INF 302 : Langages & Automates

Chapitre 5 : Grammaires et grammaires régulières

Yliès Falcone
ylies.falcone@univ-grenoble-alpes.fr
www.ylies.fr

Univ. Grenoble-Alpes, Inria
-
Laboratoire d'Informatique de Grenoble - www.liglab.fr
Équipe de recherche LIG-Inria, Corse - team.inria.fr/corse/

Année Académique 2017 - 2018

Plan Chap. 5 - Grammaires et grammaires régulières

À propos des grammaires
Description informelle

Les grammaires permettent de décrire des langages de manière *générative*, en décrivant de *manière compacte* les différentes manières de générer *un mot*.

Les grammaires utilisent des *règles de production* permettant de décrire les transformations d'une partie d'un mot en cours de génération :

$$\text{Source} \rightarrow \text{Résultat}$$

À partir de sources, nous produisons des résultats (dont certains peuvent être ensuite utilisés comme sources).

Quand une source produit plusieurs résultats possibles, nous notons :

$$\text{Source} \rightarrow \text{Résultat-1} \mid \text{Résultat-2}$$

au lieu de

$$\text{Source} \rightarrow \text{Résultat-1}$$
$$\text{Source} \rightarrow \text{Résultat-2}$$

On parle de forme de Backus-Naur.

À propos des grammaires
Description informelle (suite)

Une grammaire définit donc des règles de production sous la forme :

$$\begin{aligned}\text{Source-1} &\rightarrow \text{Résultat1-1} \mid \text{Résultat2-1} \mid \dots\\ \text{Source-2} &\rightarrow \text{Résultat1-2} \mid \text{Résultat2-2} \mid \dots\\ &\dots\\ \text{Source-n} &\rightarrow \text{Résultat1-n} \mid \text{Résultat2-n} \mid \dots\end{aligned}$$

où les résultats peuvent "contenir" des sources.

Une grammaire a une source unique au départ pour générer des mots (qui peuvent générer plusieurs sources), appelée *axiome*.

Le processus partant d'une source jusqu'à un résultat est appelé *dérivation*.

Le processus partant de l'axiome jusqu'à un résultat qui ne peut produire de nouvelles sources (cad un mot) est appelé *dérivation complète*.

À propos des grammaires
Un exemple informel

Exemple (Grammaire pour les phrases verbales en français)

- Des phrases verbales en français peuvent être générées par les règles de production suivantes en considérant Phrase comme l'axiome :

Phrase	$\rightarrow$	Sujet Verbe Complément
Sujet	$\rightarrow$	je | tu | ... | le chien | le chat | ...
Verbe	$\rightarrow$	suis | es | ... | ai | as | ...
Complément	$\rightarrow$	grand | petit | joli | gris | froid | chaud | ...

- Exemple de dérivation permettant de générer une phrase :

$$\begin{aligned} \text{Phrase} &\Rightarrow \text{Sujet Verbe Complément} \Rightarrow \text{je Verbe Complément} \\ &\Rightarrow \text{je suis Complément} \Rightarrow \text{je suis petit} \end{aligned}$$

Ce que nous notons aussi : Phrase $\Rightarrow^*$ je suis petit

- Exemples de phrases générées :
  - je suis petit
  - tu es grand
  - le chien est joli
  - le chat a froid
- Certaines phrases ne peuvent pas être générées (ex : grand es tu).
- Certaines phrases peuvent être générées mais n'ont pas de sens : je as grand.

Remarque Attention, une phrase verbale est un mot généré par la grammaire. □

À propos des grammaires
Utilisation des grammaires

Exemples d'utilisation des grammaires

- Description de la syntaxe (règles grammaticales) d'un langage de programmation. Par exemple :

Commande $\rightarrow$	x **:=** expression_arithmetique
	| **if** condition_bool **then** Commande **else** Commande **fi**.
	| Commande **;** Commande | ...
expr_arith $\rightarrow$	entier | expr_arith **+** expr_arith | expr_arith - expr_arith | ...
cond_bool $\rightarrow$	**true** | **false** | cond_bool & cond_bool | **not** cond_bool | ...

