Théorie des Langages

Expressions régulières et Grammaires

Claude Moulin

Université de Technologie de Compiègne

Printemps 2013

Sommaire

Sommaire

Alphabet

Définition :

Un alphabet est un ensemble fini non vide de symboles appelés également lettres de l'alphabet.

1. - $\Sigma_1 = \{0, 1\}$
2. - $\Sigma_2 = \{a, b, ..., z\}$

Chaîne

Définition :

Une chaîne est une suite finie de symboles choisis dans un alphabet.

1. $\Sigma_1 = \{0, 1\}$
   - 01011, 111, 00, ...
2. $\Sigma_2 = \{a, b, c, ..., z\}$
   - abc, aa, xyzt, ...

ϵ : chaîne vide ; chaîne ne contenant aucun symbole.

Fonction - Opération

- La longueur d'une chaîne est le nombre de symboles la composant.
  $|abc| = 3$;
  $|\epsilon| = 0$
- Concaténation
  $u = x_1 x_2 ... x_n$ et $v = y_1 y_2 ... y_m$ deux chaînes $\neq \epsilon$ définies sur le même alphabet Σ,
  $uv = x_1 x_2 ... x_n y_1 y_2 ... y_m$ concaténation de u et de v.

Concaténation

- Propriétés de la concaténation
  - associative : $\forall u, \forall v, \forall w, (uv)w = u(vw)$
  - élément neutre : ϵ
  - $|uv| = |u| + |v|$
  - non commutative. En général, $uv \neq vu$.
- Préfixe - Suffixe

  $\forall u \neq \epsilon, \forall v \neq \epsilon, \forall w \neq \epsilon$

  Lorsque $w = uv$, u est un préfixe de w et v est un suffixe de w.

Langage

Définition :

Un langage défini sur un alphabet Σ est un ensemble de chaînes définies sur Σ.

- Σ_2 est le langage composé de toutes les chaînes de longueur 2 sur Σ
- Σ^* est le langage composé de toutes les chaînes constructibles sur l'alphabet Σ
- Pour tout langage L construit sur un alphabet Σ on a : $L \subseteq \Sigma^*$

Opérations

- Réunion, Intersection de deux langages
  Soient L_1 et L_2 deux langages définis sur un alphabet Σ.
  $L_1 \cup L_2$ et $L_1 \cap L_2$ sont aussi deux langages définis sur Σ.
- Concaténation de deux langages
  Soient L_1 et L_2 deux langages définis sur un alphabet Σ.
  $L_1 L_2 = \{uv / u \in L_1, v \in L_2\}$ est le langage obtenu par concaténation de L_1 et de L_2.

Clôture de Kleene

Soit L un langage défini sur un alphabet Σ.

Clôture de Kleene

La clôture de Kleene du langage L, notée L^* est l'ensemble des chaînes qui peuvent être formées en prenant un nombre quelconque (éventuellement 0) de chaînes de L et en les concaténant.

- $L^0 = \{\epsilon\}$
- $L^1 = L$
- $L^2 = LL$
- $L^i = LL...L$, la concaténation de i fois L
- $L^* = \bigcup_{i \geq 0} L^i$

Sommaire

Opérateurs

Les expressions régulières sont une façon déclarative de décrire la formation des chaînes d'un langage.
On note $L(E)$ le langage engendré par une expression régulière E.

- Opérateur * (fermeture)
  - 0 * représente : $\{\epsilon, 0, 00, 000, \ldots\}$
  - 110*1 représente : $\{111, 1101, 11001, \ldots\}$
- Opérateur | (disjonction)
  - 0* | 1* représente les chaînes formées par 0* et par 1*
  - ϵ, 0, 00, 000, 1, 11, 111, ...

Règles

- la chaîne vide, ϵ, et l'ensemble vide, $\emptyset$, sont des expressions régulières.
  - $L(\epsilon) = \{\epsilon\}$ et $L(\emptyset) = \emptyset$.
- $\forall a \in \Sigma$, la chaîne formée de l'unique symbole a est une expression régulière.
  - $L(a) = \{a\}$.

Construction

- Si E et E' sont des expressions régulières alors :
  - EE' est une expression régulière, concaténation de E et E'. Elle décrit $L(EE') = L(E)L(E')$
  - $E \mid E'$ est une expression régulière. Elle décrit $L(E \mid E') = L(E) \cup L(E')$
  - (E) est une expression régulière, décrivant le même langage que E. $L((E)) = L(E)$.
  - E^* est une expression régulière. $L(E^*) = L(E)^*$

Exemples

- ab* ; a*b
- (ab)*
- (a | b)*
- a*b(a*b)*
- Questions
  - (a* | b*) = (a | b)* ?
  - (a | b)* b = (a | b)* ?
  - (a*b)* = (a | b)*b ?

