**Université de Genève**

**Faculté des Sciences Juridiques, Economiques et de Gestion de Genève**

|  |  |
| --- | --- |
| **Module :** Compilation | **Département :** Informatique |
| **Chargé du cours :** DR.9abil | **Classes :** 3LFIAG |
| **Chargé des TDs :** DR.9abil | **AU** : 2013/2014 |

**TD2 : Analyse syntaxique**

**méthode d’analyse descendante**

**Exercice 1:**

1. La grammaire est non ambiguë, en effet pour n'importe quel mot w ∈ L(G) il y a un unique arbre syntaxique, ceci est dû au fait qu'on a établi une certaine règle de priorité et de précédence pour garantir l'unicité de l'arbre syntaxique de chaque mot et par conséquence la non ambiguïté de la grammaire.

Priorité \* est plus associatif à gauche

\*

+

( )

1. Principe de l'algorithme: à partir de G (V, Σ, R, S) on obtient M= (K, Σ, Γ, ∆, s, F):

Les 3 règles du cours:

1. (s, ɛ, ɛ) (p, S)
2. (p, ɛ, A) (p, β) avec A β ∈ R càd si A est au sommet de la pile on peut le remplacer par β.
3. (p, σ, σ) (p, ɛ) ∀ σ ∈ Σ càd si le sommet de la pile coïncide avec le symbole à lire de la chaine, on le

Consomme et on le dépile.

Application des 3 règles sur l'exemple de l'exercice:

|  |  |
| --- | --- |
| 1. 1-(s, ɛ, ɛ) (p, E) 2. 2-(p, ɛ, E) (p, E+T)   3-(p, ɛ, E) (p, T)  4-(p, ɛ, T) (p, T\*F)  5-(p, ɛ, T) (p, F)  6-(p, ɛ, F) (p, (E))  7-(p, ɛ, F) (p, x1)  8-(p, ɛ, F) (p, x2) | 1. 9-(p, +, +) (p, ɛ)   10-(p, (, ( ) (p, ɛ)  11-(p,),)) (p, ɛ)  12-(p, \*, \*) (p, ɛ)  13-(p, x, x) (p, ɛ)  14-(p, 1, 1) (p, ɛ)  15-(p, 2, 2) (p, ɛ) |

K={s, p} ; Γ= V ; s=état initial ; F= {p}

1. le sommet de la pile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Etat | Pile | Restant de la chaine à lire | Transition utilisée | Règle de grammaire |
| S | $ | x1+x2\*x1$ | 1 |  |
| P | $E | x1+x2\*x1$ | 2 | E→ E+T |
| P | $T+E | x1+x2\*x1$ | 3 | E→ T |
| P | $T+T | x1+x2\*x1$ | 5 | T→ F |
| P | $T+F | x1+x2\*x1$ | 7 | F→ x1 |
| P | $T+1x | x1+x2\*x1$ | 13 | Dépiler |
| P | $T+1 | 1+x2\*x1$ | 14 | Dépiler |
| P | $T+ | +x2\*x1$ | 9 | Dépiler |
| P | $T | x2\*x1$ | 4 | T→ T\*F |
| P | $F \* T | x2\*x1$ | 5 | T→ F |
| P | $F \* F | x2\*x1$ | 8 | F→ x2 |
| P | $F \* 2x | x2\*x1$ | 13 | Dépiler |
| P | $F \* 2 | 2\*x1$ | 15 | Dépiler |
| P | $F \* | \*x1$ | 12 | Dépiler |
| P | $F | x1$ | 7 | F→ x1 |
| P | $1x | x1$ | 13 | Dépiler |
| P | $1 | 1$ | 14 | Dépiler |
| P | $ | $ |  |  |

1. M1 est non déterministe ( trouver une configuration ou on peut appliquer plus qu'une transition):

\*A la configuration de l'étape 2 il ya deux transitions applicables (tx 2 et tx3)

\*De même à l'étape 4 (tx 4 et tx 5 applicables).

