Théorie des Langages
Épisode 1 — Langages et grammaires

Thomas Pietrzak

Université Paul Verlaine — Metz

Alphabet

Un alphabet Σ est un ensemble fini. Les éléments d'un alphabet Σ sont appelés des caractères (ou des lettres, ou des digits).

Mot

Un mot (ou une chaine) m sur Σ est une concaténation de caractères de Σ.

Mot vide

Le mot vide est noté ϵ

Longueur

La longueur d'un mot m, noté $|m|$, est le nombre de caractères du mot m. On notera que ϵ est l'unique mot de longueur 0.

On note :

- Σ^* l'ensemble de tous les mots sur l'alphabet Σ
- Σ^+ l'ensemble de tous les mots sur l'alphabet Σ sans ϵ

Langage

On appelle L un langage formel (ou plus courament langage) sur l'alphabet Σ si $L \subseteq \Sigma^*$. $\emptyset$ est le langage vide.

Soient L, L_1 et L_2 des langages sur l'alphabet Σ.

- Union de L_1 et L_2 : $L_1 \cup L_2 = \{m \mid m \in L_1 \text{ ou } m \in L_2\}$
- Concaténation de L_1 et L_2 : $L_1 \cdot L_2 = \{m_1 m_2 \mid m \in L_1 \text{ et } m \in L_2\}$
- Étoile de L (fermeture de Kleene) :
  - $L^0 = \{\epsilon\}$
  - $L^1 = L$
  - $L^i = L \cdot L^{i-1}$ pour chaque entier $i \geq 2$
  - $L^* = \bigcup_{i=0}^{\infty} L^i$
  - on note $L^+ = L^* | L^0 = \bigcup_{i=1}^{\infty} L^i$

Propriétés

- $\{\epsilon\} \cdot L = L \cdot \{\epsilon\} = L$
- $\emptyset \cup L = L \cup \emptyset = L$
- $\emptyset \cdot L = L \cdot \emptyset = \emptyset$

Exemples

Soient $L = \{A, B, \ldots, Z, a, b, \ldots, z\}$ et $C = \{0, 1, \ldots, 9\}$. Considérons L et C comme des langages finis.

- $L \cup C$ est l'ensemble des lettres et des chiffres
- $L \cdot C$ est l'ensemble des chaînes formées d'une lettre suivie d'un chiffre
- L^4 est l'ensemble des chaînes de 4 lettres
- L^* est l'ensemble de toutes les chaînes de lettres, y compris ϵ
- $L(L \cup C)^*$ est l'ensemble de toutes les chaînes de lettres et de chiffres commençant par une lettre.

Si un langage L est fini on alors on peut décrire L en énumérant tous les mots de L.

Exemple

Soit L le langage de tous les mots-clés d'un langage de programmation comme Caml, C, C++, Pascal, Java, etc.
$L = \{let, rec, begin, end, function, match, try, with, for, \ldots\}$

Si un langage L est infini, comme par exemple le langage de tous les programmes (syntaxiquement corrects) que l'on peut écrire en C, il faut des moyens plus approfondis :

- Backus-Naur form (BNF pour les intimes)
- grammaires
- automates finis
- automates à pile
- machines de Turing
- ...

Un langage de programmation peut être défini par la description de :

- La syntaxe : ce à quoi ressemblent les programmes
- La sémantique : ce que les programmes signifient

Pour spécifier la syntaxe d'un langage de programmation on utilise souvent une notation appelée Backus-Nauf form (BNF).

Exemple

$$\langle \textit{mult operator} \rangle ::= * \mid /$$
$$\langle \textit{inst} \rangle ::= \textbf{if } (\langle \textit{expr} \rangle) \langle \textit{inst} \rangle \textbf{ else } \langle \textit{inst} \rangle$$
$$\langle \textit{term} \rangle ::= \langle \textit{factor} \rangle \mid \langle \textit{term} \rangle \langle \textit{mult operator} \rangle \langle \textit{factor} \rangle$$

On verra qu'une description d'un langage en utilisant la notation BNF correspond à une description par une grammaire hors-contexte.

