice établissent les résultats suivants, analogues aux précédents. D'abord il est clair que S^+, comme C^+ est un demi-anneau fermé par quasi-inversion d'éléments quasi-réguliers. Corrélativement *Supp.* (S^+) est fermé par réunion, produit et opération étoile. De ce fait, et du fait que $P^+ \subset S^+$ il résulte que *Supp.* $(L_0{}^+) \subset$ *Supp.* (S^+). De plus, l'inclusion est stricte, comme le montre la grammaire (42) qui, puisqu'elle ne contient qu'un seul terminal, est séquentielle. On a donc :

$$\textit{Supp.}\ (L_0{}^+) \subsetneq \textit{Supp.}\ (S^+) \subsetneq \textit{Supp.}\ (C^+). \tag{53}$$

Ginsburg et Rice montrent qu'il n'y a pas de grammaire séquentielle pour le langage de vocabulaire $\{a, b, c, d$, et qui contient les séquences :

$$a^{n_{2k-1}}d \ldots db^{n_2}da^{n_1}db^{n_2}d \ldots b^{n_{2k-2}}da^{n_{2k-1}} \tag{54}$$

(qui est symétrique par rapport à c) pour toute suite $(k, n_1, \ldots, n_{2k-1})$ d'entiers positifs, bien que ce langage soit engendré par la grammaire

$$\begin{aligned} \alpha &= ad\beta da + a\alpha a + aca \\ \beta &= b\beta b + bd\alpha db. \end{aligned} \tag{55}$$

Il n'existe cependant pas de relation plus forte que (53) entre *Supp.* (S^+) et les familles de *Propriété 1*, § *4.1.* La grammaire (55) est en fait linéaire, mais non séquentielle, si bien que *Supp.* $(L^+) \not\subset$ *Supp.* (S^+); et la grammaire (42) est séquentielle mais non méta-linéaire, si bien que *Supp.* $(S^+) \not\subset$ *Supp.* $(L_m{}^+)$.

Les grammaires (45) et (46) étant séquentielles, on voit que la *Propriété 2* (mais non le *théorème 2*) peut se généraliser à *Supp.* (S^+). Pour d'autres résultats sur les langages séquentiels, voir Ginsburg and Rose [19], Shamir [51].

5. Une autre caractérisation des familles de C-langages.

Dans cette section, nous aborderons d'une façon assez différente la définition des familles de langages et nous montrerons en quoi elle se rattache à la classification présentée ci-dessus. Nous nous appuierons ici sur deux notions fondamentales : celles d'*événement régulier standard* et de *langage de Dyck*, que nous définissons maintenant.

Un *événement régulier standard* A est donné par un alphabet fini χ, deux sous-ensembles J_1 et J_2 de (X, X) et la règle que $f \varepsilon A$ si et seulement si

(56) (i) $\quad f \varepsilon xF(X) \cap F(X)x'$, où $(x, x') \varepsilon J_1$

(ii) $\quad f \notin F(X)xx'F(X)$, où $(x, x') \varepsilon J_2$.

A est ainsi l'ensemble de toutes les séquences qui commencent et finissent par des lettres imposées et qui ne contiennent aucun couple de lettres consécutives appartenant à J_2. C'est, plus techniquement, l'intersection du quasi-idéal déterminé par J_1 avec le complémentaire de l'idéal bilatère engendré par tous les produits xx' $((x, x') \varepsilon J_2)$. A est en particulier ce qu'on appelle parfois un *événement 1-défini* [21] [35].

Le langage de Dyck D_{2n} est défini sur les $2n$ lettres $x_{\pm i}$ $(1 \leqslant i \leqslant n)$ comme l'ensemble de toutes les séquences f qui se réduisent à la suite vide par la suppression réitérée de couples de lettres consécutives $x_j x_{-j}$ $(-n \leqslant j \leqslant n)$. Le langage de Dyck est un objet mathématique très connu : si φ est l'homomorphisme du monoïde libre engendré par $\{x_{\pm i}\}$ sur le groupe libre engendré par le sous-ensemble $\{x_i\} : i > 0$ qui satisfait identiquement $(\varphi x_i)^{-1} = \varphi x_{-i}$, D_{2n} est le noyau de φ, c'est-à-dire l'ensemble de séquences f telles que $\varphi f = 1$.

En ce qui concerne ces notions, nous avons les résultats suivants.

Proposition 1.

Pour tout événement régulier $BC \subset (Z)$, on peut trouver un événement régulier standard A et un homomorphisme $\alpha : F(X) \to F(Z)$, tels que $B = \alpha A$.

Il est bon de remarquer qu'on peut choisir cette représentation de telle façon que non seulement $B = \alpha A$ mais encore que chaque séquence

$f \varepsilon A$ ait le même degré d'ambiguïté que la séquence correspondante $\alpha f \varepsilon B$. C'est-à-dire que si $B = Supp.\ (\beta)$ on peut trouver γ tel que $A = Supp.\ (\gamma)$ et pour tout f, $< , \alpha f > = < \beta, \alpha f >$.

On peut généraliser la *Proposition 1* aux C-langages, en utilisant la propriété suivante de D_{2n}.

PROPRIÉTÉ 1.

D_{2n} *est engendré par une C-grammaire non-ambiguë.*

Pour obtenir une grammaire non-ambiguë de D_{2n}, on introduit $2n + 1$ non-terminaux : $\alpha_{\pm i} (1 \leqslant i \leqslant n)$ et β. Considérons maintenant les $2n + 1$ équations

$$(57)\ (i) \qquad \alpha_i = x_i (1 - \Sigma_{j \neq -i} \alpha_j)^{-1} x_{-i}$$

$$(ii) \qquad \beta = (1 - \Sigma \alpha_i)^{-1}$$

(Cf. § 3.2, pour les notations.)

Intuitivement, on peut interpréter β comme la somme de toutes les séquences qui peuvent se réduire à la séquence vide par suppression de deux lettres consécutives $x_i x_{-i}$ Chaque α_i est la somme de tous les mots de *Supp.* (β) qui commencent par x_i et qui n'ont pas de facteur propre à gauche (ou à droite) dans *Supp.* (β). L'équation (57i) implique que tout $f \varepsilon$ *Supp.* (α_i) admet une factorisation et une seule

$$(58) \qquad f = x_i f_1 f_2 \ldots f_m x_{-i}$$

où chaque f_j appartient à un ensemble bien défini *Supp.* (α_i) (où j n'est pas $-i$, car nous voulons que la lettre initiale x_i ne s'annule qu'avec la lettre finale x_{-i}). De la même manière, chaque $f \varepsilon$ *Supp.* (β) admet une factorisation et une seule $f = f_1 \ldots f_m$, où les f_j appartiennent à $\cap$ *Supp.* (α_i).

