Théorie des Langages

Analyse syntaxique descendante

Claude Moulin

Université de Technologie de Compiègne

Printemps 2013

Sommaire

Exemple : Grammaire

$T = \{+, -, \times, /, (,), nb\}$; $V = \{E, E', T, T', F\}$; Axiome : E ;
$P = \{$

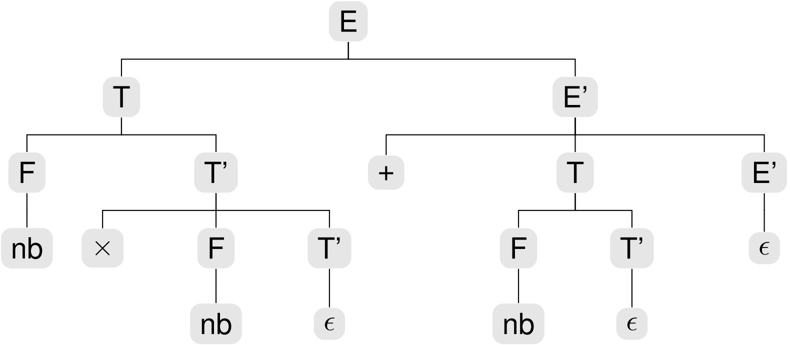
1. $E \longrightarrow TE'$
2. $E' \longrightarrow +TE'$
3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$
5. $T \longrightarrow FT'$
6. $T' \longrightarrow \times FT'$
7. $T' \longrightarrow /FT'$
8. $T' \longrightarrow \epsilon$
9. $F \longrightarrow (E)$
10. $F \longrightarrow nb$

$\}$

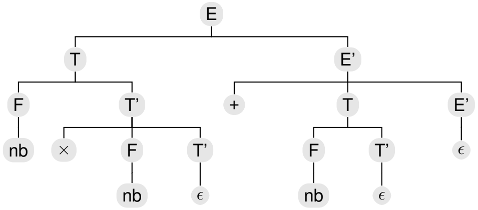
- Cette grammaire permet d'écrire des expressions telles que : (nb + nb) $\times$ (nb - nb) / nb

Exemple : arbre

- Expression : nb $\times$ nb + nb



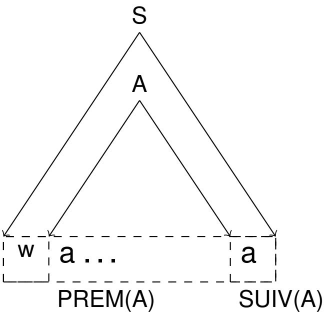
Principe



Expression : nb $\times$ nb + nb

E	nb	E $\longrightarrow$ T E'
T	nb	T $\longrightarrow$ F T'
F	nb	F $\longrightarrow$ nb
T'	$\times$	T' $\longrightarrow$ $\times$ F T'
F	nb	F $\longrightarrow$ nb
T'	+	T' $\longrightarrow$ ϵ
E'	+	E' $\longrightarrow$ + T E'
T	nb	T $\longrightarrow$ F T'
F	nb	F $\longrightarrow$ nb
T'	\$	T' $\longrightarrow$ ϵ
E'	\$	E' $\longrightarrow$ ϵ

Probléme



Le symbole à dériver est A.
Le prochain caractère à lire est a.
La chaîne à analyser est de la forme : wav. w est déjà lue.
Il faut déterminer la règle dont la partie gauche est A à utiliser à ce moment de la dérivation.

$A \longrightarrow X_1 X_2 \ldots X_n$
Les premiers symboles que peux générer A contiennent ceux que peut générer X_1
Si X_1 est nullifiable, c'est X_2 qu'il faut considérer.

Analyse des règles

- Si X_1 ne génère pas ϵ
  - $wA\beta \Rightarrow wX_1X_2\dots X_n\beta \overset{*}{\Rightarrow} wa\gamma_1X_2\dots X_n\beta$.
- Si X_1 génère ϵ et X_2 ne génère pas ϵ
  - $wA\beta \Rightarrow wX_1X_2\dots X_n\beta \Rightarrow wX_2\dots X_n\beta \overset{*}{\Rightarrow} wa\gamma_2X_3\dots X_n\beta$.
- Si $A \overset{*}{\Rightarrow} \epsilon$, $wA\beta \overset{*}{\Rightarrow} w\beta$.
- Le caractère a qui doit décider de la règle à appliquer est :
  - soit dans les premiers caractères que A génère
  - soit dans ses possibles suivants (β).