Grammaire

On appelle grammaire tout quadruplet $G = (N, T, S, P)$ où

- N est un ensemble fini de symboles non terminaux
- T est un ensemble fini de symboles terminaux, $N \cap T = \emptyset$
- $S \in N$ est le symbole de départ
- P est un ensemble fini de productions. Chaque production est de la forme $x \longrightarrow y$ est constituée d'une partie gauche $x \in (N \cup T)^*$ et d'une partie droite $y \in (N \cup T)^*$

Exemple

$G = (\{S\}, \{a, b\}, S, \{S \longrightarrow SS, S \longrightarrow ab, S \longrightarrow \epsilon\})$

Les grammaires sont utilisées comme moyens pour décrire les langages, tel que chaque grammaire produit un langage unique.

Dérivations

- m' est appelé une dérivation immédiate de m au rapport de la grammaire $G = (N, T, S, P)$, noté $m \Rightarrow_G m'$ ou $m \Rightarrow m'$, si $m, m' \in (N \cup T)^*$ et $\exists p_1, p_2 \in (N \cup T)^*$ tels que $p \rightarrow q \in P$, $m = p_1 p p_2$ et $m' = p_1 q p_2$
- m' est appelé une dérivation de m au rapport de la grammaire $G = (N, T, S, P)$, noté $m \Rightarrow_G^* m'$ ou $m \Rightarrow^* m'$, si $m = m'$ ou si il y a une séquence $m_0, m_1, \ldots, m_r$ telle que $m = m_0$, $m_{i-1} \Rightarrow m_i$ pour chaque $i \in \{1, 2, \ldots, r\}$ et $m_r = m'$.

On écrit $m_0 \Rightarrow_G m_1 \Rightarrow_G \cdots \Rightarrow_G m_r oum \Rightarrow^r m'$ et on l'appelle une dérivation de longueur r.

Langage engendré

Le langage engendré par la grammaire $G = (N, T, S, P)$, noté $L(G)$ est défini par :

$$L(G) = \{m | S \Rightarrow^* m\} \cap T^*$$

Exemple de grammaire

Soit $G = (\{E, F, T\}, \{a, +, *, (,)\}, E, P)$, et P est

- $E \rightarrow E + T | T$
- $T \rightarrow T * F | F$
- $F \rightarrow (E) | a$

Exemple de dérivation

$$\begin{aligned} E &\Rightarrow E + T \\ &\Rightarrow T + T \\ &\Rightarrow F + T \\ &\Rightarrow a + T \\ &\Rightarrow a + T * F \\ &\Rightarrow a + F * F \\ &\Rightarrow a + a * F \\ &\Rightarrow a + a * a \end{aligned}$$

Constatons que chaque grammaire produit un langage unique. Au contraire, chaque langage (engendré par une grammaire) est engendré par un nombre infini de grammaires.

Grammaires équivalentes

On dit que deux grammaires G et G' sont équivalentes et on note $G \sim G'$, si $L(G) = L(G')$.

Il existe une hiérarchie de familles de langages nommée Chomsky-hiérarchie, d'une importance énorme dans la théorie des langages :

- générales
- contextuelles
- hors-contexte
- régulières
- à choix finis

Deux de ces familles sont très utilisées pour l'analyse lexicale et l'analyse syntaxique : les langages hors-contexte et les langages réguliers.

Grammaire hors-contexte

Une grammaire $G = (N, T, S, P)$ est dite hors-contexte si pour chaque production $p \longrightarrow q$ de P la partie gauche p est constituée d'un seul non-terminal, c'est à dire $p \in N$ et $q \in (N \cup T)^*$.

Grammaire régulière

Une grammaire $G = (N, T, S, P)$ est dite régulière si chaque production $p \longrightarrow q$ de P est de la forme $A \longrightarrow \epsilon$, $A \longrightarrow a$, ou $A \longrightarrow aB$ où $A, B \in N$ et $a \in T$.

Langage hors-contexte

Un langage L est dit hors-contexte s'il existe une grammaire G hors-contexte telle que $L = L(G)$.

Langage régulier

Un langage L est dit régulier s'il existe une grammaire G régulière telle que $L = L(G)$.

Remarque

Évidemment, pour chaque langage L régulier ou hors-contexte il existe une grammaire G qui n'est pas hors-contexte satisfaisant $L = L(G)$.

On constate que par définition chaque grammaire régulière est aussi une grammaire hors-contexte. Alors un langage régulier quelconque est aussi hors-contexte.