Nous avons maintenant le résultat suivant, analogue à la *proposition 1.*

PROPOSITION 2.

Tout C-langage $L \subset F(Z)$ *est donné par un entier n, un événement régulier standard* A *sur* $X_{2n} = \{ x_{\pm i};\ 1 \leqslant i \leqslant n \}$, *un homomorphisme* φ : $F(X_{2n}) \to F(Z)$ et la règle $L = \varphi (A \cap D_{2n})$. [48] [49] [11] [12].

De nouveau, comme ci-dessus, l'énoncé implique que les séquences sont produites avec l'ambiguïté convenable. D'autre part, il est possible de choisir J_1 tel que $(x, x') \varepsilon J_1$ si x appartient à un certain sous-ensemble de X (cf. [48]).

On peut définir des familles particulières de langages comme celles qui viennent d'être envisagées, en imposant des conditions à l'événement régulier standard A et à l'homomorphisme φ. Ainsi, supposons qu'on

prenne l'événement régulier standard A sur l'alphabet $X \cup Y$ (où $X = \{ x_{\pm i};\ 1 \leqslant i \leqslant n \}$, $Y = \{ y_{\pm i};\ 1 \leqslant i \leqslant m \}$), défini par les conditions suivantes sur J_1 et J_2 :

$$(59)\quad \begin{aligned} J_1 &= \{ (x_j, x_i) : i > 0 \} \\ J_2 &= \{ (x_j, x_i) : \text{signe } (i) \neq \text{signe } (j) \} \cup \{ (y_i, y_j) : i < 0 \text{ ou } j > 0 \} \cup \\ &\qquad \{ (x_i, y_j) : i < 0 \text{ ou } j < 0 \} \cup \{ (y_i, x_j) : i > 0 \text{ ou } j > 0 \}. \end{aligned}$$

Ainsi chaque séquence a la forme $fgg'f'$, où $f, f' \in F(X)$; $g, g' \in F(Y)$; f, g (respectivement f', g') ne contiennent que des lettres d'indices positifs (respectivement négatifs). Si on désigne par X^+ et X^- les sous-ensembles de X constitués de lettres à indices positifs et négatifs, respectivement (de même Y^+ et Y^-) on peut représenter les transitions permises et exclues, par la matrice (60), où l'entrée *1* (*0*) indique que la transition de l'élément en tête d'une rangée à celui en tête d'une colonne est (n'est pas) permise et où U est une matrice de *1* et *O* une matrice de zéros.

(60)

	Y^+	Y^-	X^+	X^-
Y^+	*U*	*U*	*0*	*0*
Y^-	*0*	*U*	*0*	*U*
X^+	*U*	*0*	*U*	*0*
X^-	*0*	*0*	*0*	*U*

Mais considérons alors l'ensemble $A \cap D_{XY}$ (où D_{XY} est le langage de Dyck sur l'alphabet $X \cup Y$). Si $fg \in A$ (où $f \in F(X^+ \cup Y^+)$, $g \in F(X^- \cup Y^-)$), remplit la condition supplémentaire $fg \in D_{XY}$, g doit être l'image miroir de f (au changement près du signe des indices). C'est-à-dire, avec les notations du deuxième paragraphe de 4.3, $g = f$. Il est clair que si α est un homomorphisme de $F(X \cup Y)$ dans $F(V_T)$, $\alpha(A \cap D_{XY})$ est un langage linéaire. De plus, si on ajoute la condition $y_i = e$ pour $i < 0$ et $y_i = c$ pour tout $i > 0$, où $\alpha x_i \notin F(V_T)cF(V_T)$ pour tout i, alors $L = \alpha(A \cap D_{XY})$ est un langage linéaire minimal avec c comme symbole central désigné, et tout langage linéaire minimal est obtenu par un tel choix de α. Ceci donne une caractérisation indépendante des langages linéaires minimaux.

De plus, en ajoutant des couples supplémentaires à J_2, on peut délimiter le langage canonique A défini de telle façon que $\{ f : f \in F(X^+)$ et il existe g tel que $fg \in A$ et $g \in F(Y)F(X) \}$ soit un événement régulier quelconque (et non plus simplement le monoïde libre sur X^+, comme auparavant), si bien que $L = \alpha(A \cap D_{XY})$ sera un langage linéaire quelconque. On a ainsi une définition indépendante de la notion de « langage linéaire ». (Remarquons que ces restrictions supplémentaires sur J_2, ne jouent que sur les transitions permises dans les matrices de la diagonale principale de (60).)

D'une façon tout à fait semblable, on peut donner une définition générale d'un « langage métalinéaire ». Ainsi, par exemple, considérons le langage métalinéaire particulier engendré par la grammaire d'équations :

$$(61) \qquad \begin{aligned} \xi_i &= \xi_1 \xi_2 \\ \xi_1 &= e + \Sigma \{ a\xi_1 b : a, b \in V_T \} \\ \xi_2 &= e + \Sigma \{ a\xi_2 b : a, b \in V_T \}. \end{aligned}$$

Dans ce cas la matrice de l'événement régulier standard sous-jacent serait :

(62)

	X_1^+	X_1^-	X_2^+	X_2^-
X_1^+	U	U	0	0
X_1^-	0	U	U	0
X_2^+	0	0	U	U
X_2^-	0	0	0	U

Tous les langages métalinéaires et ceux-là seulement, seront basés sur un événement standard avec une matrice essentiellement de ce type (avec, peut-être, des restrictions supplémentaires sur la diagonale principale).

Les *propositions* 1 et 2 fournissent ainsi la possibilité de définir très naturellement la classe entière des C-langages, ainsi que différentes sous-familles de cette classe, indépendamment de la démarche adoptée dans les sections précédentes.

6. Indécidabilité.

6.1. Il est démontré dans Post [36] que le problème suivant, connu sous le nom de *problème de correspondance*, est récursivement insoluble. Si $\Sigma = (f_1, g_1), \ldots (f_n, g_n)$ est une suite de couples de séquences, nous dirons qu'une suite $I = (i_1, \ldots, i_m)$ d'entiers $(1 \leqslant i_j \leqslant n)$ satisfait Σ si :

$$(63) \qquad f_{i_1} \cdots f_{i_m} = g_{i_1} \cdots g_{i_m}.$$

Le problème de correspondance consiste à demander si, étant donnée la condition Σ, il existe une suite d'indices qui y satisfasse. Remarquons qu'ou bien Σ n'est satisfaite par aucune suite d'indices, ou bien par une infinité, puisque si $(i_1, \ldots, i_m)$ satisfait Σ, c'est encore le cas pour $(i_1, \ldots, i_m, i_1, \ldots, i_m)$. Post a montré qu'il n'existe pas d'algorithme qui détermine, pour Σ quelconque s'il n'y a pas de suite d'indices satisfaisant Σ, ou s'il y en a une infinité, ce qui constitue les deux seules possibilités.

