Théorie des Langages

Cours 6 : Grammaires

Lionel Rieg

Grenoble INP - Ensimag, 1re année

Reconnaissance et génération

- Reconnaissance : on part de w et on veut savoir si $w \in L$
  - automate fini
  - automate à pile
  - machine de Turing

  $\Rightarrow$ peu de structure pour comprendre le langage

- Génération : on veut engendrer un mot d'un langage
  - expressions régulières
  - grammaires

  $\Rightarrow$ règles structurées qui décrivent le langage

  Permet de mieux comprendre comment est fait un langage

Les grammaires sont utilisées pour décrire les langages de programmation.

Extrait de la grammaire de Python 3

```
statement: compound_stmt  | simple_stmts
simple_stmts:
    | ';'.simple_stmt+ [';'] NEWLINE
simple_stmt:
    | assignment
    | return_stmt
    | import_stmt
    | raise_stmt
    | 'pass'
    | assert_stmt
    | 'break'
    | 'continue'
compound_stmt:
    | function_def
    | if_stmt
    | for_stmt
    | try_stmt
    | while_stmt
```

Intérêt des grammaires

Un formalisme de description de langage

- permet de définir/caractériser des classes de langages
- algorithmes génériques sur chaque classe
- étudier des propriétés pour chaque classe

Hiérarchie de Chomsky = compromis expressivité/décidabilité

- Expressivité = quels genres de langages peut-on exprimer ?
- Décidabilité = existe-il un algorithme qui répond au problème ?

Langage de programmation idéal :

- expressif : sa syntaxe permet beaucoup de chose
- décidable : on doit pouvoir analyser sa syntaxe automatiquement

⇒ Compromis : trouver une classe décidable mais suffisamment expressive

Exemple du français

Exemple (Grammaire du français)

Une phrase est formée d'un sujet, d'un verbe et, optionnellement, d'un complément.
Un sujet peut être un nom propre ou un nom commun, optionnellement précédé ou suivi par un adjectif.
Un complément est un COD, un COI ou les deux.
Un nom commun peut être « voiture », « maison » ou etc. (tout le vocabulaire).
...

Une grammaire de langue naturelle :

- Explique comment construire des catégories grammaticales ... (phrase, sujet, complement, etc.) vocabulaire non terminal
- ... en utilisant des règles.
- On applique des règles jusqu'à n'avoir plus que des mots (**voiture**, **maison**, **est**, etc.). vocabulaire terminal

Définition d'une grammaire

Définition (grammaire)

Une grammaire est un quadruplet $G \stackrel{\text{def}}{=} \langle V_T, V_N, \mathsf{S}, R \rangle$ où

- V_T est un ensemble fini appelé le vocabulaire terminal
- V_N est un ensemble fini appelé le vocabulaire non terminal
- $\mathsf{S} \in V_N$ est l'axiome
- R est un ensemble de règles de la forme $u \rightarrow v$, avec
  - $u \in V^+$
  - $v \in V^*$

On pose $V \stackrel{\text{def}}{=} V_N \cup V_T$ et on demande $V_N \cap V_T = \emptyset$.

Comment s'utilise une grammaire ?

- on commence avec l'axiome S (Start symbol)
- on applique des règles tant qu'on veut
- on peut s'arrêter si le mot ne contient que des symboles terminaux (mais on peut aussi continuer)

Notations

Conventions

- On distingue graphiquement V_T et V_N (**gras**, souligné, *italique*, etc.)
- V est l'ensemble des symboles présents dans G et $V_N = V \setminus V_T$.
- L'axiome S est la partie gauche de la première règle.
- Si plusieurs règles $u \to w_1, \ldots, u \to w_n$,
  on peut les noter $u \to w_1 \mid \ldots \mid w_n$.
- Il arrive qu'on ne précise pas V_T, s'il se déduit du langage.

Notations

- $\to$ pour une règle, $\Longrightarrow$ pour son application dans un mot
- Pour mieux identifier l'utilisation d'une règle dans un mot,
  on souligne parfois sa partie gauche : $ab\underline{S}c \Longrightarrow abcac$

Exemple

Exemple (Grammaire pour $(ab)^*(ba)^+$)

$V_T = \{a, b\}$ $\quad V_N = \{X, \mathsf{AB}, \mathsf{BA}\}$ $\quad \mathsf{S} = X$

$$R = \left\{ X \underset{1}{\to} \mathsf{AB}\,\mathsf{BA}, \quad \mathsf{AB} \underset{2}{\to} ab\mathsf{AB}, \quad \mathsf{AB} \underset{3}{\to} \varepsilon, \quad \mathsf{BA} \underset{4}{\to} ba\mathsf{BA}, \quad \mathsf{BA} \underset{5}{\to} ba \right\}$$