- Génération d'un *analyseur syntaxique* qui permet de déterminer si un programme passé en entrée est syntaxiquement correct (par rapport à la grammaire du langage). Dans un compilateur, les analyseurs lexical et syntaxique sont utilisés avec un analyseur sémantique qui vérifie le sens.
- Génie logiciel : modélisation de problème sous forme de grammaires, programmation adaptative,
- Représentation de données (syntaxe des documents HTML et XML).
- Calcul évolutif (progr. génétique),
- Représentation de réseaux de neurones,
- Compression de données.

Plan Chap. 5 - Grammaires et grammaires régulières

Grammaire
Définition

Définition (Grammaire)

Une grammaire est un quadruplet (V_T, V_N, Z, P) tel que :

- V_T est un ensemble fini qui dénote l'ensemble des symboles terminaux,
- V_N est un ensemble fini qui dénote l'ensemble des symboles non-terminaux,
- $Z \in V_N$ l'axiome et
- $P \subseteq (V_T \cup V_N)^* \times (V_T \cup V_N)^*$ est l'ensemble des règles de production.

tels que V_T et V_N sont disjoints ($V_T \cap V_N = \emptyset$),

$V = V_T \cup V_N$ est le vocabulaire de la grammaire.

On note $x \rightarrow y$ au lieu de $(x, y) \in P$.

Exemple (Grammaire)

- $G_1 = (\{a, b\}, \{S\}, S, P)$ avec $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$.
- $G_2 = (\{a, b\}, \{S, A, B\}, S, P)$ avec $P = \{S \rightarrow ASB \mid \epsilon, A \rightarrow aA \mid a, B \rightarrow Bb \mid b\}$.

Dérivations en une et plusieurs étapes

Soient $G = (V_T, V_N, Z, P)$ une grammaire et $V = V_T \cup V_N$ son vocabulaire.

Définition (Relation de dérivation en une étape)

G permet de dériver un mot $u \in V^*$ en un mot $v \in V^*$ en une étape, noté $u \Rightarrow v$, s'il existe un règle $x \to y \in P$ telle que :

- $u = \alpha \cdot x \cdot \beta$,
- $v = \alpha \cdot y \cdot \beta$,

avec $\alpha, \beta \in V^*$. Les mots u et v sont appelés *formes sententielles*.

Exemple (Dérivations en une étape pour la grammaire G_2)

- $S \Rightarrow \epsilon$
- $S \Rightarrow ASB$
- $ASB \Rightarrow AB$
- $aAASBBbbb \Rightarrow aaASBbbbb$

Définition (Relation de dérivation en plusieurs étapes)

La relation de dérivation en plusieurs étapes, notée $\Rightarrow^*$, est la *fermeture transitive* de $\Rightarrow$.

Exemple (Dérivation en plusieurs étapes)

- $S \Rightarrow^* \epsilon$
- $S \Rightarrow^* ab$
- $S \Rightarrow^* aaASBbbbb$
- $S \Rightarrow^* aabb$
- $aaASBbbbb \Rightarrow^* aaabbbbb$

Langage généré par une grammaire

Soient $G = (V_T, V_N, Z, P)$ une grammaire et $V = V_T \cup V_N$ son vocabulaire.

Définition (Langage engendré par une grammaire)

$$L(G) = \{w \in V_T{}^* \mid Z \Rightarrow^* w\}$$

Exemple (Grammaire)

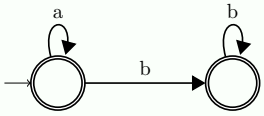
- Le langage généré par G_1 est $\left\{a^i b^i \mid i \in \mathbb{N}\right\}$.
- Le langage généré par G_2 est $\left\{a^i b^j \mid i, j \in \mathbb{N}\right\}$, aussi décrit par l'expression régulière $a^* b^*$.
- Le langage généré par $(\{a, b\}, \{S, A, B\}, S, P)$ avec $P = \{S \to AB, A \to aA \mid a, B \to Bb \mid b\}$ est $\left\{a^i b^j \mid i, j \in \mathbb{N} \setminus \{0\}\right\}$, aussi décrit par l'expression régulière $a^+ b^+$.

Grammaire
Lien avec les automates

L'ensemble des non-terminaux dans les grammaires « correspond » à l'alphabet des automates :

- Un automate reconnait des mots sur son alphabet.
- Une grammaire génère des mots sur son ensemble de symboles terminaux.