Exemple

- $L_1 = \{(ab)^n \mid n \in \mathbb{N}\}$ est régulier
- $L_2 = \{a^n b^n \mid n \in \mathbb{N}\}$ est hors-contexte

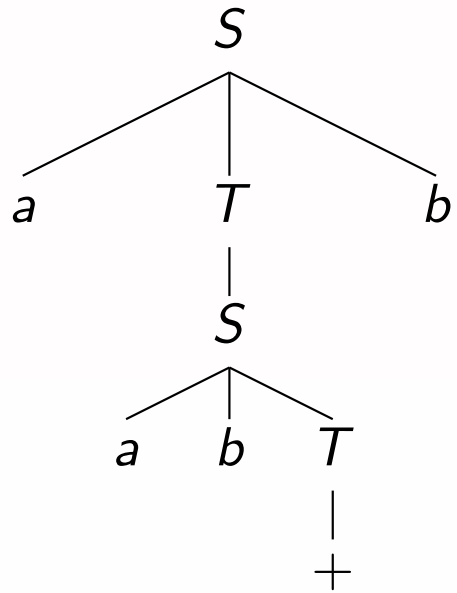
Arbre d'analyse

Les grammaires régulières et hors-contexte n'ont qu'un seul sybole à gauche de leurs règles de production. On peut donc construire un arbre d'analyse où :

- La racine est le symbole de départ.
- Pour chaque production :
  - Le symbole à gauche de la production est le père.
  - Chaque symbole à droite est un fils.
- Les noeuds internes sont des non-terminaux.
- Les feuilles sont des terminaux.

Exemple

Soit la grammaire :
$S \rightarrow aTb \mid abT$
$T \rightarrow + \mid S \mid \epsilon$
Et ceci est l'arbre d'analyse de $aab + b$



- On peut lire le mot généré en concaténant les symboles aux feuilles de gauche à droite.
- L'arbre d'analyse montre que plusieurs dérivations sont possibles pour la même chaîne.

Les unités lexicales d'un programmes sont significativement plus facilement structurables que les structures d'un rang hiérarchique (affectations, boucles, déclarations, etc.). L'analyse lexicale est donc plus simple que l'analyse syntaxique. Ce qui fait que l'analyse syntaxique est la partie la plus attractive de la compilation.

Typiquement :

- unités lexicales : grammaires régulières ou expressions régulières
- structures syntaxiques : grammaires hors-contexte

Les langages correctement parenthésés ne peuvent pas être décrits avec des expressions régulières.

Il existe plusieurs manière de décrire un langage, l'une d'entre elles utilise les expressions régulières. Chaque expression régulière r dénote un langage $L(r)$.

Expression Régulière

On définit une expression régulière sur Σ récursivement :

- ϵ est l'expression régulière dénotant le langage $\{\epsilon\}$
- si $a \in \Sigma$ alors a est l'expression régulière dénotant le langage $\{a\}$
- si r et s sont deux expressions régulières dénotant les langages $L(r)$ et $L(s)$:
  - $(r)|(s)$ est une expression régulière dénotant le langage $L(r) \cup L(s)$
  - $(r)(s)$ est une expression régulière dénotant le langage $L(r) \cdot L(s)$
  - $(r)^*$ est une expression régulière dénotant le langage $(L(r))^*$
  - (r) est une expression régulière dénotant le langage $L(r)$

Pour éviter les parenthèse superflues, on admettra par convention que l'étoile a la plus grande priorité, suivie de la concaténation, puis de l'union. Ils sont tous associatifs gauche.

Propriétés algébriques

$r\|s = s\|r$	$\|$ est commutatif
$r\|(s\|t) = (r\|s)\|t$	$\|$ est associatif
$(rs)t = r(st)$	la concaténation est associative
$r(s\|t) = rs\|rt$ $(s\|t)r = sr\|tr$	la concaténation est distributive par rapport à $\|$
$\emptyset r = r$ $r\epsilon = r$	ϵ est l'élément neutre pour la concaténation
$r^* = (r\|\epsilon)^*$	relation entre $*$ et ϵ
$r^{**} = r^*$	$*$ est idempotent

Exemples

- L'expression régulière $a|b$ dénote l'ensemble $\{a, b\}$
- L'expression régulière $(a|b)(a|b)$ dénote l'ensemble $\{aa, ab, ba, bb\}$, c'est à dire l'ensemble des chaînes de a et de b de longueur deux. Trouvez une autre expression régulière pour cet ensemble.
- L'expression régulière a^* dénote l'ensemble $\{\epsilon, a, aa, aa, \ldots\}$, c'est à dire l'ensemble des chaînes constituées d'un nombre quelconque (éventuellement nul) de a
- L'expression régulière $(a|b)^*$ dénote l'ensemble de toutes les chaînes constituées d'un nombre quelconque (éventuellement nul) de a ou de b. Trouvez une autre expression régulière pour cet ensemble.
- L'expression régulière $a|a^*b$ dénote l'ensemble constitué de la chaîne a et de toutes les chaînes commençant par un nombre quelconque (éventuellement nul) de a et se terminant par un b.

