Grammaires formelles : Grammaires de type 1 et de type 0

Karën Fort

karen.fort@sorbonne-universite.fr / https://members.loria.fr/KFort/

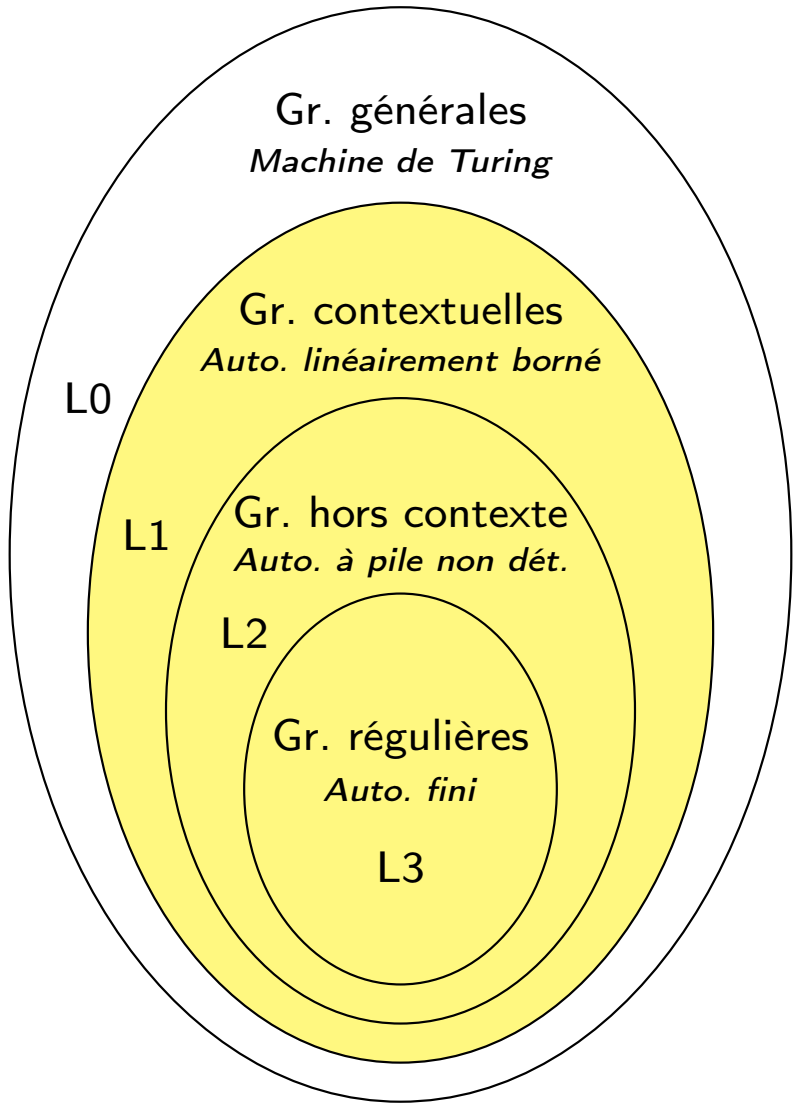
LETTRES SORBONNE UNIVERSITÉ

Quelques sources d'inspiration

par ordre d'importance décroissant

- cours de D. Battistelli (Paris 3), grâce aux notes de C. Riquier (Master 2, Paris 4)
- cours d'A. Rozenknop (Paris 13)
- cours de B. Habert (ENS de Lyon)
- *Introduction à la calculabilité* (Pierre Wolper) – InterEditions, 1991
- cours en ligne de J-F. Perrot (Paris 6), avec son accord : http://pagesperso-systeme.lip6.fr/Jean-Francois.Perrot/inalco/Automates/Cours17.html
- Wikipédia sur la hiérarchie de Chomsky http://fr.wikipedia.org/wiki/Hi%C3%A9rarchie_de_Chomsky

Place dans la hiérarchie



Type 1 : grammaires contextuelles (*context-sensitive*)

Définition

Les grammaires de type 1, dites grammaires contextuelles se définissent par des règles du type :

$$\alpha A \beta \longrightarrow \alpha a \beta \text{ avec } \begin{cases} \alpha, \beta, a \in (V_T \cup V_N)^* \\ a \neq \varepsilon \\ A \in V_N \end{cases}$$

La partie gauche est non vide et la partie droite doit contenir plus de symboles que la partie gauche (le nombre de symboles ne peut que croître dans une dérivation). Ces grammaires sont décidables.

