Compilation

- Un compilateur permet de facilité la programmation sans utuliser Binaire
- Un compilateur tradut a partir d'un langage haut niveau vers un langage Machine

code source => Compilateur => 011100 001100 code Binaire

↳ Erreur programme

Structure d'un Compilateur

Front end		Middle end		Back end
• Analyse lexical • " syntaxique • " Sémantique	=>	• Generation code intermed • optimisation	=>	• Generation code Machine • optimisation

Analyse lexical :

Construction d'un Analyseur lexical

Expression regulier => Automate finie non deterministe => Automate finie deterministe

Expression regulière :

c'est ensemble Alphabet, utulisé contient

- e = a|b (a ou b)
- e : ab (a et b)
- e = a* (au moins seule a)

(+) [Repitition ou plusieurs fois]

(?) [n'interess pas]

[a-z] [tous les lettres] exemple (ex)

Automate Finie non deterministe

. AFN : $(S, \Sigma, \delta, S_0, F)$

S : un ensemble finie d'etat

Σ : " " Alphabet

δ : Fonction transition

S_0 : etat initial

F : ensemble etat finaux

$S_0 \in S$

$F \subset S$

- Table Transition

Automate finie deterministe

AFD : $(S, \Sigma, \delta, S_0, F)$

S : un ensemble etat

Σ : " Alphabet

δ : fonction transition

S_0 : etat initial

F : ensemble etat finaux

AFN → AFD : ε-fermeteur

Grammaire formel

Def : une grammaire formel est :

$G = \{N, \Sigma, P, S\}$

N : Alphabet non terminaux

Σ : " terminaux

P : un element $\alpha \to \beta$ est appellé règle production

α : est appelée regle gauche règ

β : " " " droite regle

S : est un element non terminal est appellé axiom

Note :

$\alpha \to \beta_1, \alpha \to \beta_2, \dots, \alpha \to \beta_n$

ou les notes :

$\alpha \to \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$

proto mots :

⊙ S est une proto mots

⊙ G ne contenant aucuns symbole non terminal est appellé un mot générer

Derivation

- un regle production $\beta \to \gamma$ est mot $\alpha\beta\gamma$ la derivation est

$\alpha \beta \delta \Rightarrow \alpha \gamma \delta$

exemple :

$L_1 = \{\varepsilon, a, aa, \dots\}$

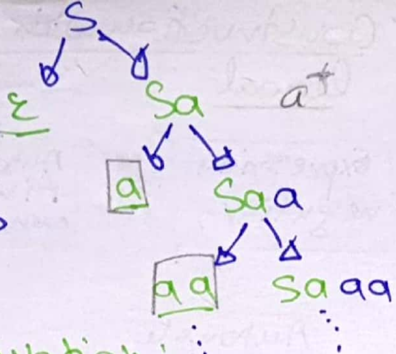
règle

$G = \{\{S\}, \{a\}, \{S \to Sa \mid \varepsilon\}, S\}$

Gramaire formel de L_1

(4)

on Appelle Arbre de derivation ⇒



Sens derivation :

$G = \{\{E, T, F\}, \{+, *, a\}, \{E \to T+E \mid T, T \to T*F \mid F, F \to a\}, E\}$

$E \Rightarrow T + E \Rightarrow T + T \Rightarrow F + T \Rightarrow F + a*T \Rightarrow a + a*F \Rightarrow a + a*a$

⊙ **derivation droite** (deriver droite seul)

$E \Rightarrow T + E \Rightarrow T + T \Rightarrow T + F*T \Rightarrow T + F*F \Rightarrow T + F*a \Rightarrow T + F*a \Rightarrow a + F*a \Rightarrow a + a*a$

⊙ **derivation gauche** (deriver gauche seul)

$E \Rightarrow T + E \Rightarrow F + E \Rightarrow a + E \Rightarrow a + F*T \Rightarrow a + a*T \Rightarrow a + a*F \Rightarrow a + a*a$

Arbre derivation correspondent → 1 seul Arbre deriver gauche → 1 seul Arbre deriver droit

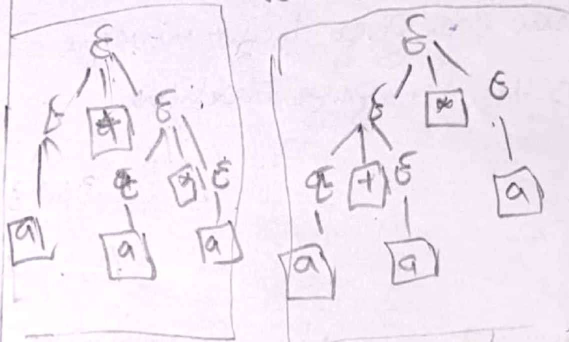
(8)