Pour des commodités de notation, on peut souhaiter donner des noms aux expressions régulières en utilisant ces noms comme s'ils étaient des symboles.

Définition régulière

Soit Σ un alphabet, une définition régulière est une suite de définitions de la forme :
$d_1 \rightarrow r_1,\ d_2 \rightarrow r_2,\ \ldots,\ d_n \rightarrow r_n$
où chaque d_i est un nom distinct est chaque r_i est une expression régulière sur l'alphabet $\Sigma \bigcup \{d_1, d_2, \ldots, d_{i-1}\}$, c'est à dire les caractères d'alphabet de base Σ sont les noms définis jusqu'alors.

Souvent une définition régulière est plus facile à trouver et à lire qu'une expression régulière (équivalente à r_n).

Étant donné une définition régulière, on peut construire une expression régulière sur Σ pour chaque r_i en remplaçant, de manière répétitive, les noms des expressions régulières par des expressions qu'ils dénotent.

Exemple

Les flottants non signés en Pascal, en C, etc. sont des chaînes comme 3543, $42,28$, $6,022E23$. La définition régulière ci-dessous donne une spécification précise de chaîne de ce langage sur l'alphabet $\{0,1,\ldots,9,+,-,E,.\}$:

$$
\begin{aligned}
\textit{chiffre} &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\textit{chiffres} &\rightarrow \textit{chiffre}\ \textit{chiffre}^* \\
\textit{fractionopt} &\rightarrow .\ \textit{chiffres} \mid \epsilon \\
\textit{exposantopt} &\rightarrow (E(+ \mid - \mid \epsilon)\ \textit{chiffres}) \mid \epsilon \\
\textit{nb} &\rightarrow \textit{chiffres}\ \textit{fractionopt}\ \textit{exposantopt}
\end{aligned}
$$

Extensions

Certaines constructions apparaissent si fréquemment dans les expressions régulières qu'il est pratique d'introduire des notations abrégées pour les représenter :

- Au moins une instance : opérateur $+$. Si r est une expression régulière dénotant le langage $L(r)$, alors $(r)^+$ est une expression régulière dénotant le langage $(L(r))^+$. On a $r^* = r^+|\epsilon$ et $r^+ = rr^*$.
- Zéro ou une instance : opérateur ?. Si r est une expression régulière dénotant le langage $L(r)$, alors $(r)?$ est une expression régulière dénotant le langage $L(r) \bigcup \{\epsilon\}$.
- Classes de caractères : la notation $[abc]$ où a, b et c sont des symboles de l'alphabet, dénote l'expression régulière $a|b|c$. Une classe de caractères comme $[a-z]$ dénote l'expression régulière $a|b|\ldots|z$.
- N'importe quel caractère : le symbole . représente n'importe quel symbole de l'alphabet.

Autres extensions introduites par le langage qui les utilisent : `http://en.wikipedia.org/wiki/Regular_expression`

Exemple

L'exemple précédent devient avec ces abréviations :

$$
\begin{aligned}
\textit{chiffre} &\longrightarrow [0-9] \\
\textit{chiffres} &\longrightarrow \textit{chiffre}^{+} \\
\textit{fractionopt} &\longrightarrow (.\ \textit{chiffres})? \\
\textit{exposantopt} &\longrightarrow (E(+ \mid - \mid \epsilon)\ \textit{chiffres})? \\
\textit{nb} &\longrightarrow \textit{chiffres}\ \textit{fractionopt}\ \textit{exposantopt}
\end{aligned}
$$

Équivalence des grammaires régulières et des expressions régulières :

Théorème 1

Soient L, L_1 et L_2 des langages réguliers, alors $L_1 \cup L_2$, $L_1 L_2$ et L^* sont aussi réguliers.