Quelques dérivations possibles :

- $\underline{X} \overset{1}{\Longrightarrow} \underline{\mathsf{AB}}\,\mathsf{BA} \overset{3}{\Longrightarrow} \varepsilon\,\underline{\mathsf{BA}} \overset{5}{\Longrightarrow} \varepsilon ba = ba$
- $\underline{X} \overset{1}{\Longrightarrow} \underline{\mathsf{AB}}\,\mathsf{BA} \overset{2}{\Longrightarrow} ab\,\underline{\mathsf{AB}}\,\mathsf{BA} \overset{2}{\Longrightarrow} abab\,\underline{\mathsf{AB}}\,\mathsf{BA}$
  $\overset{3}{\Longrightarrow} abab\,\underline{\mathsf{BA}} \overset{4}{\Longrightarrow} ababba\,\underline{\mathsf{BA}} \overset{5}{\Longrightarrow} abababa$
- $\underline{X} \overset{1}{\Longrightarrow} \mathsf{AB}\,\underline{\mathsf{BA}} \overset{4}{\Longrightarrow} \underline{\mathsf{AB}}\,ba\,\mathsf{BA} \overset{2}{\Longrightarrow} ab\,\underline{\mathsf{AB}}\,ba\,\mathsf{BA}$
  $\overset{2}{\Longrightarrow} abab\,\mathsf{AB}\,ba\,\underline{\mathsf{BA}} \overset{5}{\Longrightarrow} abab\,\underline{\mathsf{AB}}\,baba \overset{3}{\Longrightarrow} abababa$

Premiers exercices

Exercice

Donner des grammaires pour les langages suivants :

- $\{a^n b^n \mid n \in \mathbb{N}\}$
- $\{a^n b^n c^n \mid n \geqslant 1\}$

- $\{a^n b^n \mid n \in \mathbb{N}\}$:
  - $V_T \stackrel{\text{def}}{=} \{a, b\}$
  - $V_N \stackrel{\text{def}}{=} \{\mathsf{S}\}$
  - $\mathsf{S} \stackrel{\text{def}}{=} \mathsf{S}$
  - $R \stackrel{\text{def}}{=} \{\mathsf{S} \to a\mathsf{S}b, \quad \mathsf{S} \to \varepsilon\} = \{\mathsf{S} \to a\mathsf{S}b \mid \varepsilon\}$
- $\{a^n b^n c^n \mid n \geqslant 1\}$:
  - $V_T = \{a, b, c\}$
  - $\mathsf{S} \to a\mathsf{S}\mathsf{B}c \mid abc$
  - $b\mathsf{B} \to bb$
  - $c\mathsf{B} \to \mathsf{B}c$
  - donc par convention : $V_N = \{\mathsf{S}, \mathsf{B}\}$ et l'axiome est S

Relation de dérivation

Définition (Dérivation)

Soit $G = \langle V_T, V_N, \mathsf{S}, R \rangle$ une grammaire.
Étant donné $x, y \in V^*$, on dit que x dérive vers y, noté $x \Longrightarrow y$, lorsque :

- il existe une règle $u \to v$ dans R,
- x s'écrit $\alpha u \beta$ pour $\alpha, \beta \in V^*$
- y vaut $\alpha v \beta$.

Idée : on a remplacé u par v dans x pour obtenir y.

Pour être plus précis, on peut préciser la règle utilisée et sa position :

$$\alpha \underline{u} \beta \xRightarrow{u \to v} \alpha v \beta$$

Ne pas confondre $\to$ (règle) et $\Longrightarrow$ (dérivation).
Ne pas confondre $\Longrightarrow$ (dérivation) et $\Longrightarrow$ (implication logique).

Relation de dérivation itérée

Définition

- On note $\Longrightarrow^p$ les dérivations de longueur p, définies récursivement sur p par :
  - $u \Longrightarrow^0 v \stackrel{\text{def}}{=} u = v$
  - $u \Longrightarrow^{p+1} v \stackrel{\text{def}}{=} \exists w, u \Longrightarrow w \wedge w \Longrightarrow^p v$
- $\Longrightarrow^*$ est la fermeture réflexive transitive de $\Longrightarrow$: $\Longrightarrow^* \stackrel{\text{def}}{=} \bigcup_{p \in \mathbb{N}} \Longrightarrow^p$

Composition des dérivations :

Si $u_1 \Longrightarrow^{p_1} v_1$ et $u_2 \Longrightarrow^{p_2} v_2$, alors $u_1 u_2 \Longrightarrow^{p_1+p_2} v_1 v_2$

Ce n'est souvent pas la seule dérivation possible dans $u_1 u_2$!