Sommaire

Premiers

- Les premiers se calculent sur les variables d'une grammaire ainsi que sur les choix ou fin de choix.
- Un choix est une chaîne d'éléments de $V \cup T$ figurant dans la partie droite d'une règle de production.
- Une fin de choix α est une chaîne d'au moins un élément de $V \cup T$ telle qu'il existe $X \in V \cup T$ tel que $X\alpha$ soit un choix ou une fin de choix.

Définition des Premiers

- Soit $X \in V \cup T$.

Définition

PREM(X) est l'ensemble des terminaux (y compris ϵ) qui peuvent commencer une chaîne dérivée de X.

  - $a \in$ PREM(X), si $X \stackrel{*}{\Longrightarrow} a\beta$.

- Soit α un choix ou une fin de choix.

Définition

PREM(α) est l'ensemble des terminaux (y compris ϵ) qui peuvent commencer une chaîne dérivée de α.

  - $a \in$ PREM(α), si $\alpha \stackrel{*}{\Longrightarrow} a\beta$

Initialisation des ensembles

- Cas triviaux :
  - PREM(ϵ) = ϵ.
  - $\forall\ a \in T$, PREM(a) = $\{a\}$.
- Initialisation des autres cas :
  - $\forall\ A \in V$, PREM(A) = $\emptyset$.
  - Pour tout choix ou fin de choix α non vide, PREM(α) = $\emptyset$.

Algorithme

- Pour chaque règle de production $A \longrightarrow \alpha$, ajouter les éléments de PREM(α) à ceux de PREM(A), y compris ϵ.
- Pour chaque choix ou fin de choix α de la forme $X\beta$, ajouter les éléments de PREM(X) à ceux de PREM(α), sauf ϵ.
  - Lorsque PREM(X) contient ϵ, ajouter les éléments de PREM(β) à ceux de PREM(α) y compris ϵ.
- Répéter les phases précédentes tant que l'un des PREM(α) est modifié.

Exemple

$T = \{+, -, *, /, (,), nb\}$; $V = \{E, E', T, T', F\}$; Axiome : E ;
$P = \{$

1. $E \longrightarrow TE'$
2. $E' \longrightarrow +TE'$
3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$
5. $T \longrightarrow FT'$
6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$
8. $T' \longrightarrow \epsilon$
9. $F \longrightarrow (E)$
10. $F \longrightarrow nb$

$\}$

Choix particuliers

+TE'	$\{+\}$
-TE'	$\{-\}$
*FT'	$\{*\}$
/FT'	$\{/\}$
(E)	$\{(\}$

- L'ensemble des premiers d'un choix commençant par un terminal ne contient que ce terminal.

Construction

1. $E \longrightarrow TE'$
2. $E' \longrightarrow +TE'$
3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$
5. $T \longrightarrow FT'$
6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$
8. $T' \longrightarrow \epsilon$
9. $F \longrightarrow (E)$
10. $F \longrightarrow nb$

E	∅				
TE’	∅				
E’	∅				
T	∅				
FT’	∅				
T’	∅				
F	∅				

Construction

1. $E \longrightarrow TE'$
2. $E' \longrightarrow +TE'$
3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$
5. $T \longrightarrow FT'$
6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$
8. $T' \longrightarrow \epsilon$
9. $F \longrightarrow (E)$
10. $F \longrightarrow nb$

E	$\emptyset$				
TE'	$\emptyset$				
E'	$\emptyset$	$\{+, -, \epsilon\}$			règles 2, 3, 4
T	$\emptyset$				
FT'	$\emptyset$	$\{(, nb\}$			à partir de F
T'	$\emptyset$	$\{*, /, \epsilon\}$			règles 6, 7, 8
F	$\emptyset$	$\{(, nb\}$			règles 9, 10

Construction

1. $E \longrightarrow TE'$ 2. $E' \longrightarrow +TE'$ 3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$ 5. $T \longrightarrow FT'$ 6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$ 8. $T' \longrightarrow \epsilon$ 9. $F \longrightarrow (E)$ 10. $F \longrightarrow nb$

E	$\emptyset$				
TE'	$\emptyset$		$\{(, nb\}$		à partir de T
E'	$\emptyset$	$\{+, -, \epsilon\}$			
T	$\emptyset$		$\{(, nb\}$		règle 5
FT'	$\emptyset$	$\{(, nb\}$			
T'	$\emptyset$	$\{*, /, \epsilon\}$			
F	$\emptyset$	$\{(, nb\}$			

Construction

1. $E \longrightarrow TE'$ 2. $E' \longrightarrow +TE'$ 3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$ 5. $T \longrightarrow FT'$ 6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$ 8. $T' \longrightarrow \epsilon$ 9. $F \longrightarrow (E)$ 10. $F \longrightarrow nb$