On peut reformuler directement le problème de correspondance en termes de grammaires linéaires minimales. Étant donnée $\Sigma = \{ (f_1, g_1), \ldots,$

$(f_n, g_n)\}$, formons $G(\Sigma)$ avec l'unique non-terminal S et l'équation de définition :

$$S = a + f_1 S \tilde{g}_1 + \ldots + f_n S \tilde{g}_n \tag{64}$$

où a est un symbole qui ne figure dans aucun des f_i ou des g_i. Il est clair qu'il n'y a une suite d'indices satisfaisant Σ que si $G(\Sigma)$ engendre une séquence $fa\tilde{f}$. Ou, en d'autres termes, soit L_m le langage « image-miroir » constitué de toutes les séquences $fa\tilde{f}$, $f \in F(V_T)$ et soit $L(G(\Sigma))$ le langage engendré par G. Alors, ou bien il n'y a aucune suite d'indices qui satisfasse Σ auquel cas $L_m \cap L(G(\Sigma))$ est vide; ou bien il y en a une infinité, auquel cas $L_m \cap L(G(\Sigma))$ est infini. Du fait que le problème de correspondance est indécidable et que L_m est engendré par une grammaire linéaire à un seul non-terminal, on tire aussitôt que :

THÉORÈME D'INDÉCIDABILITÉ 1.

Étant données deux grammaires G_1 *et* G_2 *engendrant respectivement* L_1 *et* L_2 *il n'existe pas d'algorithme qui détermine si* $L_1 \cap L_2$ *est vide ou infini. Ceci est vrai, même si* G_1 *et* G_2 *sont des grammaires linéaires minimales et si* G_1 *est une grammaire particulière fixée de* L_m.

On voit facilement que le problème de savoir si l'intersection est vide ou finie est décidable pour les grammaires linéaires unilatères, mais dans notre cadre, pour les grammaires les plus simples, dont la capacité génératrice dépasse celle des événements réguliers, ces problèmes ne sont plus décidables.

Cette remarque est généralisée dans Bar-Hillel, Perles, Shamir [2], où l'on montre que beaucoup de problèmes relatifs aux C-grammaires sont récursivement indécidables. Leur méthode est en gros la suivante : limitons V_T à l'ensemble $\{a, 0, 1\}$. Si $\Sigma = \{(f_1, g_1), \ldots, (f_n, g_n)\}$ est un ensemble de couples de séquences dans le vocabulaire $\{0, 1\}$ (i. e., $f_i, g_i \in F\{0, 1\}$), soit $L(\Sigma)$ l'ensemble de toutes les séquences :

$$10^{i_k} \ldots 10^{i_1} a f^{i_1} \ldots f_{i_k} a \tilde{g}_{j_l} \ldots g_{j_1} a 0^{j_1} 1 \ldots 0^{j_l} 1, \tag{65}$$

où $1 \leqslant i_{i_1}, \ldots, i_k, j_1, \ldots, j_l \leqslant n$.

Pour plus de clarté, utilisons $\bar{i} = 01^i$ pour coder le nombre i. Une séquence de $L(\Sigma)$ est alors formée en choisissant des suites d'indices $I = (i_1, \ldots, i_k)$ et $J = (j_1, \ldots, j_l)$, et en formant :

$$\bar{i}_k \ldots \bar{i}_1 a f_{i_1} \ldots f_{i_k} a \tilde{g}_{j_l} \ldots \tilde{g}_{j_1} a \bar{j}_1 \ldots \bar{j}_l. \tag{66}$$

$L(\Sigma)$ joue a ors le même rôle que le langage engendré par (64) dans la démonstration précédente du *Théorème d'indécidabilité 1.* C'est évidemment un C-langage (engendré, d'ailleurs, par une grammaire méta-linéaire qui est une variante évidente de (64)). Mais le Théorème 3, § 4.3 ci-dessus

entraîne aussitôt que le complémentaire $F(V_T) \setminus L(\Sigma)$ de $L(\Sigma)$ par rapport au vocabulaire V_T est un C-langage et qu'on peut construire sa grammaire à partir de la grammaire de $L(\Sigma)$. (Remarquons, en fait, qu'on aurait pu utiliser n'importe quel code au lieu du choix particulier $\bar{i} = 01^i$, pour définir $L(\Sigma)$.)

Au lieu du langage image-miroir L_m utilisé dans la démonstration du *Théorème d'indécidabilité 1*, considérons le langage L_{dm} « double image miroir » constitué par toutes les séquences :

(67) $\quad x_1 a x_2 a \tilde{x}_2 a \tilde{x}_1$, où x_1 et x_2 sont des séquences de $\{ 0, 1 \}$.

Il n'est pas difficile de montrer que L_{dm} et son complémentaire par rapport à V_T sont tous deux des C-langages. Remarquons que

(68) $\quad L(\Sigma) \cap L_{dm} = \{ \bar{i}_k \ldots \bar{i}_1 a f_{i1} \ldots f_{ik} a \tilde{g}_{ik} \ldots \tilde{g}_{i1} a \bar{i}_1 \ldots \bar{i}_k \}$
où $(i_1, \ldots, i_k)$ satisfait Σ (c'est-à-dire où $f_{i1} \ldots f_{ik} = g_{i1} \ldots g_{ik}$).

Remarquons aussi qu'un ensemble infini de séquences de la forme (68) ne peut constituer un C-langage (ni, a fortiori, un événement régulier).

Supposons maintenant qu'il existe une solution positive au problème de correspondance pour Σ; c'est-à-dire qu'il y ait une suite d'indices satisfaisant Σ. Alors, comme nous l'avons signalé, il existe une infinité de telles séquences. Par suite $L(\Sigma) \cap L_{dm}$ est infini. Ce n'est donc ni un événement régulier ni un C-langage.

Supposons au contraire, qu'il n'y ait pas de suite d'indices satisfaisant Σ. Alors $L(\Sigma) \cap L_{dm}$ est vide; c'est donc à la fois un événement régulier et un C-langage; et, Σ étant fixée, on peut construire leurs C-grammaires $G(\Sigma)$ et G_{dm} (qui sont, en fait, métalinéaires). Ainsi, s'il y avait un algorithme permettant de déterminer si l'intersection de langages engendrés par deux C-grammaires G_1 et G_2 est vide, finie, ou est un événement régulier, ou un C-langage, cet algorithme fournirait aussi une solution du problème général de correspondance. Nous en concluons :

THÉORÈME D'INDÉCIDABILITÉ 2.

Étant données des C-grammaires G_1 *et* G_2, *il n'y a pas d'algorithme qui détermine si l'intersection des langages qu'elles engendrent est vide, finie, un événement régulier, où un C-langage. Ceci reste vrai, en particulier, quand toutes deux sont métalinéaires et quand* G_2 *est une grammaire fixée de* L_{dm}.