Exercice : Donner une définition inductive de $u \Longrightarrow^* v$.

Base : $u = v$

Induction : $\exists w, u \Longrightarrow w \Longrightarrow^* v$

Langage engendré

Définition (Langage engendré)

Soit $G = \langle V, T, V_N, \mathsf{S}, R \rangle$ une grammaire.
Pour $u \in V^* = (V_T \cup V_N)^*$, on définit le langage engendré par u :

$$\mathcal{L}(u) \stackrel{\text{def}}{=} \{w \in V_T^* \mid u \Longrightarrow^* w\}$$

Le langage d'une grammaire est le langage engendré par son axiome :

$$\mathcal{L}(G) \stackrel{\text{def}}{=} \mathcal{L}(\mathsf{S}) = \{w \in V_T^* \mid \mathsf{S} \Longrightarrow^* w\}$$

Définition (Grammaires équivalentes)

Comme pour les automates, deux grammaires G_1 et G_2 sont dites équivalentes si elles engendrent le même langage : $\mathcal{L}(G_1) = \mathcal{L}(G_2)$.

Exemples de langage engendré

- Considérons la grammaire suivante : $\mathsf{S} \to \mathsf{AB}$
  $\mathsf{A} \to a \mid a\mathsf{A}$
  $\mathsf{B} \to b\mathsf{B} \mid \varepsilon$

  Que valent V_T, V_N, V, S ?
  $V_T = \{a, b\}$ $V_N = \{S, A, B\}$ $V = \{a, b, S, A, B\}$
  Quels sont les langages engendrés par A, B, S, AbA ?
  $\mathcal{L}(\mathsf{A}) = a^+$ $\mathcal{L}(\mathsf{B}) = b^*$ $\mathcal{L}(\mathsf{S}) = a^+b^*$ $\mathcal{L}(\mathsf{A}b\mathsf{A}) = a^+ba^+$

- Autre grammaire : $\mathsf{S} \to a\mathsf{SB}c \mid \varepsilon$
  $\mathsf{B}c \to b\mathsf{B}c \mid bc \mid b$
  $cb \to bc$
  $V_T = \{a, b, c\}$ $V_N = \{S, B\}$
  $\mathcal{L}(\mathsf{B}) = \emptyset$ $\mathcal{L}(b) = \{b\}$ $\mathcal{L}(cb) = \{cb, bc\}$
  $\mathcal{L}(\mathsf{B}c) = b^+(c + \varepsilon)$
  $\mathcal{L}(\mathsf{S}) = \{a^n w \mid n \in \mathbb{N}, |w|_c \leqslant n \leqslant |w|_b, \forall x$ préfixe de $w, |x|_b \geqslant |w|_c\}$

Énumération

Une grammaire permet d'énumérer des mots. On peut définir l'ensemble Gen_p des mots sur V engendrés après p dérivations par récurrence sur p :

- On commence avec le singleton contenant l'axiome : $\mathrm{Gen}_0 = \{\mathsf{S}\}$
- On applique toutes les règles possibles à toutes les positions possibles sur les mots de Gen_p : $\mathrm{Gen}_{p+1} = \{w \in V \mid \exists u \in \mathrm{Gen}_p, u \Longrightarrow w\}$

Exemple

S $\to$ AB A $\to$ a | aA B $\to$ bB | ε

- $\mathrm{Gen}_0 = \{\mathsf{S}\}$
- $\mathrm{Gen}_1 = \{\mathsf{AB}\}$
- $\mathrm{Gen}_2 = \{a\mathsf{B}, a\mathsf{AB}, \mathsf{A}b\mathsf{B}, \mathsf{A}\}$
- $\mathrm{Gen}_3 = \{ab\mathsf{B}, a, aa\mathsf{B}, aa\mathsf{AB}, a\mathsf{A}b\mathsf{B}, a\mathsf{A}, ab\mathsf{B}, a\mathsf{A}b\mathsf{B}, \mathsf{A}bb\mathsf{B}, \mathsf{A}b, a, a\mathsf{A}\}$

$\Rightarrow$ Procédure de semi-décision pour le problème $w \in \mathcal{L}(G)$?
peu efficace, semi-decision car on ne sait pas quand s'arrêter