- La grammaire $G_2 = (\{a, b\}, \{S, A, B\}, S, P)$ avec $P = \{S \to ASB \mid \epsilon, A \to aA \mid \epsilon, B \to Bb \mid \epsilon\}$ génère le langage $\left\{a^i b^j \mid i, j \in \mathbb{N}\right\}$, cad des mots sur l'ensemble de non-terminaux $\{a, b\}$.
- L'automate



reconnaît les mêmes mots sur $\{a, b\}$.

Classification de Chomsky des grammaires

Les contraintes sur la forme de la grammaire définissent son *type*.

Type 0, Grammaires *générales* :
$x \to y$, avec $x \in V^* \cdot V_N \cdot V^*, y \in V^*$

Type 1, Grammaires *contextuelles* :
$\alpha \cdot A \cdot \beta \to \alpha \cdot \gamma \cdot \beta$, avec $\alpha, \beta, \gamma \in V^*, \gamma \neq \epsilon, A \in V_N$

Type 2, Grammaires *non-contextuelles* : (langages hors-contextes, algébriques) $A \to \gamma$, avec $A \in V_N, \gamma \in V^*$

Type 3, Grammaires *régulières* : (langages rationnels)

- linéaires gauches, $A \to B \cdot a \mid A \to a \mid A \to \epsilon$
- linéaires droites, $A \to a \cdot B \mid A \to a \mid A \to \epsilon$

avec $A, B \in V_N, a \in V_T$

Cas des langages de programmation

- Les grammaires régulières sont utilisées pour le lexique (analyse lexicale).
- Les grammaires non-contextuelles pour décrire les (syntaxiquement) programmes corrects.

Exemples de grammaires non-contextuelles

Les grammaires suivantes sont décrites par leurs règles de production.

Exemple (Grammaire d'expressions arithmétiques)

$$\begin{array}{lcl} E & \to & E + T \\ E & \to & T \end{array} \qquad \begin{array}{lcl} T & \to & T * F \\ T & \to & F \end{array} \qquad \begin{array}{lcl} F & \to & \texttt{IDF} \\ F & \to & \texttt{NUM} \\ F & \to & (E) \end{array}$$

aussi représenté

$$\begin{array}{lcl} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & \texttt{IDF} \mid \texttt{NUM} \mid (E) \end{array}$$

Exemple (Le langage $\{a^n \cdot b^n \mid n \in \mathbb{N}\}$)

$$\begin{array}{lcl} S & \to & \epsilon \\ S & \to & aSb \end{array}$$

Grammaires linéaires à droite

Les grammaires linéaires à *droite* sont t.q. les règles de production sont de la forme

- $A \to xB$, ou
- $A \to x$,

avec $x \in {V_T}^*$.

Exemple (Grammaire linéaire à droite)

- La grammaire

  $$(\{a, b\}, \{S\}, S, P) \text{ avec } P = \{S \to baS, S \to a\}$$

  génère le langage

  $$\{(b \cdot a)^n \cdot a \in \{a, b\}^* \mid n \in \mathbb{N}\}.$$

- La grammaire

  $$(\{a, b\}, \{S, A, B\}, S, P) \text{ avec } P = \{S \to A \mid B, A \to aA \mid \epsilon, B \to bB \mid \epsilon\}$$

  génère le langage

  $$\{a^n \in \{a, b\}^* \mid n \in \mathbb{N}\} \cup \{b^n \in \{a, b\}^* \mid n \in \mathbb{N}\}.$$

Grammaires linéaires à gauche

Les grammaires linéaires à *gauche* sont t.q. les règles de production sont de la forme

- $A \to Bx$, ou
- $A \to x$,

avec $x \in {V_T}^*$.

Exemple (Grammaire linéaire à gauche)

- La grammaire

  $$(\{a, b\}, \{S\}, S, P) \text{ avec } P = \{S \to Sba, S \to a\}$$

  génère le langage

  $$\{a \cdot (b \cdot a)^n \in \{a, b\}^* \mid n \in \mathbb{N}\}.$$

- La grammaire

  $$(\{a, b\}, \{S, A, B\}, S, P) \text{ avec } P = \{S \to Aab, A \to Aab \mid B, B \to a\}$$

  génère le langage

  $$\{a \cdot (a \cdot b)^n \in \{a, b\}^* \mid n \in \mathbb{N} \setminus \{0\}\}.$$

Grammaire linéaire

On peut ajouter à la hiérarchie de Chomsky le type des grammaires linéaires entre les grammaires de types 2 et 3.
Les grammaires linéaires sont telles que chaque règle de production a au plus un symbole non-terminal dans sa partie droite (il n'y a pas de contrainte sur la position de ce symbole non-terminal).