Soit $\overline{G}_{dm}$ la C-grammaire qui entendre le complémentaire $\overline{L}_{dm}$ (tous les complémentaires sont désormais pris par rapport à V_T) de L_{dm}. Et, étant donnée Σ soit $\overline{G}(\Sigma)$ la C-grammaire qui entendre le complémentaire $\overline{L(\Sigma)}$ de $L(\Sigma)$, et dont l'existence est garantie par le théorème 3, § 4.3. Considérons maintenant la grammaire G qui engendre le langage $L(G) = \overline{L}_{dm} \cup \overline{L(\Sigma)}$. Il est clair que G est une C-grammaire et peut se construire à partir de G_{dm} et $G(\Sigma)$. Mais le complémentaire $\overline{L(G)}$ de $L(G)$ est l'ensemble $L_{dm} \cup L(\Sigma) = L_{dm} \cap L(\Sigma)$ et l'on sait d'après le *Théorème*

d'indécidabilité 2 qu'il n'y a pas d'algorithme qui détermine, quand Σ est donnée, si cet ensemble est vide, fini, un événement régulier ou un C-langage. Mais, étant donnée Σ, G est déterminée comme étant une C-grammaire. On a donc :

THÉORÈME D'INDÉCIDABILITÉ 3.

Il n'y a pas d'algorithme qui détermine sur la donnée d'une C-grammaire G, si le complément du langage engendré par G *est vide, fini, un événement régulier ou un C-langage.*

Il n'existe pas, en particulier, de procédé général pour déterminer si la C-grammaire G engendre le langage universel $F(V_T)$ ou si G engendre un événement régulier (puisque le complémentaire d'un événement régulier est un événement régulier).

Par suite, il n'y a pas d'algorithme qui détermine, sur la donnée de C-langages L_1 et L_2, s'il existe ou non un transducteur appliquant L_1 sur L_2, puisque tous les langages réguliers, et eux seuls, peuvent s'obtenir par transduction à partir du C-langage $F(V_T)$. (Ginsburg et Rose, communication personnelle).

Il n'y a, d'autre part, aucune méthode générale qui permette de savoir si deux C-grammaires sont équivalentes, c'est-à-dire si elles engendrent le même langage, puisqu'une telle méthode pourrait servir à décider si une C-grammaire G est équivalente à la grammaire G_U qui engendre $F(V_T)$. Il s'ensuit aussi immédiatement qu'étant données deux C-grammaires, il n'y a pas d'algorithme permettant de décider si le langage engendré par l'une contient le langage engendré par l'autre, puisque ceci fournirait une solution du problème d'équivalence.

Ces résultats ont été esquissés pour des langages construits à partir d'un vocabulaire V_T à trois éléments, mais il est clair qu'avec un recodage convenable ils s'appliquent encore aux langages sur un vocabulaire de deux lettres ou plus. Ceci est explicité en détail dans Bar-Hillel, Perles, Shamir [2].

6.2. Nous avons observé au § 4 que les processus finis faisant intervenir des couples de séquences pouvaient être formulés d'une façon naturelle, en termes de grammaires linéaires. En particulier, comme on vient de le voir, le problème de correspondance peut se décrire directement comme problème de grammaires linéaires minimales. Il en est de même pour un second problème combinatoire, également dû à Post appelé le problème de l'étiquette (« Tag problem »).

On peut énoncer une forme générale de ce problème de la manière suivante. Soit W l'ensemble des séquences (monoïde libre) sur un vocabulaire fini quelconque, et P un sous-ensemble fini de séquences non nulles de W remplissant la condition qu'aucune séquence de W n'a plus d'un facteur à gauche dans P. C'est-à-dire qu'il n'existe pas de séquences $p_1, p_2, w_1, w_2, w_3 (p_i \varepsilon P, w_i \varepsilon W)$ telles que $p_1 \neq p_2$ et $w_1 = p_1 w_2 = p_2 w_3$.

Soit V l'ensemble des séquences de W sans facteur à gauche dans P. C'est-à-dire, $v \varepsilon V$ si et seulement s'il n'existe aucun $p \varepsilon P$ tel que $v = pw$, pour $w \varepsilon W$. V est manifestement un ensemble récursif, d'ailleurs régulier. Soit α une application de P dans W(α définit donc un ensemble de couples de séquences (p, w), où $w = \alpha p$, $p \varepsilon P$, $w \varepsilon W$). Définissons une application T sur W, où :

$$(69) \qquad \begin{aligned} Tf &= f'\alpha p \text{ si } f = pf' \\ Tf &= H \quad \text{si } f \varepsilon V \ (H \varepsilon W). \end{aligned}$$

Considérons le problème :

(70) étant donnée une séquence f, existe-t-il un entier n tel que $T^n f = H$?

En considérant que T définit le calcul d'une machine de Turing, (70) est le *problème de l'arrêt* pour cette machine.

Minsky [30] a montré que (70) est un problème récursivement indécidable.

Le problème de l'étiquette, tel qu'il est formulé par Post, est un cas particulier de (70), ci-dessus, où T remplit les conditions supplémentaires suivantes : P est l'ensemble de toutes les séquences de longueur k, pour $k \geqslant 2$, fixé quelconque; αp ne dépend que du symbole le plus à gauche de p. Même avec cette restriction, le problème (70) est insoluble, comme l'a montré Minsky. Ce résultat est assez surprenant du fait du caractère déterministique *(monogène)* du processus générateur T.

Un pas dans la direction d'une reformulation du problème de l'étiquette généralisé en termes de grammaires linéaires minimales, est de l'énoncer de la manière suivante. W, P, V, α, T étant donnés comme ci-dessus, la question (70) ne peut recevoir de réponse affirmative que si

$$(71) \qquad \begin{aligned} &\text{il y a des séquences } p_1 \ldots p_n \varepsilon P \text{ et } v \varepsilon V \text{ telles que} \\ &\qquad p_1 \ldots p_n v = f\alpha p_1 \ldots \alpha p_n. \end{aligned}$$

On peut alors réénoncer le problème de l'étiquette généralisé, sous la forme du problème suivant relatif aux grammaires linéaires. Étant donnés W, P, V, α, T, définissons la grammaire G engendrant $L(G)$ par la seule équation

$$(72) \qquad S = \Sigma_i \nu_i c + \Sigma_i (p_i \tilde{S} \alpha p_i)$$

où $\nu_i \varepsilon V$, $p \varepsilon P$, et $c \notin W$ est le marqueur central distingué. Définissons le langage $M(f) = \{ fgc\tilde{g} \}$ (ainsi $M(f) = fL_m$; où L_m est le langage « image miroir » défini plus haut). Alors la réponse à (71) (ou, ce qui revient au même, à (70)) est affirmative si et seulement si l'intersection de $L(G)$ et $M(f)$ est non vide. On voit ainsi qu'il n'existe pas d'algorithme pour déterminer, pour f fixé, si le langage $M(f)$ a ou non une intersection non vide avec un langage, dont la grammaire correspond à (72) (même dans le cas

particulier où P est l'ensemble de toutes les séquences de longueur k, k fixé $\geqslant 2$, et où αp ne dépend que de la lettre de P la plus à gauche).