Exemples

- Opérateur unaire $+$: au moins une fois
  $E^+ = EE^*$
- Opérateur unaire $?$: 0 ou 1 fois
  $E? = E \mid \epsilon$
- Classes de caractères :
  $[abc] = a \mid b \mid c$
  $[a-z] = a \mid b \mid c \mid ... \mid z$
  $[A-Ca-z] = A \mid B \mid C \mid a \mid b \mid c \mid ... \mid z$

Priorité des opérateurs

- + et * sont des opérateurs unaires et ont la plus forte priorité
- La concaténation est un opérateur binaire et a la seconde priorité
- | est un opérateur binaire et a la plus faible priorité
- Concaténation et | sont associatives à gauche
- Exemple :
  - (a) | ((b)*(c)) = a | b*c

Lois algébriques

- $E \mid E' = E' \mid E$
- $E \mid (E' \mid E'') = (E \mid E') \mid E''$
- $\emptyset \mid E = E \mid \emptyset = E$
- $E \mid E = E$
- $E(E'E'') = (EE')E''$
- $\epsilon E = E\epsilon = E$
- $\emptyset E = E\emptyset = \emptyset$
- $E(E' \mid E'') = EE' \mid EE''$
- $(E \mid E')E'' = EE'' \mid E'E''$

Conséquences

- $(E^*)* = E^*$
- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $E^+ = EE^* = E^*E$
- $E^+ \mid \epsilon = E^*$
- $E? = E \mid \epsilon$
- $p(qp)* = (pq) * p$
- $(p|q)* = (p * |q*)*$
- $(p * q*)* = (p|q)* = (p * |q*)*$

Définitions régulières

- lettre $\longrightarrow$ [A-Za-z]
- chiffre $\longrightarrow$ [0-9]
- id $\longrightarrow$ lettre (lettre | chiffre)*
- chiffres $\longrightarrow$ chiffre (chiffre)*
- fraction $\longrightarrow$. chiffres | ϵ
- exposant $\longrightarrow$ E(+ | - | ϵ) chiffres | ϵ
- nombre $\longrightarrow$ chiffres fraction exposant

- id reconnaît : a, a0b, begin
- nombre reconnaît : 0, 1.0, 2E4, 1.5E-8, 0.25E-0
- nombre ne reconnaît pas : 0., .1, 1E2.0

Sommaire

ER et langages - 1

- Dans les langages de programmation les expressions régulières ont deux utilisations :
  - Elles permettent de vérifier qu'une chaîne respecte une forme donnée ;
  - Elles permettent de rechercher des chaînes d'un type particulier à l'intérieur d'un texte.

ER et langages - 2

- Les expressions régulières étendent les opérateurs de base avec :
  - des classes de caractères ;
  - des définitions de groupes ;
  - des conditions de limites (mots, ligne, fichier) ;
  - des facteurs de répétitions ;
  - des motifs prospectifs.

Classes de caractères - 1

[...]	Union des caractères entre les crochets
[^...]	Complémentaire des symboles entre les crochets
[$x - y$]	L'un des symboles entre x et y
\d	[0-9]
\D	[^0-9]
\w	[a-zA-Z0-9]
\W	[^a-zA-Z0-9]
\s ou \p{Space}	[\t\n\x0B\f\r]
\S	[^\t\n\x0B\f\r]

Exemple :

\d\D\w	20-juin-2012	0-j

Classes de caractères - 2

\p{Lower}	Un caractère minuscule
\p{Upper}	Un caractère majuscule
\p{ASCII}	Un caractère dont le code ascii est compris entre 0 et 128
\p{Alpha}	Un caractère minuscule ou majuscule
\p{Alnum}	\p{Alpha} ∪ \p{Digit}
\p{Punct}	`[! "#$%&'()*+,-./:;<=>?@[\]^_`{|}~]`
\p{Print}	\p{Alnum} ∪ \p{Punct}
\p{Blank}	Caractères espace ou tabulation
\p{XDigit}	Un chiffre hexadécimal `[0-9a-fA-F]`
.	Un caractère quelconque
$< cl1 > \mid < cl2 >$	Union des classes cl1 et cl2
$[< cl1 > \&\& < cl2 >]$	Intersection des classes cl1 et cl2

Groupe - 1

- Les parenthèses dans une expression régulière sont des méta symboles permettant de délimiter des groupes dans une chaîne satisfaisant l'expression.
- Un groupe est une suite de caractères à l'intérieur d'une chaîne satisfaisant l'expression.
- Chaque groupe possède un rang unique.
- Le rang d'un groupe dans une expression régulière est égal au nombre de parenthèses ouvrantes le précédant.

Groupe - 2

- Un groupe peut-être non capturant. Il n'a pas de rang attribué.
- Le rang d'un groupe est égal au nombre de parenthèses ouvrantes le précédant - le nombre de groupes non capturant le précédant.
- Il est possible de faire référence à un groupe capturant précédent et de l'insérer dans le patron.

(. . .)	Groupe capturant
(? :. . .)	Groupe non capturant
\n	Référence arrière au groupe capturant de rang n

Conditions aux limites

^ ou \a	Début de chaîne
$ ou \z	Fin de chaîne
(?m)^	Début de ligne ou de chaîne
(?m)$	Fin de ligne ou de chaîne
\b	Début ou fin de mot
\B	Absence de début ou de fin de mot
\G	Fin du précédent groupe satisfaisant le gabarit

Exemple 1 :

\w-(\d(\d))	20-juin-2012	n-20	20	0

Exemple 2 :

\d\b	20-juin-2012	0	2

Facteurs de répétition

{n}	Exactement n occurrences d'une classe de caractères
+	Au moins 1 occurrence d'une classe de caractères
{n,}	Au moins n occurrences d'une classe de caractères
{n,p}	De n à p occurrences d'une classe de caractères
?	0 ou 1 occurrence
*	Un nombre quelconque de fois, éventuellement zéro

Opérateurs avides (gloutons)

Opérateur avide

Un opérateur est dit avide lorsque l'algorithme sous-jacent utilise le maximum de caractères pour satisfaire le facteur de répétition en tenant compte du contexte.