Preuve

Soient G, G_1, G_2 des grammaires telles que $L = L(G)$, $L_1 = L(G_1)$ et $L_2 = L(G_2)$. On peut construire facilement des grammaires régulières G_{union}, G_{concat} et $G_{étoile}$ telles que :

- $L_1 \cup L_2 = L(G_{union})$
- $L_1 L_2 = L(G_{concat})$
- $L^1 = L(G_{étoile})$

Théorème 2

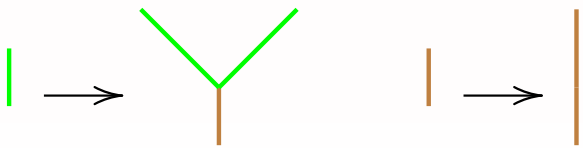
Les langages définis par des expressions régulières sont précisément les langages réguliers.

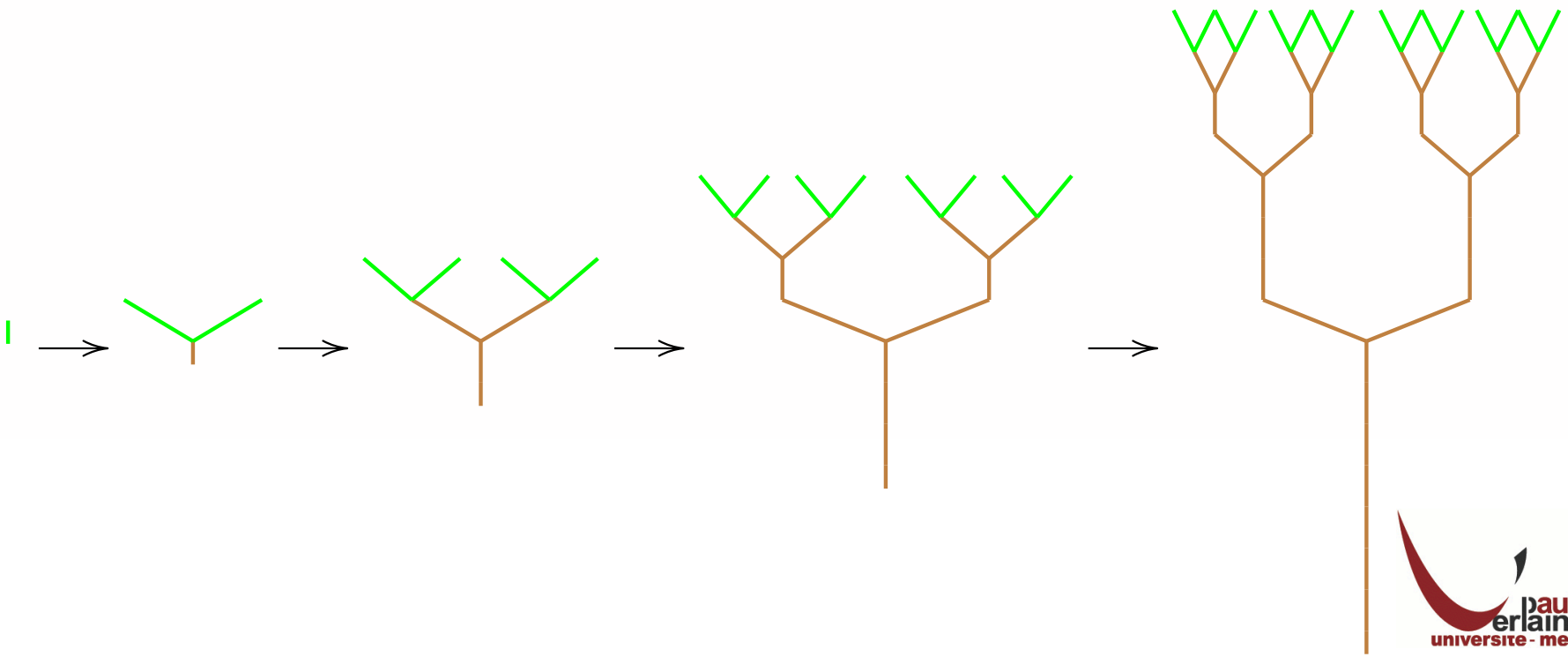
Preuve

La démonstration de ce dernier théorème se fait par une procédure qui traduit une grammaire régulière G en une expression régulière r telle que dénote $L(G)$.

L-systèmes

Lindenmayer utilise des grammaires pour étudier la croissance de végétaux.
`http://algorithmicbotany.org/papers/`
Livre : The Algorithmic Beauty of Plants (à télécharger gratuitement)





http://cgg.ms.mff.cuni.cz/thesis/novy/