Exemple (Grammaire linéaire)

- La grammaire $(\{a, b\}, \{S\}, S, P)$ avec $P = \{S \to bSa, S \to \epsilon\}$ génère le langage

  $$\{b^n a^n \in \{a, b\}^* \mid n \in \mathbb{N}\}.$$

- $(\{a, b\}, \{S\}, S, P)$ avec $P = \{S \to A, A \to aB \mid \epsilon, B \to Ab\}$ génère le langage

  $$\{a^n b^n \in \{a, b\}^* \mid n \in \mathbb{N}\}.$$

Exemple (Grammaire non linéaire)

- La grammaire $(\{a, b\}, \{S, A, B\}, S, P)$ avec $P = \{S \to SS \mid \epsilon \mid aSb \mid bSa\}$ génère le langage

  $$\{w \in \{a, b\}^* \mid |w|_a = |w|_b\}.$$

Plan Chap. 5 - Grammaires et grammaires régulières

Grammaires régulières et automates à états finis

Équivalence entre grammaires régulières et AEFDs

1. À chaque grammaire régulière, on peut associer un ϵ-AEFND qui reconnaît le même langage (correspondance entre vocabulaire terminal et état)
2. Réciproquement, à chaque AEFD, on peut associer une grammaire régulière qui reconnaît le même langage.

Nous démontrons ces deux points en donnant des procédures de traduction entre grammaires et automates.

Corollaire

Les grammaires régulières génèrent les langages à états (et les langages réguliers par le théorème de Kleene).

Corollaire : retour sur la fermeture des langages réguliers

Étant données deux grammaires G et G' générant des langages L et L', respectivement, nous pouvons trouver des grammaires générant les langages

- $L \cup L'$,
- $L \cap L'$,
- L^*
- $\overline{L}$,
- $L \cdot L'$

à partir des grammaires G et G'. Nous trouverons ces grammaires en TD.

Plan Chap. 5 - Grammaires et grammaires régulières

Grammaire linéaire à droite vers automate

Intuition

Intuitivement, nous construisons un ϵ-AEFND selon les étapes suivantes :

1. Création des états
   - un état par symbole non-terminal
   - l'axiome est l'état initial
   - un état final F qui n'est pas associé à un axiome.
2. Création des transitions associées aux règles de production :
   - Pour chaque règle $A \to xB$, nous créons une transition entre l'état A et l'état B étiquetée par x
   - Pour chaque règle $A \to x$, nous créons une transition entre l'état A et l'état F étiquetée par x.

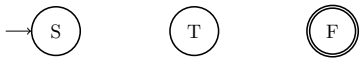
L'automate obtenu est tel qu'il peut y avoir des mots (formés par la concaténation de plusieurs terminaux) sur les transitions.

3. Nous créons ainsi des états et transitions permettant d'avoir uniquement un non-terminal étiquetant chaque transition.
   (Pour chaque transition $A \xrightarrow{x} X$ avec $x = x_0 \cdots x_n \in V_T^+$, $n \in \mathbb{N} \setminus \{0\}$ et X un état qui est soit associé à un non-terminal soit l'état F, nous créons les états intermédiaires $Ax_{\dots 0}X, Ax_{\dots 1}X, \dots, Ax_{\dots n-1}X$ (le nom assure l'unicité) et les transitions $A \xrightarrow{x_0} Ax_{\dots 0}X, Ax_{\dots 0}X \xrightarrow{x_1} Ax_{\dots 1}X, \dots, Ax_{\dots n-1}X \xrightarrow{x_n} X$.)

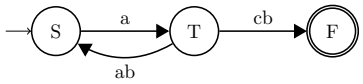
Grammaire linéaire à droite vers automate
Intuition sur un exemple

Considérons la grammaire $(\{a, b\}, \{S, T\}, S, \{S \to aT, T \to abS \mid cb\})$.

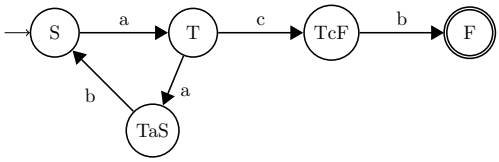
1. Nous créons les états, déterminons l'état initial et ajoutons l'état final :



2. Nous créons les transitions associées à chaque type de règle de production :



3. Nous créons les états et transitions intermédiaires pour avoir uniquement un symbole étiquetant chaque transition :



Grammaire linéaire à droite vers automate
La transformation - première version

Soit $G = (V_T, V_N, Z, P)$ une grammaire régulière à droite.