*, +, {n,p} sont avides.

Patron	\d {2,6}3 \d+
Chaîne	25331
\d {2,6}	253

Opérateurs passifs

Opérateur passif

Un opérateur est dit passif lorsque l'algorithme sous-jacent utilise le minimum de caractères pour satisfaire le facteur de répétition en tenant compte du contexte.

{n,p} ?, * ?, + ? sont passifs.

Patron	\d {2, 6}?3 \d+
Chaîne	25331
\d {2, 6} ?	25

Opérateurs possessifs

Opérateur possessif

Un opérateur est dit possessif lorsque l'algorithme sous-jacent utilise le maximum de caractères pour satisfaire le facteur de répétition sans tenir compte du contexte.

{n,p}+, *+, ++ sont possessifs.

Patron	\d {2, 7} +3 \d{2}
Chaîne	25331734
\d {2, 7}+	2533173
3 et \d{2}	Non satisfait

Motifs prospectifs

- Les motifs prospectifs (lookahead) servent à tester si un élément d'une expression régulière est suivi d'un sous motif particulier.

(?=...)	Prévision positive avant
(?!...)	Prévision négative avant
(?<=...)	Prévision positive arrière
(?<!...)	Prévision négative arrière

Code Java

```
public static void test1() {
  String patternString = "\\d+";
  String text = "rge5r43";
  Pattern p = Pattern.compile(patternString);
  Matcher m = p.matcher(text);
  boolean found = m.find();
  System.out.println("Succès : " + found);
  System.out.println("Position début : " + m.start());
  System.out.println("Sélection : " + m.group());
  // System.out.println("Groupe : " + m.group(1));
  System.out.println("Position fin : " + m.end());
}
```

Méthodes de la classe Matcher

- `find` : analyse la chaîne d'entrée en cherchant la prochaine séquence qui satisfait le patron.
- `matches` : essaie de satisfaire la chaîne entière en regard du patron.
- `start` : retourne le rang du premier caractère de la séquence qui satisfait le patron.
- `end` : retourne le rang du dernier caractère de la séquence qui satisfait le patron.
- `group` : retourne la chaîne qui satisfait le patron.
- `groupCount` : retourne le nombre de groupes dans la chaîne qui satisfait le patron.
- `group(int n)` : retourne le groupe de rang n dans la chaîne qui satisfait le patron.

Sommaire

Exercice 1

- Une séquence de 1 à 3 lettres, suivie d'une séquence de 1 à deux chiffres, suivie de la même séquence de chiffres suivie de la première séquence de lettres.
- Exemples : ab55ab, -abC4141aBc-

Exercice 1

- Une séquence de 1 à 3 lettres, suivie d'une séquence de 1 à deux chiffres, suivie de la même séquence de chiffres suivie de la première séquence de lettres.
- Exemples : ab55ab, -abC4141aBc-

Réponse : (?i)(\w{1,3})(\d{1,2})\2\1

Exercice 2

- Test d'un mot de passe : il doit contenir au moins un chiffre, au moins une lettre et au moins un caractère de ponctuation. De plus il doit avoir au moins 8 caractères.
- Exemple : abce.fr/ty2

Exercice 2

- Test d'un mot de passe : il doit contenir au moins un chiffre, au moins une lettre et au moins un caractère de ponctuation. De plus il doit avoir au moins 8 caractères.
- Exemple : abce.fr/ty2

Réponse : $(?i)(?=.*\d)(?=.*[a-z])(?=.*\p\{Punct\}).\{8,\}$

Sommaire

Sommaire

Langages non réguliers

- Le langage : $L_{01} = \{0^n 1^n \mid n \geq 1\}$ est non régulier
- Raisonnement par l'absurde : soit un AFD représentant ce langage.
  - Il existe deux nombres différents i et j tels que après avoir lu les préfixes 0^i et 0^j, l'AFD soit dans le même état.
  - A partir de cet état, l'AFD lit un certain nombre de 1 et arrive dans un état final.
  - On aurait $0^i 1^p \in L_{01}$ et $0^j 1^p \in L_{01}$ avec $i \neq j$
- Les expressions régulières ne sont pas suffisantes pour représenter les langages de programmation

Grammaire

$S \longrightarrow \epsilon$
$S \longrightarrow 0$
$S \longrightarrow 1$
$S \longrightarrow 0S0$
$S \longrightarrow 1S1$

Cette grammaire permet de définir le langage des palindromes sur $\{0, 1\}$

Niveau 1 : ϵ, 0, 1
Niveau 2 : 00, 000, 010, 11, 101, 111
Niveau 3 :
0000, 00000, 00100, 0110, 01010, 01110
1001, 10001, 10101, 1111, 11011, 11111

