Trad 1

TD 1

TD 1 :

yylength : longueur du lexème reconnu
yytext : lexème.

Exercice 1 :

Grammaire proposée :

$S \to A \mid L$
$N \to idf \mid nul$
$A \to (A.A) \mid N \mid L$
$L \to (L')$
$L' \to L', L' \mid A$

Grammaire utilisée pour la suite :

$A \to V \mid (A.A) \mid (AS)$
$S \to ,AS \mid \wedge$
$V \to idf \mid nul$

Analyse LL(1).

① Calculer les SD
② Remplir la matrice LL(1).
i.e : si $t \in SD(A \to w)$, Alors placer :
$M[A,t] := A \to w$.

Rappel/Attaque : si pour $A \to w_1$, $A \to w_2$,
$SD(A \to w_1) \cap SD(A \to w_2) \neq \emptyset$
Alors on STOP. Grammaire NON LL(1).

terminaux :
$T = \{(,), idf, nul, ., ,, \$\}$

réécriture : $X \to cY$
$aXb \to acYb$

dérivation : $\{réécriture\}^+$

Symbole directeur, premier, suivant.
$SD(A \to w) = Premier(w\ Suivant(A))$
$Premier(abC) = \{a\}$
$Premier(Acd) = Premier(A)$
si $A \overset{*}{\to} \wedge$, $= Premier(A) \cup Premier(cd)$.
$Premier(NUT)^+ \to T$
Suivant : $N \to 2^T$.

$X_1 \to S_1 a \mid S_1 b$ $\quad X_2 \to S_2 a \mid S_2 b$
$S_1 \to dB_1$ $\quad S_2 \to dB_2 C$
$B_1 \to a$ $\quad B_2 \to a$
$C \to y \mid \varepsilon$
$Suivant(S) = \{a, b, \$\}$
$Suivant(B) = Suivant(S) = \{a, b, \$\}$
$Suivant(C) = \{a, b, \$\}$
$Suivant(B_2) = \{y, a, b, \$\}$ car $\{y, \$\} = Premier(C)$
et car $\{a, b, \$\} = Suivant(S_2)$ et $C \overset{*}{\to} \varepsilon$.

① les SD :

SD(V → idf) = idf
SD(V → nil) = nil
SD(A → V) = {idf, nil}
SD(A → (A.A)) = {(}
SD(A → (AS)) = {(}
) problème !

SD(S → , AS) = {,}

Donc on factorise la grammaire :

A → V | (AA'
A' → .A) | S)
S → , AS | ^
V → idf | nil

SD(A' → .A)) = {.}
SD(A' → S)) = {,, ;)}
SD(A → (AA') = {(}
SD(S → ^) = {)}

idem pour les autres.

cible

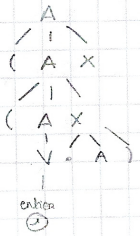
A $
(AX $
A X $
(AXX$
AXX $
VXX$
|
ʌ

XX $
.A)X$

texte Source

((ʌ-2),3) $
(ʌ-2),3) $
ʌ-2),3) $

.2),3) $



ASD

Fonctions Réursives

A chaq non-terminal, on associe une fonction qui vérifie q le texte vérifie la syntaxe décrite par les ≠ alternatives définissant ce non-terminal.

Pour choisir entre les ≠ alternatives, on dispose d'une unité lexicale d'avance. On avancera ds le texte chaq x qu'une unité-lex. est reconnue pour q ttes les fonctions puissent bénéficier de cette u.lex d'avance.

Application sur la grammaire LL(1), non récursive gauche, précédente

4 non-terminaux => 4 fonctions :

fonction analyse_A

" analyse_X

" analyse_S

" analyse V

- la fonction code_unite() : entier délivre le code de l'unité lex lue /courante
- la procédure lire() avance ds le texte et construit la prochaine u.lex
- On suppose q ttes les u.lex sont codées par un entier ;

code_(

code_)

code_...

⋮

TD1

(4)

V -> entier | nil

```
fonction analyse_V
    si (code_unité() = code_entier ou code_unité() = code_nil)
        alors lire()
              // construction du prochain code
        sinon ERREUR ← VRAI
    fsi
```

X -> .A) | S)

```
fonction analyse_X
    si (code_unite() = code_.)
        alors lire()
              analyse_A()
              si (!ERREUR et code_unité() = code_))
                  alors lire()
              sinon ERREUR ← VRAI
        sinon analyse_S()
              si (!ERREUR et code_unité() = code_))
                  alors lire()
              sinon ERREUR ← VRAI
```

S -> , A S | ^

```
fonction analyse_S
    si (code_unité() = code_,)
        alors lire()
              analyse_A()
              si (non ERREUR)
                  alors analyse_S()
```

A -> V | (A X

```
fonction analyse_A
    si (code_unite = code_()
        alors lire(),
              analyse_A(),
              analyse_X(),
        sinon analyse_V()
    fsi
```

Programme principal

Fonction main ().

lire ()

analyse_A ()

si (! erreur)

alors si (code_write(1) = code_$)

alors écrire ("Analyse OK !")

sinon écrire ("Erreur !! ")

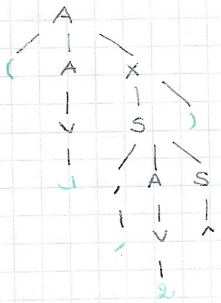
Fonctions sémantiq en ASD

X -> $\wp_1 \alpha \wp_2 A \wp_3 b \wp_4 B \wp_5 c$

---

3) Application sur la grammaire

(1,2)



A -> v | ($\wp_1$ A X

X -> . $\wp_4$ A) $\wp_6$ | S) $\wp_6$

S -> , $\wp_3$ A S | ^ $\wp_7$

V -> entier $\wp_2$ | nil $\wp_5$

$\wp_1$: écrire ("(")

$\wp_2$: écrire ("entier")

$\wp_3$: écrire (". (")

$\wp_4$: écrire (". ")

$\wp_5$: écrire ("nil")

$\wp_6$: écrire (")")

$\wp_7$: écrire (".nil")