Définition (Automate associé à une grammaire régulière droite)

L'"ϵ-AEFND" associé à G est $A_G = (V_N \cup \{F\}, Z, V_T, \Delta, \{F\})$ avec

$$\begin{aligned}\Delta = & \quad \{(N, x, N') \mid (N, x \cdot N') \in P\} \\ \cup & \quad \{(N, x, F) \mid (N, x) \in P\} \\ \cup & \quad \{(N, \epsilon, F) \mid (N, \epsilon) \in P\}\end{aligned}$$

Ce qui est généré n'est pas stricto-sensu un ϵ-AEFND car Δ peut contenir des transitions étiquetées avec des mots de longueur strictement plus grande que 1.

Remarque Nous pouvons aussi transformer la grammaire (et obtenir une grammaire équivalente) afin d'assurer que toute les règles sont de la forme $A \to x \cdot B$ ou $A \to x$ avec $|x| \leq 1$ en transformant chaque règle

$$A \to x_0 \cdots x_n B$$

en $A \to x_0 \cdot A\_x_0\_B, A\_x_0\_B \to x_1 \cdot A\_x_{x_0 \cdot x_1}, \ldots, A\_x_{\ldots n-1}\_B \to x_n \cdot B$.
Ainsi, nous obtenons directement un ϵ-AEFND en appliquant la transformation ci-dessus à la grammaire transformée. □

Grammaire linéaire à droite vers automate
La transformation - seconde version

Soit $G = (V_T, V_N, Z, P)$ une grammaire régulière à droite.

Définition (Automate associé à une grammaire régulière droite)

L'ϵ-AEFND associé à G est $A_G = (V_N \cup \{F\} \cup \text{etats\_inter}(\Delta'), Z, V_T, \Delta', \{F\})$ avec

$$\begin{aligned}\Delta = & \quad \{(N, x, N') \mid (N, x \cdot N') \in P\} \\ \cup & \quad \{(N, x, F) \mid (N, x) \in P\} \\ \cup & \quad \{(N, \epsilon, F) \mid (N, \epsilon) \in P\}\end{aligned}$$

$$\begin{aligned}\Delta' = & \quad \Delta \\ & \setminus \{(N, x, N') \in \Delta \mid |x| > 1\} \\ & \cup \{(N, x_0, Nx_0N'), (Nx_{\ldots n-1}N', x_n, N'), (N_{x\ldots i-1}N', x_i, Nx_{\ldots i}N') \\ & \qquad \mid (N, x, N') \in \Delta \text{ et } x = x_0 \cdots x_n \text{ et } |x| > 1 \text{ et } i \in [1, n-1]\}\end{aligned}$$

$$\text{etats\_inter}(\Delta') = \left\{N, N' \mid \exists x \in V_T : (N, x, N') \in \Delta'\right\}$$

Correction de la synthèse d'automate depuis des grammaires

Étant donnée une grammaire, le langage reconnu par l'automate obtenu à partir de cette grammaire est le langage généré par la grammaire.

Plan Chap. 5 - Grammaires et grammaires régulières

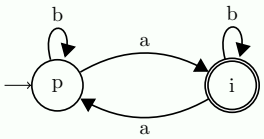
Automate vers grammaire
Intuition

Idée :

- L'ensemble des symboles non-terminaux est l'alphabet de l'automate.
- Associer un non-terminal par état.
- L'axiome est l'état initial.
- Les états accepteurs sont associés à des règles de production où la partie droite est ϵ.
- Pour chaque transition de q vers q' sur un symbole a, alors on a une règle de production $q \to a \cdot q'$.

Exemple (Automate vers grammaire)

Considérons l'automate



Nous lui associons la grammaire $(\{P, I\}, \{a, b\}, P, Prod)$, avec :

$$Prod = \{P \to b \cdot P \mid a \cdot I, I \to b \cdot I \mid a \cdot P \mid \epsilon\}.$$

Automate vers grammaire

Soit $A = (Q, q_0, \Sigma, \delta, F)$ un AEFD.