Définition

- T : un alphabet, ou ensemble des symboles terminaux ou terminaux.
- V : Un ensemble fini non vide de variables, ou non terminaux ou catégories syntaxiques. $T \cap V = \emptyset$.
- S, unique : une variable appelée symbole de départ
- P : un ensemble fini non vide de règles de production
  - de forme $A \longrightarrow \alpha$
  - où A est une variable et α une chaîne de 0 ou plus terminaux et variables.
  - Les parties gauche et droite de la flèche sont appelées la tête et le corps de la production.
  - $P \subset \{(A, \alpha) \mid A \in V; \alpha \in (V \cup T)*\}$

Grammaire des palindromes

$T = \{0, 1\}$
$V = \{S\}$
S : symbole de départ

$P = \{$
$S \longrightarrow \epsilon$
$S \longrightarrow 0$
$S \longrightarrow 1$
$S \longrightarrow 0S0$
$S \longrightarrow 1S1$
$\}$

Exemple

T = { d_m, d_f, nom_m, nom_f, vb }
W = { PH, GN, N, V, D }
S = PH

$$
\begin{aligned}
PH &\longrightarrow GN\ V\ GN \\
GN &\longrightarrow D\ N \\
N &\longrightarrow nom\_m \\
N &\longrightarrow nom\_f \\
D &\longrightarrow d\_m \\
D &\longrightarrow d\_f \\
V &\longrightarrow vb
\end{aligned}
$$

T = {a, e, b, c, d, g, h, l, r, t, ' '}
W = { s, d_m, d_f, nom_m, nom_f, vb }
S = s

$$
\begin{aligned}
s &\longrightarrow d\_m \mid d\_f \mid nom\_m \\
&\quad \mid nom\_f \mid vb \mid ' ' \\
d\_m &\longrightarrow l\ e \\
d\_f &\longrightarrow l\ a \\
nom\_m &\longrightarrow c\ h\ a\ t \\
nom\_f &\longrightarrow b\ a\ l\ l\ e \\
vb &\longrightarrow r\ e\ g\ a\ r\ d\ e
\end{aligned}
$$

Exemple

T = { d_m, d_f, nom_m, nom_f, vb }
W = { PH, GN, N, V, D }
S = PH

$$
\begin{aligned}
PH &\longrightarrow GN\ V\ GN \\
GN &\longrightarrow D\ N \\
N &\longrightarrow nom\_m \\
N &\longrightarrow nom\_f \\
D &\longrightarrow d\_m \\
D &\longrightarrow d\_f \\
V &\longrightarrow vb
\end{aligned}
$$

T = {a, e, b, c, d, g, h, l, r, t, ' '}
W = { s, d_m, d_f, nom_m, nom_f, vb }
S = s

$$
\begin{aligned}
s &\longrightarrow d\_m \mid d\_f \mid nom\_m \mid nom\_f \mid vb \mid ' ' \\
d\_m &\longrightarrow l\ e \\
d\_f &\longrightarrow l\ a \\
nom\_m &\longrightarrow c\ h\ a\ t \\
nom\_f &\longrightarrow b\ a\ l\ l\ e \\
vb &\longrightarrow r\ e\ g\ a\ r\ d\ e
\end{aligned}
$$

- le chat regarde la balle (sémantiquement correct)

Exemple

T = { d_m, d_f, nom_m, nom_f, vb }
W = { PH, GN, N, V, D }
S = PH

$$
\begin{aligned}
PH &\longrightarrow GN\ V\ GN \\
GN &\longrightarrow D\ N \\
N &\longrightarrow nom\_m \\
N &\longrightarrow nom\_f \\
D &\longrightarrow d\_m \\
D &\longrightarrow d\_f \\
V &\longrightarrow vb
\end{aligned}
$$

T = {a, e, b, c, d, g, h, l, r, t, ' '}
W = { s, d_m, d_f, nom_m, nom_f, vb }
S = s

$$
\begin{aligned}
s &\longrightarrow d\_m \mid d\_f \mid nom\_m \mid nom\_f \mid vb \mid ' ' \\
d\_m &\longrightarrow l\ e \\
d\_f &\longrightarrow l\ a \\
nom\_m &\longrightarrow c\ h\ a\ t \\
nom\_f &\longrightarrow b\ a\ l\ l\ e \\
vb &\longrightarrow r\ e\ g\ a\ r\ d\ e
\end{aligned}
$$

- la balle regarde le chat (sémantiquement incorrect)

Exemple

T = { d_m, d_f, nom_m, nom_f, vb }
W = { PH, GN, N, V, D }
S = PH

$$
\begin{aligned}
PH &\longrightarrow GN\ V\ GN \\
GN &\longrightarrow D\ N \\
N &\longrightarrow nom\_m \\
N &\longrightarrow nom\_f \\
D &\longrightarrow d\_m \\
D &\longrightarrow d\_f \\
V &\longrightarrow vb
\end{aligned}
$$

T = {a, e, b, c, d, g, h, l, r, t, ' '}
W = { s, d_m, d_f, nom_m, nom_f, vb }
S = s

$$
\begin{aligned}
s &\longrightarrow d\_m \mid d\_f \mid nom\_m \mid nom\_f \mid vb \mid ' ' \\
d\_m &\longrightarrow l\ e \\
d\_f &\longrightarrow l\ a \\
nom\_m &\longrightarrow c\ h\ a\ t \\
nom\_f &\longrightarrow b\ a\ l\ l\ e \\
vb &\longrightarrow r\ e\ g\ a\ r\ d\ e
\end{aligned}
$$