Remarquons que le *Théorème d'Indécidabilité 1*, ci-dessus, résulte aussi directement de l'insolubilité du problème de l'étiquette. En effet le problème de correspondance et celui de l'étiquette sont tous deux relatifs à la cardinalité de l'intersection d'un langage linéaire minimal L avec les langages $M(f)$, où $f = e$ et L quelconque, dans le cas du problème de correspondance, tandis que f est quelconque et L remplit la condition (72), ci-dessus, pour le problème de l'étiquette.

7. Ambiguïté.

7.1. Nous avons défini une série de puissances caractéristiques r, comme ayant chaque coefficient $< r, f >$ égal à zéro ou un. Nous dirons qu'une C-grammaire est *non ambiguë*, si le terme principal de sa solution est une série de puissances caractéristiques. Dans ce cas, chaque phrase engendrée, l'est avec une seule description structurale et le « déparenthésage » n'apporte aucune ambiguïté. Disons qu'un C-langage est *intrinsèquement ambigu*, si toutes ses C-grammaires sont ambiguës.

Il est bien connu qu'aucun événement régulier n'est intrinsèquement ambigu : tout événement régulier est le support d'une série de puissances caractéristiques qui est le terme principal de la solution d'une grammaire linéaire unilatère [14], [37]. Cependant, cette remarque ne s'étend pas à la classe entière des C-grammaires. Parikh [34] a montré qu'il existe des C-langages inhéremment ambigus. Un exemple de langage inhéremment ambigu est l'ensemble :

$$\{ a^n b^m c^p : n = m \text{ ou } m = p \}. \tag{73}$$

Dans ce cas les séquences de la forme $a^n b^n c^n$ doivent avoir une ambiguïté au moins égale à 2 dans toute C-grammaire engendrant (73) (et il existe une C-grammaire engendrant (73) dans laquelle leur ambiguïté est exactement égale à 2).

Nous n'avons pas d'exemples qui montrent l'étendue de l'ambiguïté inhérente dans les C-langages, ou des types particulier de C-langages.

Remarquons qu'une conséquence immédiate du *Théorème d'Indécidabilité 1* du § 6, est qu'il ne peut exister d'algorithme déterminant si une C-grammaire, ou même une grammaire linéaire est ou non ambiguë. Supposons d'ailleurs, comme plus haut, que $\Sigma = \{ (f_1, g_1), \ldots (f_n, g_n) \}$ est une suite de couples de séquences. Choisissons $n + 1$ symboles nouveaux, $x_0, \ldots, x_n$ et construisons les grammaires G_f de règles :

$$S_f \to x_0,\ S_f \to x_i S_f \tilde{f}_i \quad (1 \leqslant i \leqslant n)$$

et G_g de règles :

$$S_g \to x_0,\ S_g \to x_i S_g g_i \quad (1 \leqslant i \leqslant n).$$

Il est clair que G et G_g sont non-ambiguës et le problème de correspondance pour Σ a une solution affirmative, si et seulement s'il existe une séquence engendrée à la fois par G_f et G_g c'est-à-dire si et seulement si la grammaire G_{fg} est ambiguë, G_g contenant les règles de G_f, de G_g et les règles $S \to S_f$, $S \to S_g$, où S est le symbole initial de G_{fg}. Donc il ne peut y avoir aucune procédure pour décider si, étant donné Σ quelconque, la grammaire G_{fg} associée ainsi à Σ est non ambiguë.

La grammaire G_{fg} est linéaire, elle a trois non-terminaux et un marqueur central, et l'on voit que pour cette classe de grammaires, le problème de l'ambiguïté est indécidable.

On peut penser que cette remarque s'étend aux grammaires à deux non-terminaux. En revanche, la question intéressante de savoir si le problème de l'ambiguïté est encore indécidable pour les grammaires linéaires minimales reste ouverte. Résumons la question de l'ambiguïté telle qu'elle se présente à l'heure actuelle, par les résultats suivants.

THÉORÈME D'AMBIGUÏTÉ 1.

Il existe des C-langages inhéremment ambigus.

THÉORÈME D'AMBIGUÏTÉ 2.

Il n'existe pas d'algorithme permettant de savoir si une C-grammaire (même linéaire, à marqueur central désigné), est, ou non, ambiguë.

8. Transduction finie.

Nous voulons décrire une famille particulièrement simple de transformations d'un langage en un autre. La première, et aussi la plus importante, est l'*homomorphisme.*

Soit L un langage quelconque sur un vocabulaire terminal Z et supposons que pour tout $z \varepsilon Z$, on se donne un langage L sur un deuxième vocabulaire X. Notons par ΘL l'ensemble de toutes les séquences (sur X) qu'on obtient en prenant un mot $g = z_{i_1} z_{i_2} \ldots z_{i_m} \varepsilon L$ et en remplaçant chaque z_{i_j} par un mot arbitraire de $L_{z_{i_j}}$. Le nom d' « homomorphisme » explique en soi la transformation. En effet, si on considère les anneaux $A(Z)$ et $A(X)$ des séries formelles de puissances par rapport aux variables $z \varepsilon Z$ et $x \varepsilon X$, et si on note par Θ, l'homomorphisme de $A(Z)$ dans $A(X)$ induit par l'application $\Theta z =$ la série formelle de puissances associées à L_z, alors ΘL est le support de l'image par Θ de la série formelle de puissance associée à L.

Une interprétation en rapport avec notre démarche présente peut être donnée si les L_z sont des C-langages. Dans ce cas, supposons que L est produit par la C-grammaire G_z(de vocabulaire non-terminal Y), et que chaque L_z est produit par la C-grammaire G_z (avec l'ensemble des non-terminaux Y_z et la lettre initiale $y_{z,0}$). Supposons que les ensembles Y

soient disjoints et considérons une C-grammaire $\overline{G}$ de non-terminaux $Y \cup Z \cup \bigcup_{z \varepsilon Z} Y_z$ constituée des règles de G, des G_z, et des règles $z \to y_{z,o}$ ($z \varepsilon Z$). (Plus simplement, nous identifions chaque z à $y_{z,o}$). Il est clair que G produit exactement ΘL.

Généralisons maintenant cette construction au type suivant de relation contextuelle : soient $R_i (i \varepsilon I)$ et $R_{i'} (i' \varepsilon I')$ deux familles finies d'événements réguliers telles que chaque $g \varepsilon F(Z)$ appartient à un élément et un seul de chaque famille. Supposons aussi que pour chaque triplet ($z \varepsilon Z$, $i \varepsilon I$, $i' \varepsilon I'$), on ait un langage $L_{z,i,i'}$ dans le vocabulaire X.