Type 1 : grammaires contextuelles (*context-sensitive*)

Définition

Les grammaires de type 1, dites grammaires contextuelles se définissent par des règles du type :

$$\alpha A \beta \longrightarrow \alpha a \beta \text{ avec } \begin{cases} \alpha, \beta, a \in (V_T \cup V_N)^* \\ a \neq \varepsilon \\ A \in V_N \end{cases}$$

La partie gauche est non vide et la partie droite doit contenir plus de symboles que la partie gauche (le nombre de symboles ne peut que croître dans une dérivation). Ces grammaires sont décidables.

L'appellation "contextuelle" vient de ce que les mots α et β sont interprétés comme le contexte dans lequel le non-terminal A peut se réécrire en a.

Exemple de grammaire contextuelle

$$P = \begin{cases} S \longrightarrow aSBC & (1) \\ S \longrightarrow aBC & (2) \\ CB \longrightarrow HB & (3) \\ HB \longrightarrow HC & (4) \\ HC \longrightarrow BC & (5) \\ aB \longrightarrow ab & (6) \\ bB \longrightarrow bb & (7) \\ bC \longrightarrow bc & (8) \\ cC \longrightarrow cc & (9) \end{cases}$$

 Quel est le langage engendré par cette grammaire ?

Exemple de grammaire contextuelle

$$P = \begin{cases} S \longrightarrow aSBC & (1) \\ S \longrightarrow aBC & (2) \\ CB \longrightarrow HB & (3) \\ HB \longrightarrow HC & (4) \\ HC \longrightarrow BC & (5) \\ aB \longrightarrow ab & (6) \\ bB \longrightarrow bb & (7) \\ bC \longrightarrow bc & (8) \\ cC \longrightarrow cc & (9) \end{cases}$$

Quel est le langage engendré par cette grammaire ?

$\{a^n b^n c^n | n > 0\}$

Exemple de grammaire contextuelle

$$P = \begin{cases} S \longrightarrow aSBC & (1) \\ S \longrightarrow aBC & (2) \\ CB \longrightarrow HB & (3) \\ HB \longrightarrow HC & (4) \\ HC \longrightarrow BC & (5) \\ aB \longrightarrow ab & (6) \\ bB \longrightarrow bb & (7) \\ bC \longrightarrow bc & (8) \\ cC \longrightarrow cc & (9) \end{cases}$$


Quel est le langage engendré par cette grammaire ?


Dériver "aaabbbccc"

Structure des grammaires contextuelles

Types variés, y compris des graphes.

Automates linéairement bornés

Définition

« La classe d'automates associée aux grammaires CS est celle des machines de Turing auxquelles on impose que la quantité de mémoire utilisée par le calcul (la longueur de bande nécessaire) ne croisse que de manière linéaire avec la longueur du mot donné (et non pas de manière quadratique ou - pire - exponentielle).
On notera que cette contrainte sur la quantité de mémoire utilisée ne dit rien sur le temps de calcul, qui peut être arbitrairement grand. »
[JF Perrot, cours 17]

Expressivité des grammaires contextuelles

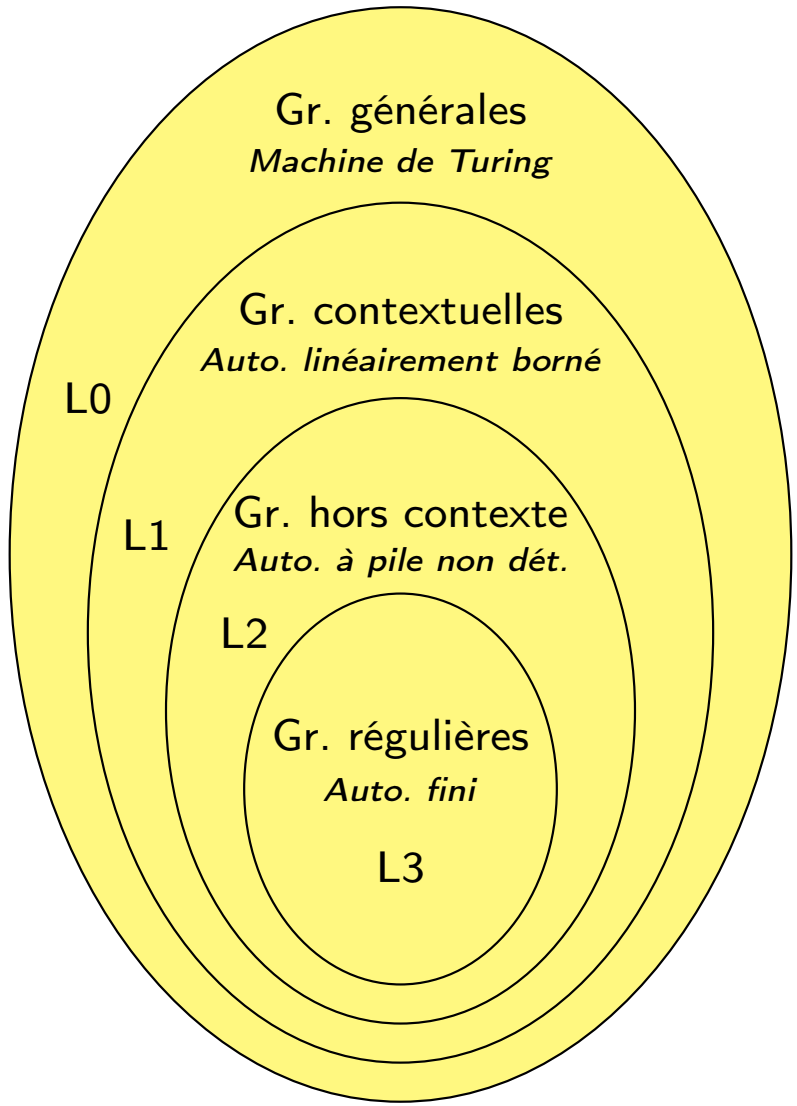
Les grammaires contextuelles sont adaptées pour exprimer les croisements (*respectivement*)