- le chat regarde le balle (accord incorrect)

Dérivation

$G = (T, V, S, P)$ une grammaire ; $\alpha A \beta$ une chaîne :

- $\alpha \in (V \cup T)^*$
- $\beta \in (V \cup T)^*$
- $A \in V$

Dérivation

$G = (T, V, S, P)$ une grammaire ; $\alpha A \beta$ une chaîne :

- $\alpha \in (V \cup T)^*$
- $\beta \in (V \cup T)^*$
- $A \in V$

$A \longrightarrow \gamma$ une règle de production de G avec $\gamma \in (V \cup T)^*$

Dérivation

$G = (T, V, S, P)$ une grammaire ; $\alpha A \beta$ une chaîne :

- $\alpha \in (V \cup T)^*$
- $\beta \in (V \cup T)^*$
- $A \in V$

$A \longrightarrow \gamma$ une règle de production de G avec $\gamma \in (V \cup T)^*$

$\alpha A \beta$ se dérive en $\alpha\gamma\beta$: $(\alpha A \beta \underset{G}{\Longrightarrow} \alpha\gamma\beta)$ ou $(\alpha A \beta \Longrightarrow \alpha\gamma\beta)$

Dérivation

$G = (T, V, S, P)$ une grammaire ; $\alpha A \beta$ une chaîne :

- $\alpha \in (V \cup T)^*$
- $\beta \in (V \cup T)^*$
- $A \in V$

$A \longrightarrow \gamma$ une règle de production de G avec $\gamma \in (V \cup T)^*$

$\alpha A \beta$ se dérive en $\alpha\gamma\beta$: $(\alpha A \beta \underset{G}{\Longrightarrow} \alpha\gamma\beta)$ ou $(\alpha A \beta \Longrightarrow \alpha\gamma\beta)$

Exemple : $S \underset{G}{\Longrightarrow} 0S0 \underset{G}{\Longrightarrow} 00S00 \underset{G}{\Longrightarrow} 001S100 \underset{G}{\Longrightarrow} 0011100$

Extension

On étend la relation $\Longrightarrow$ en $\overset{*}{\Longrightarrow}$ pour représenter 0, 1 ou plusieurs pas de dérivation.

Pour tout $\alpha \in (V \cup T)^*$ on écrit :

- $(\alpha \overset{*}{\Longrightarrow} \alpha)$
- Si $(\alpha \overset{*}{\Longrightarrow} \beta)$ et $(\beta \Longrightarrow \gamma)$ alors $(\alpha \overset{*}{\Longrightarrow} \gamma)$
- $S \overset{*}{\Longrightarrow} 001S100$

Définition

Défintion

Le langage défini par la grammaire $G = (T, V, S, P)$ appelé $L(G)$ est l'ensemble des chaînes dérivées du symbole de départ :
$L(G) = \{w \in T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$

- Etant donnée une grammaire $G = (T, V, S, P)$, un syntagme α est une chaîne de $(T \cup V)*$ telle que $S \overset{*}{\Longrightarrow} \alpha$.
- Une dérivation terminale w est un syntagme appartenant à $T*$: $S \overset{*}{\Longrightarrow} w$ (un programme correct).

Sommaire

Réécriture des règles

- $I \longrightarrow a$
- $I \longrightarrow b$
- se réécrivent en : $I \longrightarrow a \mid b$
- $I \longrightarrow \epsilon$
- $I \longrightarrow b$
- se réécrivent en : $I \longrightarrow b$?

Dérivation la plus à gauche

$T = \{a, b, 0, 1, +, \times, (,)\}$. Variables : $V = \{E, I\}$.

Symbole de départ E.

Règles de production P :

$E \longrightarrow I \mid E + E \mid E \times E \mid (E)$

$I \longrightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Dérivation la plus à gauche de : $a \times (b + a0)$

$$E \underset{lm}{\Longrightarrow} E \times E \underset{lm}{\Longrightarrow} I \times E \underset{lm}{\Longrightarrow} a \times E \underset{lm}{\Longrightarrow} a \times (E) \underset{lm}{\Longrightarrow} a \times (E + E)$$
$$\underset{lm}{\Longrightarrow} a \times (I + E) \underset{lm}{\Longrightarrow} a \times (b + E) \underset{lm}{\Longrightarrow} a \times (b + I)$$
$$\underset{lm}{\Longrightarrow} a \times (b + I0) \underset{lm}{\Longrightarrow} a \times (b + a0)$$

Dérivation la plus à droite

$T = \{a, b, 0, 1, +, \times, (,)\}$. Variables : $V = \{E, I\}$.

Symbole de départ E.