Alors pour tout $y = z_{j_1} z_{j_2} \ldots z_{j_k}$ on remplace chaque z_{j_i} par une séquence arbitraire du langage $L(z_k, i, i')$ où i et i' sont déterminés par la condition que la séquence $z_{j_1} z_{j_2} \ldots z_{j_{k-1}}$ est dans R_i et que la séquence $z_{j_1} \ldots z_{j_k}$ est dans $R_{i'}$. Il est facile de voir qu'on peut, sans diminuer la généralité, supposer que pour toute séquence g appartenant à un ensemble quelconque R_{i_1} et pour $z \varepsilon Z$, l'ensemble R_{i_z} contenant gz ne dépend que de l'indice i_1, et de la lettre z. Autrement dit, on peut supposer donnés un ensemble d'états I, une application de transition $I \times Z \to I$ et un état initial $i_o \varepsilon I$, tels que $z_{j_1} \ldots z_{j_{k-1}} \varepsilon R_i$ si et seulement si i est l'état atteint à partir de i_o et après lecture de $z_{j_1} z_{j_2} \ldots z_{j_{k-1}}$.

Une construction analogue s'applique à R_i et pour des raisons de clarté, on écrira l'application correspondante sous la forme d'une multiplication à gauche. Étant données les deux applications $I \times Z \to I$ et $Z \times I' \to I'$, on notera σg, pour tout $g = z_{j_1} z_{j_2} \ldots z_{j_m} \varepsilon F(Z)$, la séquence de triplets :

$$(74) \quad (i_1, z_{j_1}, i'_m)\,(i_2, z_{j_2}, i'_{m-1}) \ldots (i_k, z_{j_k}, i'_{m-k+1}) \ldots (i_m, z_{j_m}, i'_1)$$

où, par récurrence :

$$(75) \quad i'_2\ i_2 = i_1 z_{j_1}, i_3 = i_2 z_{j_2}, \ldots, i_m = i_{m-1} z_{j_m},\ i_k = i_{k-1} z_{j_{k-1}} \text{ et } = z_{j_m} i'_1,\ i'_3 = z_{j_{m-1}} i'_2, \ldots, i'_m = z_{j_2} i'_{m-1}.$$

Avec ces notations on peut considérer les transformations décrites comme se déroulant en deux étapes :

(76) (i) remplacement de chaque $g \varepsilon L$ par la séquence $\sigma g = (i, z_{j_1}, i_m) \ldots (i_m, z_{j_m}, i')$ dans un alphabet U consistant en des triplets (i, z, i');
(ii) remplacement dans σg de chaque triplet $(i_k, z_{j_k}, i'_{m-k+1})$ par une séquence quelconque du langage $L(z_{jk}, i_k, i'_{m-k+1})$.

Puisque la deuxième étape n'est qu'un homomorphisme, il suffit de discuter la première. Pour cela, soit U l'ensemble de tous les triplets (i, z, i') et considérons le langage L' obtenu à partir de L en ajoutant à sa grammaire toutes les règles $z_j \to (i, z_j, i')$ (avec $i \varepsilon I$, $i' \varepsilon I'$, quelconques).

Il est clair qu'une séquence de L' appartient à l'ensemble $\{\sigma g : g \in L\}$ si et seulement si elle satisfait la condition (75) ci-dessus, ou encore si elle appartient à l'événement régulier $\overline{R}$ défini par la condition (75) sur l'ensemble $F(U)$ de toutes les séquences dans l'alphabet U.

La première étape consiste donc seulement en un homomorphisme de L dans l'ensemble de toutes les séquences sur U (ce qui donne L') suivi de l'intersection de L' avec un événement régulier.

Donnons maintenant une dernière interprétation de ce que nous avons fait : pour tout $z \in Z$, soit μz la matrice dont les lignes et les colonnes sont indexées par des couples ($i \notin I$, $i' \notin I'$), et dont les entrées sont les suivantes :

(77) $\mu z_{(i,i')\,(i'',i''')}$ = le triplet (i, z, i''') si $i'' = iz$ et $i' = z\,i''' = o$ autrement.

Si on calcule alors :
$\mu z_j\, \mu z_{j_2} \ldots \mu z_{j_n} = \mu g$, il est facile de vérifier que l'entrée $(i, i'_m)\,(i_m, i'_1)$ de μg est justement σg. De là il s'ensuit que $\{\sigma g : g \in\} L = L' \cap \overline{R}$ est aussi un C-langage. μ est effectivement un homomorphisme, on remplace chaque non-terminal y par une matrice μy dont les entrées sont de nouveaux non-terminaux et on vérifie que μ commute avec les substitutions utilisées pour définir le langage comme solution d'un système d'équations. Par ailleurs l'identification des entrées dans l'image μ de nos équations donne un nouveau système d'équations du type habituel qui définit exactement $L' \cap \overline{R}$ [46]. Plus simplement encore, on peut définir μ' comme plus haut à ceci près que pour chaque entrée non-nulle on prend la série formelle de puissances $L(z_j, i, i')$ au lieu du triplet (i_1, z_j, i'). Les deux étapes de la construction sont alors télescopées en une seule, et la série de puissances associée au langage (sur X) obtenue par notre transformation est simplement une entrée de

$$\Sigma \{ \mu' g : g \in L \}. \tag{78}$$

Ceci est le fondement de la démonstration du Théorème 2, § 4, plus haut.

9. Rapports avec la théorie des automates.

Jusqu'à présent, nous avons étudié des processus générateurs, les langages et les systèmes de descriptions structurales qu'ils définissent, et certaines applications finies sur ces langages, d'un point de vue entièrement abstrait. Pour relier ces remarques à la théorie des automates, il est commode d'introduire une assymétrie temporelle dans nos considérations.

Un automate M peut être considéré comme un dispositif consistant en un ensemble d'états Σ (mémoire de M) qui accepte (ou ce qui revient

au même produit) une suite de symboles d'un vocabulaire (alphabet) V, selon des instructions fixées, énonçables de manière finie (qu'on peut donner en associant à chaque $v \varepsilon V$ une application φ_v de Σ dans lui-même (ou dans l'ensemble des parties de Σ, dans le cas d'un automate « non déterministique »). Si on prend un état initial et un ensemble d'états final, on définit un langage $M(L)$ constitué de séquences pouvant être acceptées par M, quand il fonctionne selon ses instructions de l'état initial à l'état final, allant de $S \varepsilon \Sigma$ à $S' \varepsilon \Sigma$, en acceptant v, seulement si $\varphi_v(S) = S'$ (ou $S' \varepsilon \varphi_v(S)$, dans le cas non déterministique).