Applications

Les grammaires sur lesquelles portent la majorité des recherches en TAL se situent entre le type 1 et le type 2.

Il existe des algorithmes d'analyse syntaxique (*parsing*) qui permettent de dire si une chaîne appartient ou non au langage défini par la grammaire : ces méthodes existent pour les grammaires de type 1, 2 et 3 mais pas pour les grammaires de type 0 (on parle d'indécidabilité)

Place dans la hiérarchie



Type 0 : grammaires non-contraintes

Définition

Les grammaires de type 0, dites grammaires générales ou non-contraintes, se définissent par des règles du type (aucune restriction n'est imposée aux règles) :

$$\alpha \longrightarrow \beta \text{ avec } \begin{cases} \alpha \in V^* \\ \beta \in V^* \end{cases}$$

Ces grammaires ne sont pas décidables : il n'existe pas d'algorithme qui puisse déterminer si une chaîne appartient au langage

La principale cause de cette indécidabilité est qu'on puisse avoir des dérivations raccourcissantes

Exemple de grammaire non-contrainte

$$P = \begin{cases} S \longrightarrow aSDE \mid \varepsilon & (1) \\ aD \longrightarrow ab & (2) \\ bE \longrightarrow bc & (3) \\ cD \longrightarrow DE & (4) \\ bD \longrightarrow bb & (5) \\ cE \longrightarrow cc & (6) \end{cases}$$

Exemple de grammaire non-contrainte

$$P = \begin{cases} S \longrightarrow aSDE \mid \varepsilon & (1) \\ aD \longrightarrow ab & (2) \\ bE \longrightarrow bc & (3) \\ cD \longrightarrow DE & (4) \\ bD \longrightarrow bb & (5) \\ cE \longrightarrow cc & (6) \end{cases}$$

Cette grammaire :