E	$\emptyset$			$\{(, nb\}$	règle 1
TE'	$\emptyset$		$\{(, nb\}$		
E'	$\emptyset$	$\{+, -, \epsilon\}$			
T	$\emptyset$		$\{(, nb\}$		
FT'	$\emptyset$	$\{(, nb\}$			
T'	$\emptyset$	$\{*, /, \epsilon\}$			
F	$\emptyset$	$\{(, nb\}$			

Construction

1. $E \longrightarrow TE'$
2. $E' \longrightarrow +TE'$
3. $E' \longrightarrow -TE'$
4. $E' \longrightarrow \epsilon$
5. $T \longrightarrow FT'$
6. $T' \longrightarrow *FT'$
7. $T' \longrightarrow /FT'$
8. $T' \longrightarrow \epsilon$
9. $F \longrightarrow (E)$
10. $F \longrightarrow nb$

E	$\emptyset$			$\{(, nb\}$	$\{(, nb\}$
TE'	$\emptyset$		$\{(, nb\}$		$\{(, nb\}$
E'	$\emptyset$	$\{+, -, \epsilon\}$			$\{+, -, \epsilon\}$
T	$\emptyset$		$\{(, nb\}$		$\{(, nb\}$
FT'	$\emptyset$	$\{(, nb\}$			$\{(, nb\}$
T'	$\emptyset$	$\{*, /, \epsilon\}$			$\{*, /, \epsilon\}$
F	$\emptyset$	$\{(, nb\}$			$\{(, nb\}$

Sommaire

Suivants

- Les suivants sont définis pour tous les non terminaux d'une grammaire.
- $X \in V$.

Définition

SUIV(X) est l'ensemble des terminaux qui peuvent suivre immédiatement X lors d'une dérivation.

- $a \in$ SUIV(X) s'il existe un choix contenant Xa.
- Si un choix contient XYa et si Y est nullifiable alors a est un suivant de X.

Algorithme

- Pour toute variable A, SUIV(A) = $\emptyset$.
- Ajouter un marqueur de fin de chaîne à l'axiome ($ par exemple).
- Pour chaque règle $A \longrightarrow \alpha B \beta$, ajouter les éléments de PREM(β) à ceux de SUIV(B), sauf ϵ le cas échéant.
  - lorsque $\epsilon \in$ PREM(β), ajouter les éléments de SUIV(A) aux éléments de SUIV(B).
- Pour chaque règle $A \longrightarrow \alpha B$, ajouter les éléments de SUIV(A) aux éléments de SUIV(B).
- Répéter les 2 phases précédentes tant que l'un des SUIV(A) est modifié.

Exemple

$E \longrightarrow TE'$ $E' \longrightarrow \epsilon$ $T' \longrightarrow /FT'$	$E' \longrightarrow +TE'$ $T \longrightarrow FT'$ $T' \longrightarrow \epsilon$	$E' \longrightarrow -TE'$ $T' \longrightarrow *FT'$ $F \longrightarrow (E)$	$F \longrightarrow nb$

E	$\{(, nb\}$	FT'	$\{(, nb\}$	TE'	$\{(, nb\}$	T'	$\{*, /, \epsilon\}$
E'	$\{+, -, \epsilon\}$	F	$\{(, nb\}$	T	$\{(, nb\}$		

	Init	1	2	SUIV(A)
E	$\emptyset$	$\{\$,)\}$		$\{\$,)\}$
E'	$\emptyset$	$\{\$,)\}$		$\{\$,)\}$
T	$\emptyset$	$\{+, -, \$\}$	$\{+, -, \$,)\}$	$\{+, -, \$,)\}$
T'	$\emptyset$	$\{+, -, \$\}$	$\{+, -, \$,)\}$	$\{+, -, \$,)\}$
F	$\emptyset$	$\{*, /, +, -, \$\}$	$\{*, /, +, -, \$,)\}$	$\{*, /, +, -, \$,)\}$

Sommaire

Table d'analyse

Une table d'analyse est un tableau M à deux dimensions qui indique, pour chaque variable A et chaque terminal a (ainsi que \$), la règle de production à appliquer.