Définition (Grammaire régulière associée à un automate)

La grammaire associée à l'automate est (Q, Σ, q_0, P) avec

$$P = \left\{(q, a \cdot q') \mid (q, a, q') \in \delta\right\} \cup \{(f, \epsilon) \mid f \in F\}$$

Correction de la synthèse d'automate depuis des grammaires

Étant donné un automate, le langage généré par la grammaire obtenu à partir de l'automate est le langage reconnu par l'automate.

Remarque Cette transformation s'adapte aisément pour générer des grammaires à partir d'AEFNDs et ϵ-AEFNDs. □

Plan Chap. 5 - Grammaires et grammaires régulières

Traduction d'expression régulière en grammaire
Rappel sur les expressions régulières

La syntaxe des expression régulières sur un alphabet Σ est définie inductivement par :

- ϵ et $\emptyset$ sont des expressions régulières.
- Si $a \in \Sigma$ alors a est une expression régulière.
- Si e et e' sont des expressions régulières, alors $e + e'$ est une expression régulière.
- Si e et e' sont des expressions régulières, alors $e \cdot e'$ est une expression régulière.
- Si e est une expression régulière, alors e^* est une expression régulière.

Traduction d'expression régulière en grammaire
Rappel sur les expressions régulières

Définition (Traduction compositionnelle des expressions régulières en grammaire linéaire à droite)

Expressions régulières de base :

- Pour ϵ : $(\{S\}, \emptyset, S, \{S \to \epsilon\})$
- Pour $\emptyset$: $(\{S\}, \emptyset, S, \emptyset)$
- Pour $a \in \Sigma$: $(\{S\}, \{a\}, S, \{P \to a\})$

Expressions régulières composées : soient $G_E = (N_E, T_E, S_E, P_E)$ et $G_{E'} = (N_{E'}, T_{E'}, S_{E'}, P_{E'})$ des grammaires linéaires à droite pour des expressions régulières e et e'.
Pour simplifier, nous supposons que $N_E \cap N_{E'} = \emptyset$.

- Pour $e + e'$: $(N_E \cup N_{E'}, T_E \cup T_{E'}, S, P_E \cup P_{E'} \cup \{S \to S_E \mid S_{E'}\})$
- Pour $e \cdot e'$: $(N_E \cup N_{E'}, T_E \cup T_{E'}, S, P'_E \cup P_{E'} \cup \{S \to S_E \mid S_{E'}\})$ avec $P'_E = \{A \to x \cdot S_{E'} \mid A \to x \in P_E\} \cup \{A \to x \cdot B \in P_E\}$.
- Pour e^* : $(N_E, T_E, S_E, P_E \cup \{A \to x \cdot S_E \mid A \to x \in P_E\} \cup \{S_E \to \epsilon\})$

Traduction d'expression régulière en grammaire
Rappel sur les expressions régulières

Exemple (Traduction compositionnelle des expressions régulières en grammaires linéaires à droite)

Pour $(a + b)^*$.

- Grammaire pour a : $(\{S_a\}, \{a\}, S_a, \{S_a \to a\})$
- Grammaire pour b : $(\{S_b\}, \{b\}, S_b, \{S_b \to b\})$
- Grammaire pour $a + b$: $(\{S, S_a, S_b\}, \{a, b\}, S, \{S \to S_a \mid S_b, S_a \to a, S_b \to b\})$
- Grammaire pour $(a + b)^*$:
  $(\{S, S_a, S_b\}, \{a, b\}, S, \{S \to S_a \mid S_b \mid \epsilon, S_a \to a \cdot S, S_b \to b \cdot S\})$

Plan Chap. 5 - Grammaires et grammaires régulières

Résumé

Les grammaires permettent de décrire les langages de manière générative.
(Les automates permettent de décrire les langages (à états) de manière *opérationnelle*.
Les expressions régulières permettent de décrire les langages (réguliers) de manière *déclarative*.)

Grammaire et grammaires régulières

- Grammaires : définition, dérivation et langage généré.
- Types de grammaires et classification de Chomsky.
- Équivalence entre grammaires régulières et AEFDs :
  - une grammaire régulière est linéaire à droite ou à gauche, on parle de grammaire régulière à droite ou à gauche,
  - traduction d'une grammaire régulière à droite vers un ϵ-AEFND,
  - traduction d'une grammaire régulière à gauche vers un ϵ-AEFND,
  - traduction d'un ϵ-AEFND/AEFND/AEFD vers une grammaire linéaire à droite.