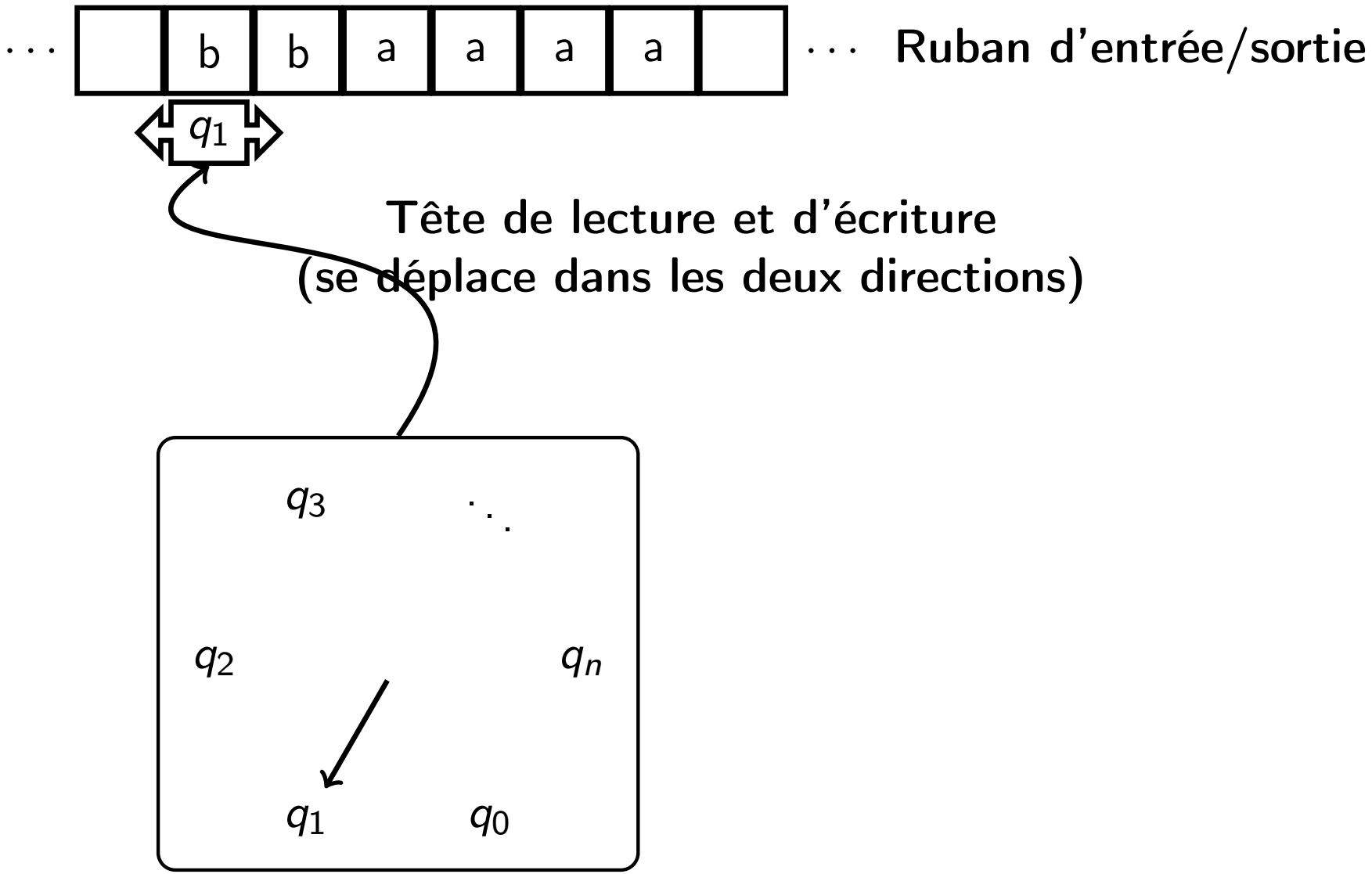
- n'est pas hors-contexte : plusieurs règles comportent plusieurs symboles en partie gauche
- n'est pas contextuelle : dans la règle $cD \longrightarrow DE$ on ne retrouve pas le contexte gauche (c) en partie droite de la règle

Structure des grammaires non-contraintes

Les possibilités de description structurales sont très variées et difficiles à représenter par un graphe :

- des nœuds peuvent apparaître puis disparaître dans l'élaboration du graphe
- un élément terminal qui apparaît en cours de dérivation ne subsiste pas forcément jusqu'à la fin

Machine de Turing



Machine de Turing en vidéos

Explications :
https://interstices.info/jcms/nn_72391/comment-fonctionne-une-machine-de-turing

Illustration :
https://www.dailymotion.com/video/xrn0yi

Turing et la calculabilité

Calculabilité

« On a démontré que le modèle de la machine de Turing était une des formalisations de la notion de calculabilité : tout ce qui est calculable peut être calculé par une machine de Turing »

[JF Perrot, cours 17]

Turing, en même temps que Church, mais d'une toute autre manière, démontre (par la machine de Turing), qu'il y a des limites à la calculabilité

Types de grammaires formelles

Grammaire	Automate	Règles de production
Type 0	Machine de Turing	$\alpha \longrightarrow \beta$
Type 1	Automate linéairement borné	$\alpha A \beta \longrightarrow \alpha a \beta$
Type 2	Automate à pile non déterministe	$A \longrightarrow \alpha$
Type 3	Automate fini	$A \longrightarrow w\ B$, $A \longrightarrow w$

Point sur **les** récursivités

Si $A \Longrightarrow^* \varphi A \psi$, alors A est un élément récursif de la grammaire

- si $A \Longrightarrow \varphi A \psi$, on parle de récursivité directe (sinon indirecte)
- si $\varphi = \varepsilon$ et $\psi \neq \varepsilon$ alors A est récursif à gauche
- si $\varphi \neq \varepsilon$ et $\psi = \varepsilon$ alors A est récursif à droite
- si $\varphi \neq \varepsilon$ et $\psi \neq \varepsilon$ alors on parle d'auto-imbrication

Exercice sur **les** récursivités

$$P = \begin{cases} S \longrightarrow ABC & (1) \\ A \longrightarrow aAa & (2) \\ B \longrightarrow bB & (3) \\ C \longrightarrow Dd & (4) \\ D \longrightarrow Ca & (5) \\ C \longrightarrow c & (6) \\ A \longrightarrow a & (7) \\ B \longrightarrow b & (8) \\ D \longrightarrow d & (9) \end{cases}$$

Quels sont les éléments récursifs et de quel type sont-ils ?