Règles de production P :

$E \longrightarrow I \mid E + E \mid E \times E \mid (E)$

$I \longrightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Dérivation la plus à droite de : $a \times (b + a0)$

$$E \underset{rm}{\Longrightarrow} E \times E \underset{rm}{\Longrightarrow} E \times (E) \underset{rm}{\Longrightarrow} E \times (E + E) \underset{rm}{\Longrightarrow} E \times (E + I)$$
$$\underset{rm}{\Longrightarrow} E \times (E + I0) \underset{rm}{\Longrightarrow} E \times (E + a0) \underset{rm}{\Longrightarrow} E \times (I + a0)$$
$$\underset{rm}{\Longrightarrow} E \times (b + a0) \underset{rm}{\Longrightarrow} I \times (b + a0) \underset{rm}{\Longrightarrow} a \times (b + a0)$$

Arbre de dérivation

$T = \{a, b, 0, 1, +, \times, (,)\}$.

Variables : $V = \{E, I\}$.

Symbole de départ E.

Règles de production P :

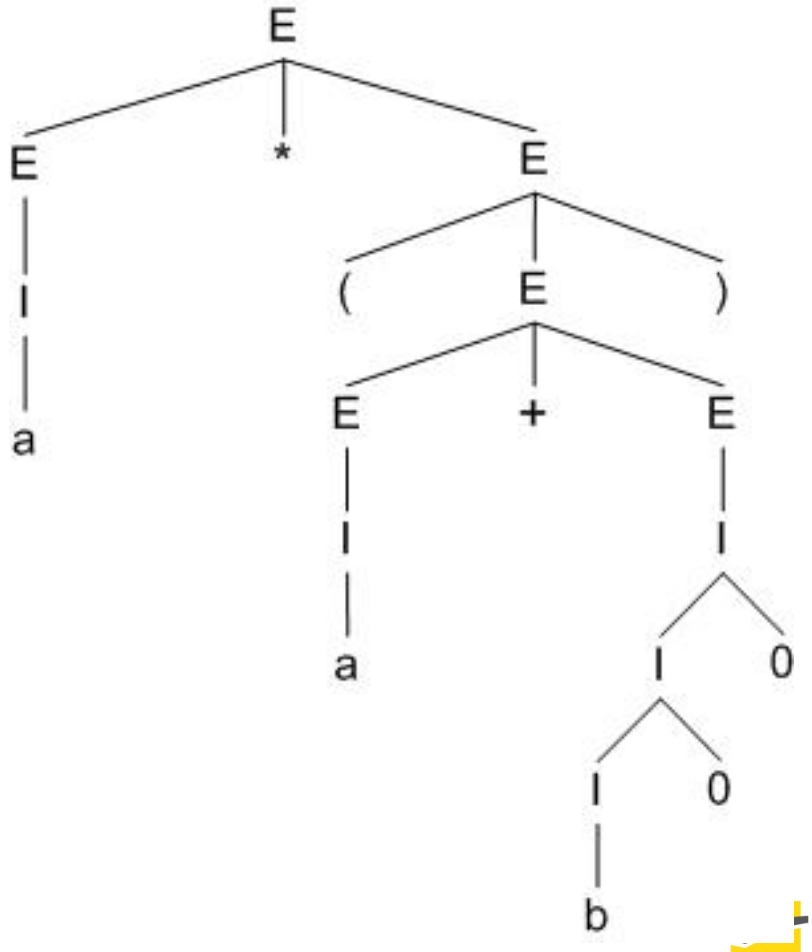
$E \longrightarrow I \mid E + E$

$E \longrightarrow E \times E \mid (E)$

$I \longrightarrow a \mid b \mid Ia \mid Ib$

$I \longrightarrow I0 \mid I1$

$a \times (a + b00)$



Définition : Arbre de dérivation

Définition

On appelle arbre de dérivation (ou arbre syntaxique), tout arbre tel que :

- la racine est le symbole de départ.
- les feuilles sont des symboles terminaux ou ϵ.
- les nœuds sont des non-terminaux.
- les fils d'un nœud X sont $Y_0, Y_1, \ldots, Y_n$ si et seulement s'il existe une règle de production :
  $X \longrightarrow Y_0 Y_1 \ldots Y_n$ ($Y_i \in T \cup V$).

Ambiguïté

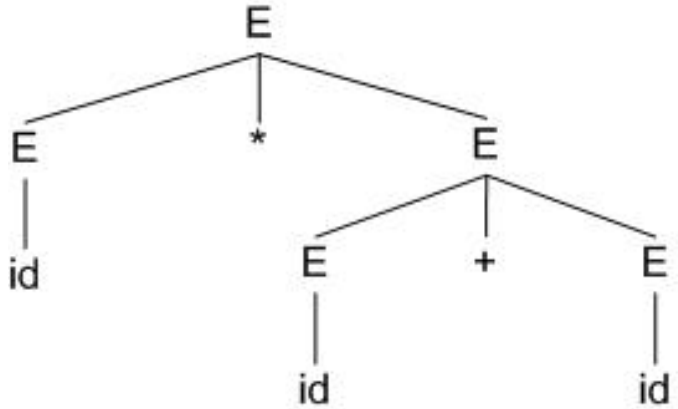
$T = \{id, +, \times, (,)\}$ Variables : $V = \{E\}$.