{

1. E→ E+T sont deux règles récursives à gauche ( A→\*Aα / α un ensemble de terminaux)

T→ T\*F

|  |  |
| --- | --- |
| 1. G'(V', Σ', R', E)   V'={+,\*,(,),x,1,2,ɛ,T,T',E,E',F,A}  Σ'={+,\*,(,),x,1,2,ɛ }  R':   1. E→ TE' élimination de la 2. E'→ +TE' récursivité à gauche 3. E'→ ɛ 4. T→ FT' élimination de la 5. T'→ \*FT' récursivité à gauche 6. T'→ ɛ 7. F→( E) 8. F→ xA 9. A→ 1 factorisation 10. A→ 2 | 1. 1. (s, ɛ, ɛ) (p, E)   2. (p, ɛ, E) (p, TE')  3. (p, ɛ, E') (p, +TE')  4. (p, ɛ, E') (p, ɛ)  5. (p, ɛ, T) (p, FT')  6. (p, ɛ, T') (p,\*FT')  7. (p, ɛ, T') (p, ɛ)  8. (p, ɛ, F) (p, (E))  9. (p, ɛ, F) (p, xA)  10. (p, ɛ, A) (p, 1)  11. (p, ɛ, A) (p, 2)  12. (p, +, +) (p, ɛ)  13. (p, \*, \*) (p, ɛ)  14. (p, ɛ, ɛ) (p, ɛ)  15. (p, (, () (p, ɛ)  16. (p, ), )) (p, ɛ)  17. (p, x, x) (p, ɛ)  18. (p, 1, 1) (p, ɛ)  19. (p, 2, 2) (p, ɛ) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Etat | Pile | Restant de la chaine à lire | Transition utilisée | Règle de grammaire |
| S | $ | x1+x2\*x1$ | 1 |  |
| P | $ E | x1+x2\*x1$ | 2 | E→ TE' |
| P | $E'T | x1+x2\*x1$ | 5 | T→ FT' |
| P | $E'T'F | x1+x2\*x1$ | 9 | F→ xA |
| P | $E'T'Ax | x1+x2\*x1$ | 17 | dépiler |
| P | $E'T'A | 1+x2\*x1$ | 10 | A→ 1 |
| P | $E'T'1 | 1+x2\*x1$ | 18 | dépiler |
| P | $E'T' | +x2\*x1$ | 7 | T'→ ɛ |
| P | $E' | +x2\*x1$ | 3 | E'→ +TE' |
| P | $E'T+ | +x2\*x1$ | 12 | Dépiler |
| P | $E'T | x2\*x1$ | 5 | T→ FT' |
| P | $E'T'F | x2\*x1$ | 9 | F→ xA |
| P | $E'T'Ax | x2\*x1$ | 17 | Dépiler |
| P | $E'T'A | 2\*x1$ | 11 | A→ 2 |
| P | $E'T'2 | 2\*x1$ | 19 | Dépiler |
| P | $E'T' | \*x1$ | 6 | T'→\*FT' |
| P | $E'T'F\* | \*x1$ | 13 | Dépiler |
| P | $E'T'F | x1$ | 9 | F→ xA |
| P | $E'T'Ax | x1$ | 17 | Dépiler |
| P | $E'T'A | 1$ | 10 | A→ 1 |
| P | $E'T'1 | 1$ | 18 | Dépiler |
| P | $E'T' | $ | 7 | T'→ ɛ |
| P | $E' | $ | 4 | E'→ ɛ |
| P | $ | $ |  |  |

**Exercice 3:**

* + S→ Ab| a **|**AA
  + A→ Sa **|** Ac **|** B
  + B→Sd

1. On choisit un ordre : S, A, B : S peut contenir les symboles A, B mais, B ne doit pas contenir ni A, ni S.

Algo : (voir cours) ; on procède par remplacement, puis élimination de la récursivité immédiate pour chaque symbole,

Pour le symbole A :

A→ Sa est transformée en : A→Aba

A→aa

A→AAa

élimination de la récursivité immédiate de A:

A→A**ba** A→aaA’**|**BA’

A→aa A’→baA’**|**AaA’**|**cA’

A→A**Aa** A’→ɛ

A→A**c**

A→B

Pour le symbole B :

Remplacement de S

B→Sd B→Abd Remplacement de A

B→AAd B→aaA’bd**|**B**A’bd**

B→ad B→aaA’Ad**|**B**A’Ad**

B→ad

Elimination de la

recursivité immédiate

de B

B→aaA’bdB’**|**aaA’AdB’**|**adB’

B’→A’bdB’**|**A’AdB’

B’→ ɛ

La grammaire G' après élimination de la récursivité:

S→ Ab **|** a **|**AA

A→aaA’**|**BA’

A’→baA’**|**AaA’**|**cA’

A’→ɛ

B→aaA’bdB’**|**aaA’AdB’**|**adB’

B’→A’bdB’**|**A’AdB’

B’→ ɛ

1. Factorisation :

S→ Ab **|**AA S→ AD

S→ a D → b **|** A

S→ a

A→aaA’**|**BA’