Lien avec le théorème du point fixe : $\bigcup_{p \leqslant n} \mathrm{Gen}_p = f^{n+1}(\emptyset)$
avec $f : X \mapsto \{w \mid u \in X \wedge u \Longrightarrow w\} \cup \{S\}$

Hiérarchie de Chomsky

Classification des grammaires par restriction sur les règles

- type 3 : grammaire régulière — la plus décidable
- type 2 : grammaire hors-contexte (HC)
- type 1 : grammaire sous-contexte
- type 0 : grammaire générale — la plus expressive

Étude du compromis expressivité/décidabilité

Propriétés étudiés :

- $w \in \mathcal{L}(G)$?
- $\mathcal{L}(G) = \emptyset$?
- $|\mathcal{L}(G)| = \infty$?
- $\mathcal{L}(G_1) = \mathcal{L}(G_2)$?

Grammaire régulière (type 3)

- Restriction : règle de la forme

$$\mathsf{A} \to x\mathsf{B} \quad \text{ou} \quad \mathsf{A} \to \varepsilon, \quad \text{avec} \quad \mathsf{A}, \mathsf{B} \in V_N, x \in V_T$$

  Grammaire linéaire à droite
  - linéaire : un seul non-terminal à droite de la règle
  - à droite : ce non-terminal est en dernière position

    (à gauche = en première position)

- Reconnaisseurs : automates finis
- Idée : non-terminal = état de l'automate (sans ε-transition)
- Exemple : $\mathsf{A} \to a\mathsf{A} \mid b\mathsf{B} \qquad \mathsf{B} \to b\mathsf{B} \mid \varepsilon$

  $\mathcal{L}(\mathsf{A}) = a^* b^+$
- Caractérisation équivalente :

$$\mathsf{A} \to w\mathsf{B} \quad \text{ou} \quad \mathsf{A} \to w, \quad \text{avec} \quad \mathsf{A}, \mathsf{B} \in V_N, w \in V_T^*$$

Conversion grammaire régulière / automate fini

L'automate ne doit avoir qu'un unique état initial
et pas d'ε-transition (mais on sait faire).

Automate fini		**Grammaire régulière**
Alphabet V		Symboles terminaux V_T
États Q		Symboles non-terminaux V_N
Transition $(q, x, q') \in \delta$	$\Longleftrightarrow$	Règle $q \rightarrow xq'$
État initial i		Axiome S
États acceptants $q \in F$		Règle $q \rightarrow \varepsilon$

Invariant de correction : (preuve par induction)

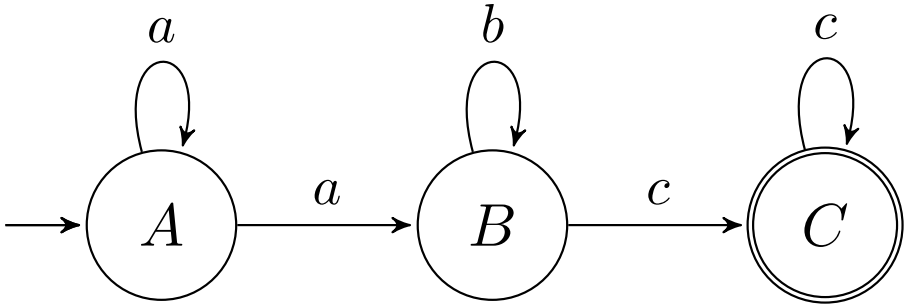
χ chemin de q à q' de trace w $\Longleftrightarrow$ $q \Longrightarrow^* wq'$

Exercice : faire les conversions suivantes

- Automate fini vers grammaire régulière : $a^+b^*c^+$
- Grammaire régulière vers automate fini : $\mathsf{P} \rightarrow a\mathsf{I} \mid b\mathsf{P}$
  $\mathsf{I} \rightarrow a\mathsf{P} \mid b\mathsf{I} \mid \varepsilon$

Correction de l'exercice

- Automate fini vers grammaire régulière : $a^+b^*c^+$



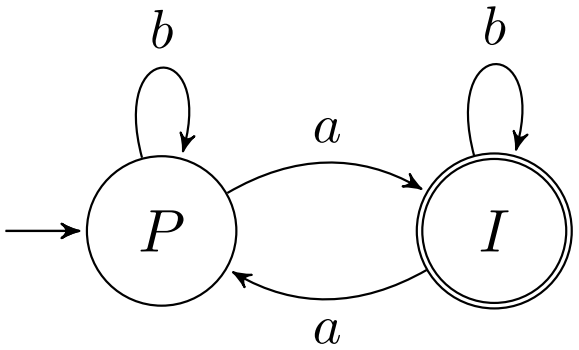
$\rightsquigarrow$

$$\begin{array}{lcl} \mathsf{A} & \to & a\mathsf{A} \mid a\mathsf{B} \\ \mathsf{B} & \to & b\mathsf{B} \mid c\mathsf{C} \\ \mathsf{C} & \to & c\mathsf{C} \mid \varepsilon \end{array}$$