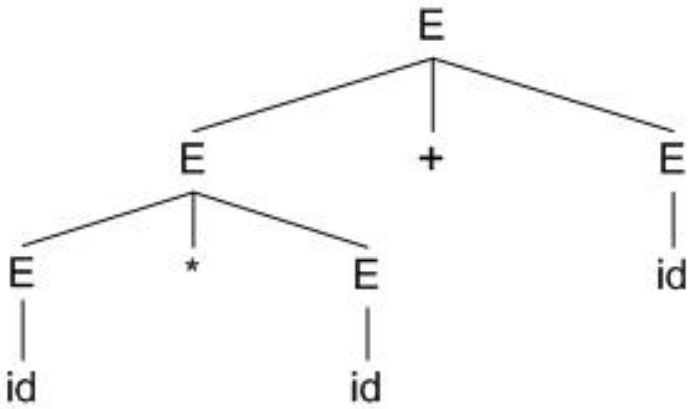
Symbole de départ E.

Règles de production :

$E \longrightarrow E + E \mid E \times E \mid (E) \mid id$

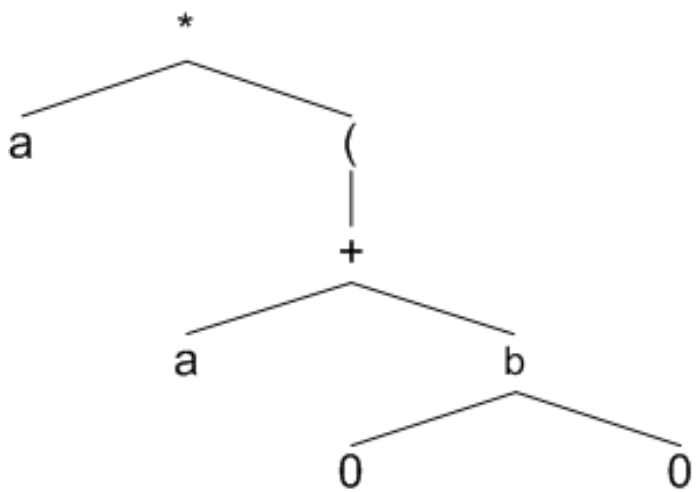
$id \times id + id$





Arbre Syntaxique Abstrait

$a \times (a + b00)$ peut être représenté par une autre structure d'arbre où les noeuds sont des symboles terminaux.



Sommaire

Forme de Backus-Naur (BNF)

- La notation de Backus-Naur a été utilisée dès 1960 pour décrire le langage ALGOL 60, et depuis est employée pour définir de nombreux langages de programmation.
- Le symbole des règles de réécriture est " : :=".
- Les symboles désignant les variables sont inclus entre chevrons : "$<$" et "$>$".
- Un ensemble de règles dont les parties gauches sont identiques, telles que :

  $$\begin{array}{llll} A & ::= & \alpha_1 & \| \\ A & ::= & \alpha_2 & \| \end{array} \quad \begin{array}{llll} A & ::= & \ldots & \\ A & ::= & \alpha_p & \end{array}$$

  peut être écrit : $A ::= \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_p$

Extension de la forme BNF

- Les items optionnels sont placés entre les méta symboles “[“ et “]”.

```
<if_statement> ::=  if <boolean_expression> then
                         <statement_sequence>
                    [ else
                         <statement_sequence> ]
                    end if ;
```

- Les items à répéter (opérateur *) sont placés entre les méta symboles “{” et “}”.

```
<identifier> ::= <letter> { <letter> | <digit> }
```

Grammaires de Type 3

- Grammaires régulières à droite.
  - Forme des règles : $A \longrightarrow aB$ ou $A \longrightarrow a$ avec $A, B \in V$ et $a \in T$.
- Grammaires régulières à gauche.
  - Forme des règles : $A \longrightarrow Ba$ ou $A \longrightarrow a$ avec $A, B \in V$ et $a \in T$.
- Le membre de gauche de chaque règle est constitué d'un seul symbole non terminal,
- Le membre de droite est constitué d'un symbole terminal éventuellement suivi (respectivement précédé) d'un seul non terminal.

Grammaires de Type 2

- Grammaires hors-contexte.
- Forme des règles :
  - $A \longrightarrow \alpha$
  - $A \in V$ et $\alpha \in (V \cup T)^*$.
  - α peut valoir ϵ
- Le membre de gauche de chaque règle est constitué d'un seul symbole non terminal.
- Le membre de droite est une séquence de symboles terminaux et de variables

Grammaires de Type 1

- Grammaires contextuelles (CSG).
- Forme des règles :
  - $\alpha A \beta \longrightarrow \alpha \gamma \beta$
  - $A \in V$; $\alpha, \beta \in (V \cup T)^*$ et $\gamma \in (V \cup T)^+$.
- Le symbole non terminal A n'est remplacé par la forme γ de $(V \cup T)^+$ que si les contextes α et β sont présents à gauche et à droite.
- La longueur de la partie gauche de toute règle est inférieure ou égale à celle de sa partie droite.
- Exception : la forme $S \longrightarrow \epsilon$ est autorisée si S n'apparaît pas dans la partie droite d'une règle (seul l'axiome est nullifiable).