La taille de la mémoire de M, ou sa vitesse de croissance au cours du calcul, fournit un certain indice de richesse de langage $L(M)$, au moyen duquel on peut comparer diverses familles de langages du type qui vient d'être envisagé.

Étant donné un ensemble de séquences L, écrivons $f \sim f'$, si pour tout g, $fg \varepsilon L$ si et seulement si $f'g \varepsilon L$. Il est clair que $\sim$ est une équivalence. De plus on peut manifestement prendre les classes d'équivalence définies par $\sim$ comme états d'un automate $M(L)$ qui accepte L, puisque toute l'information sur f servant à la poursuite du calcul de $M(L)$, f étant lue, est donnée par la classe d'équivalence de f.

Remarquons que L est la réunion de certaines de ces classes d'équivalence, et que $f \sim f'$ implique que $fg \sim f'g$ pour tout g.

En second lieu, étant donné L, écrivons $f \equiv f'$ si et seulement si pour tout g, $gf \sim gf'$ il est clair que $f \equiv f'$ si et seulement si pour tout g, g', $gfg' \varepsilon L$ si et seulement si $gf'g' \varepsilon L$. Ainsi $\equiv$ est symétrique et il est facile de montrer que $f_1 \equiv f_2$ et $f_3 \equiv f_4 \equiv$ est donc une relation de congruence et les $\equiv$ — classes de l'ensemble $F(V)$ peuvent être multipliées entre elles ce qui donne un monoïde quotient de $F(V)$. Ce monoïde quotient $F'(V) = \varphi F(V)$ est tel que $L = \varphi^{-1}\varphi L$ et il est associé canoniquement à L_0 [41].

Cette remarque rattache la théorie présentée ici à celle des monoïdes. L'intérêt est que dans certains cas les $\sim$ -classes (et le monoïde quotient) ont une interprétation simple traduisible en langage d'automates, et inversement que certaines notions algébriques (en particulier, celle d'*extension*), reçoivent une interprétation simple.

Revenons maintenant au problème de la caractérisation des familles de langages en termes d'automates; il est bien connu que la sous-famille *Supp.* (L_0^+) des C-langages est caractérisée de façon unique par le fait que pour chaque langage $L \varepsilon$ *Supp.* (L_0^+), il existe un automate $M(L)$ à mémoire bornée, qui accepte L.

Considérons maintenant la famille *Supp.* (L_0), c'est-à-dire l'ensemble des supports des séries de puissance qui sont solutions de systèmes d'équations « linéaires unilatères » à coefficients entiers positifs ou négatifs.

Comme nous l'avons vu $L \varepsilon$ *Supp.* (L_0), si et seulement si $L =$ *Supp.* $(r_1 - r_2)$ où $r_1, r_2 \notin (L_0^+)$. On peut alors montrer que les énoncés suivants sont équivalents.

(79) (i) $L \varepsilon$ *Supp.* $(L_0{}^+)$;
(ii) il existe une correspondance biunivoque entre les $\sim$-classes pour L et un espace de dimension finie de vecteurs entiers $\nu(f)$ tels que pour tout $x \varepsilon V$, $\nu(fx) = \nu(f)\mu x$, où μx est une matrice;
(iii) $F/\equiv$ est isomorphe à un monoïde de matrices entières de dimension finie (les matrices μ de (ii);
(iv) L est accepté par un automate $M(L)$, dont la mémoire est un espace de dimension finie de vecteurs à coordonnées entières, et les transitions sont comme dans (ii), ci-dessus.

(Schützenberger, [44], la classe d'automates A étant celle qui est définie par (79 iv).)

Considérons maintenant les deux restrictions suivantes sur la classe d'automates A,

(80) (i) il existe un N tel que pour tout $f \varepsilon F(V)$, $|| \nu(f) || < N$;
(ii) pour tout $f, f', f'' \varepsilon F(V)$ et $\varepsilon > 0$, $\lim_{n\to\infty} e^{-\varepsilon n} || \nu(f'f^n f'') || = 0$

où $|| \nu ||$ est la longueur du vecteur ν au sens habituel.

Il est clair que (80 i) entraîne (80 ii). De plus L est manifestement un événement régulier ($L \varepsilon$ *Supp.* $(L_0{}^+)$ seulement si (80 i) est remplie par un automate de la classe A qui accepte L. Un automate de la classe A qui remplit la condition (80 ii) est appelé un automate fini à compteur dans Schützenberger [47] où de tels dispositifs sont étudiés. On peut montrer qu'en gros (80 ii) signifie que la quantité d'information (en bits) enregistrée dans la mémoire ne croît pas plus vite qu'une fonction linéaire du logarithme de la longueur d'un mot d'entrée.

Il est intéressant de remarquer que (comme dans le cas de la classe totale des C-grammaires), il n'y a pas d'algorithme qui permette de décider, étant donné $M \varepsilon A$, s'il y a ou non un f accepté par M [28]. De plus, on peut facilement montrer que le même problème pour les automates finis à compteur est indécidable, si le dixième problème de Hilbert (existence d'une solution entière pour une équation diophantienne quelconque) est indécidable [47].

Considérons maintenant un automate M qui présente une structure du type suivant : les états de M (les $\sim$-classes dans le langage d'entrée de M) sont identifiés aux séquences dans un certain alphabet nouveau (« interne ») et pour tout $\nu \varepsilon V$, « l'instruction de calcul » φ_ν, application : [$\sim$-classe de f]$\to$[$\sim$-classe de $f\nu$] consiste en l'addition ou la suppression de lettres à l'extrémité droite de la séquence interne associée à [$\sim$-classe de f]. Nous appellerons un tel automate (en accord avec la terminologie habituelle), un automate à pile. Les automates à pile forment une sous-classe restreinte de la classe des automates linéaires bornés étudiés par Myhill [31], Ritchie [39].

Si M est un automate à pile, le langage L qu'il accepte est un

C-langage et tout C-langage peut s'obtenir par un homomorphisme à partir d'un langage accepté par un automate à pile [48], [49]. En particulier si D est un langage de Dyck et A un événement régulier standard (cf. § 5) $D \cap A$ est accepté par un automate à pile.

Un automate à pile non déterministique est un automate du type ci-dessus, à ceci près que φ_v applique un état dans un ensemble d'états. Nous pouvons maintenant démontrer directement que les C-langages (les langages de la classe *Supp.* (C^+)) sont ceux qu'accepte un automate à pile non déterministique [11] [12].

Traduit par G. FAUCONNIER.

BIBLIOGRAPHIE

[1] AJDUKIEWICZ, K. Die Syntaktische Konnexität. *Studia Philosophica*, 1 (1935), 1-27.

[2] BAR-HILLEL, Y., PERLES, M. and SHAMIR, E. On formal properties of simple phrase structure grammars. *Tech. Report*, 4, July 1960.