Quel est le type de la grammaire ?

Règles et grammaires

Types de règles

Les restrictions apportées aux règles sont de plus en plus sévères depuis le type 0 jusqu'au type 3

Types de grammaires

Il est très fréquent qu'une même grammaire ait des règles de types différents.

Le type d'une grammaire est celui de sa ou de ses règles les plus hautes dans la hiérarchie des règles.

Langages

Types de langage

Théoriquement, un langage est de type 1 ssi il existe une grammaire de type 1 au sens strict qui peut l'engendrer et si l'on peut prouver qu'il est impossible d'engendrer ce langage avec une grammaire de type 1 +1.

S'il est facile de satisfaire la première condition, la 2e est quasiment impossible à satisfaire.

$\Rightarrow$ notion impossible à déterminer

Propriétés de fermeture

Fermeture

On dit qu'il y a fermeture quand le langage résultat d'une opération est toujours de même type que les langages avec lesquels on a fait l'opération.

Type de langage	$\cup$	$\cap$	complément	produit	*	inverse
0	oui	oui	non	oui	oui	oui
1	oui	oui	?	oui	oui	oui
2	oui	non	non	oui	oui	oui
3	oui	oui	oui	oui	oui	oui

Des grammaires « intermédiaires »

Wikipédia, Hiérarchie_de_Chomsky

- entre le type 0 et le type 1 : les langages récursifs acceptés par les machines de Turing qui s'arrêtent toujours
- entre le type 1 et le type 2 : les langages à grammaires indexées, définis par des grammaires plus générales que les grammaires contextuelles
- entre le type 2 et le type 3 : les langages algébriques déterministes, pour lesquels il existe une caractérisation par automate, mais pas par les grammaires

Grammaires contextuelles :

- définition
- structure, limites
- récursivité

Grammaires non-contraintes :

- définition
- structure, limites
- récursivité

Exercice 1 : à faire, à rendre **avant** la fin du TD, noté

De quel type sont les (règles de) grammaires suivantes :

1. $S \longrightarrow B$
2. $S \longrightarrow bS$
3. $bS \longrightarrow S$
4. $S \longrightarrow SB$
5. $SS \longrightarrow SB$
6. $SS \longrightarrow SaBC$
7. $S \longrightarrow bb$
8. $SS \longrightarrow b$
9. $W \longrightarrow e$
10. $WVC \longrightarrow e$
11. $wvc \longrightarrow e$
12. $WvW \longrightarrow WvyzW$

Exercice 2 : à faire, à rendre **avant** la fin du TD, noté

Montrer par l'exemple que la grammaire suivante engendre $\{a^i b^j c^k | 1 \leq i \leq j \leq k\}$:

$$P = \begin{cases} S \longrightarrow aTbX & (1) \\ S \longrightarrow abX & (2) \\ T \longrightarrow aTbC & (3) \\ T \longrightarrow TbC & (4) \\ T \longrightarrow TC & (5) \\ T \longrightarrow bC & (6) \\ T \longrightarrow C & (7) \\ Cb \longrightarrow bC & (8) \\ CX \longrightarrow Xc & (9) \\ X \longrightarrow c & (10) \end{cases}$$

Exercice 3 : à faire, à rendre **avant** la fin du TD, noté

Quel est le langage engendré par la grammaire suivante :

$$P = \begin{cases} S \longrightarrow AB & (1) \\ A \longrightarrow aAX & (2) \\ A \longrightarrow aX & (3) \\ B \longrightarrow bBd & (4) \\ B \longrightarrow bYd & (5) \\ Xb \longrightarrow bX & (6) \\ XY \longrightarrow Yc & (7) \\ Y \longrightarrow \varepsilon & (8) \end{cases}$$

Exercice 4 : à faire, à rendre **avant** la fin du TD, noté

On considère le langage L des mots sur $\{a, b, c\}$ qui contiennent au moins une fois la chaîne bac.

 Définir formellement L et construire une grammaire hors-contexte puis une grammaire régulière décrivant L