Algorithme de construction :

Pour chaque production A $\longrightarrow \alpha$
  pour tout $a \in$ PREM(α) et a $\neq \epsilon$
    ajouter $A \longrightarrow \alpha$ dans la case M[A, a]
  si $\epsilon \in$ PREM(α), alors
    pour chaque $b \in$ SUIV(A)
      ajouter $A \longrightarrow \alpha$ dans la case M[A, b]

Exemple

$1 : E \longrightarrow TE'$ $4 : E' \longrightarrow \epsilon$ $7 : T' \longrightarrow /FT'$	$2 : E' \longrightarrow +TE'$ $5 : T \longrightarrow FT'$ $8 : T' \longrightarrow \epsilon$	$3 : E' \longrightarrow -TE'$ $6 : T' \longrightarrow *FT'$ $9 : F \longrightarrow (E)$	$10 : F \longrightarrow nb$

E	$\{(, nb\}$	FT'	$\{(, nb\}$	TE'	$\{(, nb\}$	T'	$\{*, /, \epsilon\}$
E'	$\{+, -, \epsilon\}$	F	$\{(, nb\}$	T	$\{(, nb\}$		

	SUIV(A)
E'	$\{\$,)\}$
T'	$\{+, -, \$,)\}$

	+	-	*	/	(	)	nb	\$
E					1		1	
E'	2	3				4		4
T					5		5	
T'	8	8	6	7		8		8
F					9		10	

Automate

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

Q	=	$\{q\}$	l'ensemble des états.
Σ	=	T	l'ensemble des symboles d'entrée.
Γ	=	$T \cup V$	l'alphabet de pile.
δ			la fonction de transition.
q_0	=	q	l'état initial.
Z_0	=	\$	le symbole de fond de la pile.
			on empile aussi le symbole \$
F	=	$\emptyset$	acceptation par pile vide.

Transition

Match : $(q, ax, a\gamma, y) \vdash (q, x, \gamma, y)$

Le symbole d'entrée est le symbole de sommet de la pile.
Il est consommé et le sommet de la pile est dépilé.

Production : $(q, bx, A\gamma, y) \vdash (q, bx, \alpha\gamma, yy_i)$

- Le sommet de la pile A. Règle : $A \longrightarrow \alpha$ (M[A, b]).
- A est remplacé au sommet de la pile par le choix de la règle, en empilant à partir des symboles les plus à droite.
- En sortie le rang y_i, de la règle utilisée, est ajouté.

Acceptation : $(q, \$, \$, y)$

Analyse de : 3 + 4 * 5

	+	-	*	/	(	)	nb	$
E					1		1	
E'	2	3				4		4
T					5		5	
T'	8	8	6	7		8		8
F					9		10	

Entrée	Pile	Transition	Sortie
3 + 4 * 5$	E$	Production	1
3 + 4 * 5$	TE'$	Production	1, 5
3 + 4 * 5$	FT'E'$	Production	1, 5, 10
3 + 4 * 5$	nb T'E'$	Match	1, 5, 10

Analyse de : 3 + 4 * 5

	+	-	*	/	(	)	nb	$
E					1		1	
E'	2	3				4		4
T					5		5	
T'	8	8	6	7		8		8
F					9		10	

Entrée	Pile	Transition	Sortie
+ 4 * 5$	T'E'$	Production	1, 5, 10, 8
+ 4 * 5$	E'$	Production	1, 5, 10, 8, 2
+ 4 * 5$	+TE'$	Match	1, 5, 10, 8, 2

Analyse de : 3 + 4 * 5

	+	-	*	/	(	)	nb	$
E					1		1	
E'	2	3				4		4
T					5		5	
T'	8	8	6	7		8		8
F					9		10	

Entrée	Pile	Transition	Sortie
4 * 5$	TE'$	Production	1, 5, 10, 8, 2, 5
4 * 5$	FT'E'$	Production	1, 5, 10, 8, 2, 5, 10
4 * 5$	nb T'E'$	Match	1, 5, 10, 8, 2, 5, 10
* 5$	T'E'$	Production	1, 5, 10, 8, 2, 5, 10, 6
* 5$	*FT'E'$	Match	1, 5, 10, 8, 2, 5, 10, 6

Analyse de : 3 + 4 * 5

	+	-	*	/	(	)	nb	$
E					1		1	
E'	2	3				4		4
T					5		5	
T'	8	8	6	7		8		8
F					9		10	

Entrée	Pile	Transition	Sortie
5$	FT'E'$	Production	1, 5, 10, 8, 2, 5, 10, 6, 10
5$	nb T'E'$	Match	1, 5, 10, 8, 2, 5, 10, 6, 10
$	T'E'$	Production	1, 5, 10, 8, 2, 5, 10, 6, 10, 8
$	E'$	Production	1, 5, 10, 8, 2, 5, 10, 6, 10, 8, 4
$	$	Accept	1, 5, 10, 8, 2, 5, 10, 6, 10, 8, 4

Sommaire

Grammaire LL(1)

Définition

Une grammaire est dite LL(1) lorsque sa table d'analyse ne comporte pas plusieurs règles de production pour une même cellule.