[3] — Applied Logic Branch. The Hebrew University of Jerusalem. In *Zeit. für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, Band 14, Heft 2 (1961), 143-172.

[4] — Finite state languages. *Bull. Research Council Israel*, 8F (1960), 155-166.

[5] BIRKELAND, R. Sur la convergence de développements qui expriment les racines de l'équation algébrique générale. *C. R. Acad. Sciences*, 171 (1920), 1370-1372; 172 (1921), 309-311.

[6] ČULIK, K. Some notes on finite state languages. *Časopis pro pěstovani Mat.* (1961), 86, 43-55.

[7] CHOMSKY, N. Three models for the description of language. *I. R. E. Trans. PGIT*, 2 (1956), 113-124.

[8] — On certain formal properties of grammars. *Information and Control*, 2 (1959), 137-167.

[9] — A note on phrase structure grammars. *Information and Control*, 2 (1959), 393-395.

[10] — On the notion « Rule of Grammar ». *Proc. Symp. Applied Math.*, 12, Am. Math. Soc. (1961).

[11] — Context-free grammars and pushdown storage. *Quarterly Progress Reports*, n° 65, Research Laboratory of Electronics, M. I. T. (1962).

[12] — Formal properties of grammars, in Bush, Galanter, Luce (eds.) *Handbook of Mathematical Psychology*, vol. 2, Wiley (1963).

[13] — The logical basis for linguistic theory. *Proc IXth Int. Cong. Linguists.* Cambridge, Mass. (1962).

[14] CHOMSKY, N. and MILLER, G. A. Finite state languages. *Information and Control*, 1 (1958), 91-112.

[15] — Introduction to the formal analysis of natural languages, in Bush, Galanter, Luce (eds.), *Handbook of Mathematical Psychology*, vol. 2, Wiley (1963).

[16] DAVIS, M. *Computability and Unsolvability.* New York, McGraw-Hill (1958).

[17] ELGOT, C. C. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98 (1961), 21-51.

[18] GINSBURG, S. and RICE, H. G. Two families of languages related to ALGOL. *Technical Memorandum. Systems Development Corporation.* Santa Monica, California (1961).

[19] GINSBURG, S. and ROSE, G. F. Operations which preserve definability in languages. *Technical Memorandum. Systems Development Corporation.* Santa Monica, California (1961).

[20] JUNGEN, R. Sur les séries de Taylor n'ayant que des singularités algebrico-logarithmiques sur leur cercle de convergence. *Comm. Math. Helvetici*, 3 (1931), 286-306.

[21] KLEENE, S. C. Representation of events in nerve nets, and finite automata. *Automata Studies*, Princeton University Press (1956), 3-41.

[22] KULAGINA, O. Ob odnom sposobe apredelenija grammatičeskix ponjatij. *Problemy Kivernetiki*, 1. Moscou (1958).

[23] LAMBEK, J. The mathematics of sentence structure. *Am. J. Math.*, 65 (1958), 153-170.

[24] — On the calculus of syntactic types. *Proc. Symposium Applied Math.*, 12, Am.

[25] LYNDON, R. C. Equations in free groups. *Trans. Am. Math. Soc.*, 96 (1960), 445-457.

[26] MCNAUGHTON, R. The theory of automata, in *Advances in Computers*, vol II (Academic Press).

[27] MAHLER, K. On a theorem of Liouville in fields of positive characteristic. *Canadian J. of Math.*, 1 (1949), 397-400.

[28] MARKOV, A. A. ob odnoi nevazrešimoi probleme, *Doklady Akad. Nauk*, n. s. 78 (1951), 1089-1092.

[29] MILLER, G. A. and CHOMSKY, N. Finitary models of language users, in Bush, Galanter, Luce (eds.). *Handbook of Mathematical Psychology*, vol. 2, Wiley (1963).

[30] MINSKY, M. L. Recursive unsolvability of Post's problem of Tag. *Ann. of Math.*, 74 (1961), 437-455.

[31] MYHILL, J. Linear bounded automata. WADD *Tech. Note* 60-165. Wright Air Dvpt. Division. Wright Patterson Air Force Base Ohio (1960).

[32] NEWELL, A. and SHAW, J. C. Programming the logic theory machine. *Proc. Western Joint Computer Conference* (1957), 230.

[33] ŒTTINGER, A. G. Automatic syntactic analysis and the pushdown store. *Proc. of Symposia in Applied Math.*, 12, Am. Math. Soc. (1961).

[34] PARIKH, R. J. Language generating devices. *Quarterly Progress Report* nº 60, Research Laboratory of Electronics, M. I. T. January (1961), 199-212.

[35] PERLES, M. RABIN, M. O. and SHAMIR, E. The theory of definite automata. *Tech. Report*, nº 6, O. N. R. (1961).

[36] POST, E. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52 (1946), 264-268.

[37] RABIN, M. O. and SCOTT, D. Finite automata and their decision problems. I. B. M. *Journal of Research*, 3 (1959), 115-125.

[38] RANEY, G. N. Functional composition patterns and power-series reversion, *Trans. Am. Math. Soc.*, 94 (1960), 441-451.

[39] RITCHIE R. W. *Classes of recursive functions of predictable complexity.* Thèse, Dept. of Math, Princeton Univ. (1960).

[40] SCHEINBERG, S. Note on the Boolean properties of context-free languages. *Information and Control*, 3 (1960), 372-375.

[41] SCHÜTZENBERGER, M. P. On an application of semi-group methods. *I. R. E. Trans.*, IT2 (1956), 47-60.

[42] — Un problème de la théorie des automates. *Séminaire Dubreuil-Pisot.* Paris, déc. (1959).

[43] — A remark on finite transducers. *Information and Control*, 4 (1961), 185-196.

[44] — On the definition of a family of automata. *Information and Control*, 4 (1961), 245-270.

[45] — Some remarks on Chomsky's context-free languages. *Quarterly Progress Report*, nº 68, Research Laboratory of Electronics, M. I. T., oct. (1961).

[46] — On a theorem of R. Jungen. *Proc. Am. Math. Soc.*, 13 (1962), 885-890.

[47] — Finite counting automata. *Information and Control*, 5 (1962), 91-107.

[48] — On Context-free languages and pushdown storage. *Information and Control*, 6 (1963), 246-264.

[49] — Certain elementary families of automata. *Symp. on mathematical theory of automata*, Polytechnic Institute of Brooklyn, 1962.

[50] SHEPHERDSON, J. C., The reduction of two-way automata to one-way automata. I. B. M. *Journal of Research* (1959), 198-200.

[51] SHAMIR, E. On sequential languages. *Tech. Report*, nº 7, O. N. R. (1961).

[52] YAMADA, A. *Counting by a class of growing automata.* Thèse Univ. of Penna., Philadelphia (1960).