Grammaires de Type 0

- Pas de restrictions sur les règles.
- Forme des règles :
  - $\alpha \longrightarrow \beta$
  - $\alpha \in (V \cup T)^+$, $\beta \in (V \cup T)^*$.

$$G_{T_0} \supset G_{T_1} \supset G_{T_2} \supset G_{T_3}$$

Exemple (type 0)

$T = \{a, b, c\}$ Variables : $V = \{A, B, C, S\}$.
Axiome S.
Règles de production :

$S \longrightarrow ABC \mid ASBC$

$CB \longrightarrow BC$

$A \longrightarrow a$

$aB \longrightarrow ab$

$bB \longrightarrow bb$

$bC \longrightarrow bc$

$cC \longrightarrow cc$

Quelle est la forme des mots du langage ?

Mots

$S \longrightarrow ABC \mid ASBC\,;\, CB \longrightarrow BC\,;$
$A \longrightarrow a\,;\, aB \longrightarrow ab\,;\, bB \longrightarrow bb\,;\, bC \longrightarrow bc\,;\, cC \longrightarrow cc$

$S \Longrightarrow ABC \Longrightarrow aBC \Longrightarrow abC \Longrightarrow abc$

$S \Longrightarrow ASBC \Longrightarrow aSBC \Longrightarrow aABCBC \Longrightarrow aaBCBC$
$\Longrightarrow aabCBC \Longrightarrow aabcBC$: impasse

$S \Longrightarrow ASBC \Longrightarrow aSBC \Longrightarrow aABCBC \Longrightarrow aaBCBC$
$\Longrightarrow aaBBCC \Longrightarrow aabBCC \Longrightarrow aabbCC \Longrightarrow aabbcC \Longrightarrow aabbcc$

Forme

$S \longrightarrow ABC \mid ASBC\,;\; CB \longrightarrow BC\,;$
$A \longrightarrow a\,;\; aB \longrightarrow ab\,;\; bB \longrightarrow bb\,;\; bC \longrightarrow bc\,;\; cC \longrightarrow cc$

Forme $a^n b^n c^n$?

Comment obtenir : $aaabbbccc$?

$$S \Longrightarrow ASBC \Longrightarrow aSBC \Longrightarrow aASBCBC \Longrightarrow aaSBCBC$$
$$\Longrightarrow aaABCBCBC \Longrightarrow aaaBCBCBC \Longrightarrow aaaBBCCBC$$
$$\Longrightarrow aaaBBCBCC \Longrightarrow aaaBBBCCC \Longrightarrow aaabBBCCC$$
$$\Longrightarrow aaabbBCCC \Longrightarrow aaabbbCCC \Longrightarrow aaabbbcCC$$
$$\Longrightarrow aaabbbccC \Longrightarrow aaabbbccc$$

Sommaire

ER et grammaires

Pour toute expression régulière R, il existe une grammaire G telle que L(G) = L(R).

Soit R une expression régulière sur un alphabet Σ

- Pas d'opérateurs (n = 0)
  R = $\emptyset$ ou R = ϵ ou R = x avec $x \in \Sigma$
- On crée un symbole S et une grammaire $G = (T, V, S, P)$:
  T = Σ, V = {S}
- Règles
  - R = $\emptyset$, pas de règle, L(G) = $\emptyset$
  - R = ϵ, 1 règle, $S \longrightarrow \epsilon$, L(G) = {$\epsilon$}
  - R = x, 1 règle, $S \longrightarrow x$, L(G) = {x}

Récurrence

On suppose l'hypothèse de récurrence vraie pour des expressions comportant au plus n occurrences d'opérateurs.

Soit R une expression ayant n+1 occurrences d'opérateurs.

$R = R_1|R_2$ ou $R = R_1R_2$ ou $R = R_1*$

R_1 et R_2 possèdent au plus n occurrences d'opérateurs.

Il existe $G_1 = (T, V_1, S_1, P_1)$ et $G_2 = (T, V_2, S_2, P_2)$ telles que $L(R_1) = L(G_1)$ et $L(R_2) = L(G_2)$

Disjonction

- $R = R_1 | R_2$
  - P' = $\{S \longrightarrow S_1 | S_2\}$
  - $G = (T, V, S, P)$
    - $V = V_1 \cup V_2 \cup \{S\}$
    - $P = P_1 \cup P_2 \cup P'$
    - $L(G) = L(R_1) \cup L(R_2)$

Concaténation

- $R = R_1 R_2$
  - P' = $\{S \longrightarrow S_1 S_2\}$
  - $G = (T, V, S, P)$
    - $V = V_1 \cup V_2 \cup \{S\}$
    - $P = P_1 \cup P_2 \cup P'$
    - $L(G) = L(R_1)L(R_2)$

Fermeture

- $R = R_1*$
  - P' = $\{S \longrightarrow S_1 S | \epsilon\}$
  - $G = (T, V, S, P)$
    - $V = V_1 \cup \{S\}$
    - $P = P_1 \cup P'$
    - $L(G) = L(R_1)*$

Exemples

- $R = (a|b)*$
  - $S \longrightarrow aS|bS|\epsilon$
- $R = (a|b) * abb(a|b)*$
  - $S \longrightarrow TabbT\,;\, T \longrightarrow aT|bT|\epsilon$
  - $S \longrightarrow aS|bS|T\,;\, T \longrightarrow abbU\,;\, U \longrightarrow aU|bU|\epsilon$