A’→baA’**|**AaA’**|**cA’|ɛ

B→**a**aA’bdB’**|a**aA’AdB’**|a**dB’ B →aE B → aE

E →**a**A’bdB’**|a**A’AdB’**|**dB’ E → aA'F | dB'

F → bdB’|AdB’

B’→**A’**bdB’**|A’**AdB’**|** ɛ B'→A'F **|** ɛ

On obtient:

G' : S→ AD **|** a

D → b **|** A

A→aaA’**|**BA’

A’→baA’**|**AaA’**|**cA’|ɛ

B → aE

E → aA'F **|** dB'

F → bdB’**|**AdB’

B'→A'F **|** ɛ

1. Oui, la grammaire G' est LL(1) car elle n'est pas recursive à gauche.

**Exercice 4:**

1. Si on considère la somme de 2 et 3, on peut écrire sous trois formes:

-notation infixée: 2+3

-notation préfixée: + 2 3

Notation postfixée: 2 3 +

La grammaire donnée définit la notation postfixée des expressions arithmétiques.

1. **E** →**E** E op la grammaire est récursive à gauche donc elle n'est pas LL(1).
2. Elimination de la récursivité:

G1:

E → nb E'

E' → E op E'

E' → ɛ

Pas de factorisation nécessaire.

|  |  |
| --- | --- |
| Premier(E)={nb}  Premier(E') ={nb,ɛ} | suivant(E)={op, $}  suivant(E') ={op,$ } |

Table d'analyse:

|  |  |  |  |
| --- | --- | --- | --- |
|  | nb | op | $ |
| E | E → nb E' |  |  |
| E' | E' → E op E' | E' → ɛ | E' → ɛ |

G1 est LL(1):puisque chaque case de la table d'analyse contient une seule règle.

w= **nb nb op nb op**

|  |  |  |
| --- | --- | --- |
| pile | Restant à lire de w | action |
| $ E | nb nb op nb op $ | E → nb E' 1 |
| $ E'nb | nb nb op nb op $ | dépiler |
| $ E' | nb op nb op $ | E' → E op E' 2 |
| $ E' op E | nb op nb op $ | E → nb E' 3 |
| $ E' op E' nb | nb op nb op $ | Dépiler |
| $ E' op E' | op nb op $ | E' → ɛ 4 |
| $ E' op | op nb op $ | dépiler |
| $ E' | nb op $ | E' → E op E' 5 |
| $ E' op E | nb op $ | E → nb E' 6 |
| $ E' op E' nb | nb op $ | dépiler |
| $ E' op E' | op $ | E' → ɛ 7 |
| $ E' op | op $ | dépiler |
| $ E' | $ | E' → ɛ 8 |
| $ | $ | Le mot est accepté |

1. Construction de l'arbre syntaxique du mot w en réalisant la séquence de sortie "action" de l'analyse syntaxique. Selon la table d'analyse on obtient la séquence 1,2,3….8.

E

nb E'

E op E'

nb E' E op E'

ɛ nb E' ɛ

ɛ

**Exercice 2:**

B → B∨B |B∧B| B⇒B| ¬B | (B) | id | vrai | faux

1. Pour montrer que G est ambiguë, il suffit d'exhiber un mot de L(G) ayant deux arbres de dérivation, exemple w= id ∨ ¬id ∧ vrai

B B

B ∧ B

B ∨ B

B ∨ B vrai

Id B ∧ B id ¬ B

¬ B vrai id

id

1. Pour éliminer l'ambiguïté il faut tenir compte de la priorité des opérateurs et de l'associativité à gauche:

Priorité descendante: () , ¬, ∧, ∨, ⇒

Soit G' (V', Σ' , R', B) tel que :

V={B, E, T, R, id, vrai, faux}

R': B **→** B⇒E | E

E **→** E∨T | T

T **→** T∧R | R

R **→** ¬R | (B) | id | vrai | faux

Rq:

Pour obtenir une grammaire qui n'est pas ambiguë, on doit effectuer des choix de priorités et d'associativités.

Choix de priorité: () est plus prioritaire que ¬, ¬ est plus prioritaire que ∧,etc…

Choix de l'associativité: L'opérateur ∧ est associatif à gauche : *a* ∧ *b* ∧ *c = (a* ∧ *b)* ∧ *c*. On décide que les opérateurs ∨ et ¬ sont également associatifs à gauche (on pourrait les choisir associatifs à droite).