- Grammaire régulière vers automate fini :

$$\begin{array}{lcl} \mathsf{P} & \to & a\mathsf{I} \mid b\mathsf{P} \\ \mathsf{I} & \to & a\mathsf{P} \mid b\mathsf{I} \mid \varepsilon \end{array}$$

$\rightsquigarrow$



Grammaire hors-contexte/non-contextuelle (type 2)

- Restriction : règle de la forme

$$\mathsf{A} \rightarrow w \qquad \text{avec} \quad \mathsf{A} \in V_N, w \in V^*$$

- Idée : pas de contexte autour du non-terminal à transformer
- Exemple : $\mathsf{S} \rightarrow a\mathsf{S}b \mid \varepsilon$
  $\mathcal{L}(\mathsf{S}) = \{a^n b^n \mid n \in \mathbb{N}\}$
- Reconnaisseurs : automates à pile

Bon compromis expressivité/décidabilité
$\Rightarrow$ algorithmes efficaces de reconnaissance
$\Rightarrow$ utilisées pour les langages de programmation

La classe qu'on étudiera le plus.

Grammaire sous-contexte/contextuelle (type 1)

- Restriction : règle de la forme

$$\alpha \mathsf{A} \beta \ \to \ \alpha w \beta \qquad \text{avec} \quad \mathsf{A} \in V_N, \quad \alpha, \beta, w \in V^* \quad \text{et } w \neq \varepsilon$$

- Idée : non-terminal à transformer, avec un contexte
- Caractérisation équivalente (plus utile) :

$$u \ \to \ v \qquad \text{avec} \quad u, v \in V^+ \text{ et } |u| \leqslant |v|$$

- Exemple : $\mathsf{S} \ \to \ a\mathsf{S}\mathsf{B}c \mid abc \qquad c\mathsf{B} \ \to \ \mathsf{B}c \qquad a\mathsf{B} \ \to \ ab$
  $\mathcal{L}(\mathsf{S}) = \{a^n b^n c^n \mid n \geqslant 1\}$

- Reconnaisseurs : machines de Turing linéairement bornée

Propriétés des grammaires sous-contexte

- La caractérisation $|u| \leqslant |v|$ implique que la taille d'un mot ne peut pas décroître au cours d'une dérivation.
  $\Rightarrow w \in \mathcal{L}(\mathsf{S})$ devient décidable : on peut s'arrêter lorsque la taille des mots engendrés dépasse celle du mot à tester.
  1. Commencer avec $\mathrm{Gen}_0 = \{\mathsf{S}\}$
  2. Calculer $D_p = \{x \in \mathrm{Gen}_p \mid |x| \leqslant |w|\}$
  3. Jusqu'à trouver w ou qu'il soit vide.

- Lorsque ε doit être dans le langage d'une grammaire sous-contexte, on ajoute :
  - $Z \in V_N$, nouvel axiome
  - $Z \to \mathsf{S} \mid \varepsilon$ avec S l'ancien axiome

Grammaire générale (type 0)

- Restriction : aucune
- Reconnaisseurs : machine de Turing

Les plus expressives, mais peu de chose décidables

$\Rightarrow$ même $w \in \mathcal{L}(G)$ n'est plus décidable

Relations entre classes

Inclusions :

- régulière $\subset$ hors-contexte
- hors-contexte sans ε-règle $\subset$ sous-contexte (ε-règle $= u \to \varepsilon$)
- sous-contexte $\subset$ générale

Classes de langages

Un langage est dit régulier/hors-contexte/sous-contexte s'il peut être généré par une grammaire de même type.

Pas d'algorithme pour décider de la classe d'un langage

Décidabilité de propriétés :

Propriétés \ type de G	3 (reg)	2 (HC)	1 (SC)	0 (gen)
$w \in \mathcal{L}(G)$?	✓	✓	✓	✗
$\mathcal{L}(G) = \emptyset$?	✓	✓	✗	✗
$\lvert\mathcal{L}(G)\rvert = \infty$?	✓	✓	✗	✗
$\mathcal{L}(G_1) = \mathcal{L}(G_2)$?	✓	✗	✗	✗