Une grammaire

- ambiguë,
- récursive à gauche,
- non factorisée à gauche,

n'est pas LL(1).

Récursivité à gauche

Grammaire immédiatement récursive à gauche :

Une grammaire est immédiatement récursive à gauche si elle contient une règle de la forme :

$$A \longrightarrow A\alpha$$

Grammaire récursive à gauche :

Une grammaire est récursive à gauche si elle contient une variable A se dérivant en $A\alpha$ ($A \stackrel{*}{\Longrightarrow} A\alpha$).

Conflit

Une grammaire immédiatement récursive à gauche n'est pas LL(1).
$A \longrightarrow A\alpha$
$A \longrightarrow \beta$
Pour chaque premier(β), il existe au moins deux règles applicables. La grammaire n'est pas LL(1).

Une grammaire récursive à gauche n'est pas LL(1).
$A \stackrel{*}{\Longrightarrow} A\alpha$
$A \longrightarrow \beta$

Une grammaire avec une récursivité à gauche cachée n'est pas LL(1).
$A \stackrel{*}{\Longrightarrow} \alpha A\beta$ et $\alpha \stackrel{*}{\Longrightarrow} \epsilon$

Elimination

- Si $A \longrightarrow A\alpha$ et $A \longrightarrow \beta$
- A produit des chaînes de la forme : $\beta\alpha*$.

Pour éliminer la récursivité à gauche immédiate, il suffit d'ajouter une variable A' et de remplacer les règles telles que :

- $A \longrightarrow A\alpha$
- $A \longrightarrow \beta$

par les règles :

- $A \longrightarrow \beta A'$
- $A' \longrightarrow \alpha A' \mid \epsilon$

Factorisation à gauche

Une grammaire n'est pas factorisée à gauche, si deux productions de la même variable A commencent par le même symbole.

Une grammaire non factorisée à gauche ne peut pas être LL(1).

- $A \longrightarrow X\alpha$ et $A \longrightarrow X\beta$
- sont deux règles applicables pour chaque prem(X).

Algorithme

Pour chaque variable A

Trouver α le plus long préfixe commun à deux ou plusieurs de ses choix

si $\alpha \neq \epsilon$

Remplacer $A \longrightarrow \alpha\beta_1|\ldots|\alpha\beta_n|\gamma_1|\ldots|\gamma_p$

(les γ_i ne commencent pas par α)

par :

$A \longrightarrow \alpha A'|\gamma_1|\ldots|\gamma_p$

$A' \longrightarrow \beta_1|\ldots|\beta_n$

Recommencer tant que deux productions commencent par le même symbole.

Conflit Premier-Premier

1. $S \longrightarrow aAb$
2. $A \longrightarrow cd$
3. $A \longrightarrow ce$

PREM(S) = { a }   SUIV(S) = {$}
PREM(A) = { c }   SUIV(A) = {b}

Table d'analyse

	a	b	c	d	$
S	1				
A			2 ; 3		

Grammaire non-factorisée à gauche. Ex : chaîne acdb.

Autre conflit Premier-Premier

1. $S \longrightarrow aTd$
2. $T \longrightarrow Tb$
3. $T \longrightarrow c$

PREM(S) = { a } SUIV(S) = {$}
PREM(T) = { c } SUIV(T) = {b, d}

Table d'analyse

	a	b	c	d	$
S	1				
T			2 ; 3		

Grammaire récursive à gauche. Ex : chaîne acbd.

Conflit Premier-Suivant

1. $S \longrightarrow aTb$
2. $T \longrightarrow bT$
3. $T \longrightarrow \epsilon$

PREM(S) = { a } SUIV(S) = {$}
PREM(T) = { b, ϵ } SUIV(T) = {b}

Table d'analyse

	a	b	$
S	1		
T		2 ; 3	

T nullifiable et PREM(T) $\cap$ SUIV(T) $\neq \emptyset$. Ex : abb.

Conclusion

Une grammaire est LL(1) lorsque :

1. Il n'y a pas de conflit Premier-Premier. Pour chaque non terminal, les premiers des choix doivent être tous distincts.
2. Il n'y a pas de conflit Premier-Suivant. Pour chaque non terminal ayant un choix nullifiable, les Premier et Suivant doivent être distincts.
3. Il n'y a pas de choix nullifiables multiples.