Ambiguë :

Grammaire Ambiguë si il existe (au moins) deux Arbre derivation

ex : $E \to E + E \mid E * E \mid a$

$w = a + a * a$



Arbre 1 Methode 1 — Arbre 2 Methode 2

type grammaire

- type 0 : sans restriction (si $|\alpha| > 1$)
- type 1 : context ⇓ (si $\alpha A \beta \to \alpha \beta \beta$) quelque soit
- type 2 : hors context : si $(A \to \alpha)$, A ⇒ non terminal, α ⇒ quelque soit

(6)

type 3 : régulière

gauche / droite

si $A \to \alpha B$ (terminal)

si $A \to B\alpha$ (terminal)

① Elimination

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \varepsilon$

non rég

$A \to \beta A'$

$A' \to \alpha A' \mid \varepsilon$

exemple

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid id$

sol

$E \to T E'$

$E' \to + T E' \mid \varepsilon$

$T \to F T'$

$T' \to * T' \mid \varepsilon$

$F \to (E) \mid id$

factorisation

$A \to \alpha\beta_1 \mid \alpha\beta_2$

$A \to \alpha A'$ $(A' \to \beta_1 \mid \beta_2)$

non terminal

exemple:

$S \to iEtS \mid iEtSeS \mid a$

$E \to b$

sol:

$S \to iETSA' \mid a$

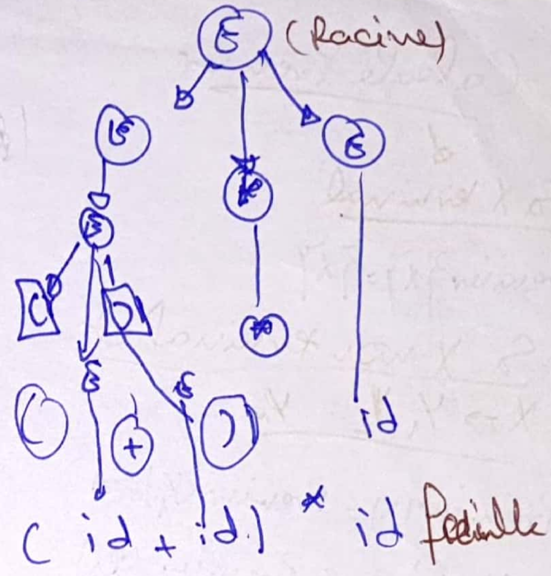
$A' \to eS \mid \varepsilon$

$E \to b$

Analyse syntaxique Descendante

- Commencer a partir non terminal de depart (Racine) pour arriver Alphabet (feuille)
- Remplacer a non terminal A par α $(A \to \alpha)$

exemple



Analyse descendante LL1

① Ensemble Premier

Premier $(\alpha) = \{a \in T \cup \{\varepsilon\} \mid \alpha \overset{+}{\Rightarrow} a\beta\}$

- Ensemble premier contient des terminaux ou ε
- Ensemble des terminaux peuvent apparaître premier

$A \overset{+}{\Rightarrow} aB$

non terminal

Calcule Premier

α

① Si X terminal

Premier{x} = {x}

② Si X non terminal et

$X \to Y_1 Y_2 \ldots Y_m$

Premier(X) = Premier(Y_1) si

(ε ...) $\varepsilon \in$ Premier (Y_1)

②-1

premier (X) = (premier Y_1) ∪ (prem Y_2)

− {ε} si $PY_1 \in \varepsilon$ et $\varepsilon \notin PY_2$

ε-2

Premier X = P(Y_1) ∪ P(Y_2) .... ∪ P(Y_n)

si P(Y_1) ∈ ε et ε ∈ P(Y_2) .... et

$\varepsilon \notin P(Y_m)$

③ ε ∈ P(X)

si $\varepsilon \in P(Y_1)$ et $\varepsilon \in P(Y_2)$ ... et $\varepsilon \in P(Y_m)$

② Ensemble suivant:

Suivant (a) = {a ∈ Premier (γ) | $a \overset{+}{\Rightarrow} \beta a \gamma$}

⊙ Ensemble Suivant contient terminaux peuvent apparaître juste après un symbole dans une suite dérivation

$\alpha \overset{+}{\Rightarrow} \beta \alpha \gamma$

Calcule Suivant

Mettre \$ dans Suivant (S):

. si S est un non-terminal de départ

Mettre Premier (β) − {ε} dans Suivant (B) Suivant Rocane

si il existe une production $A \to \alpha B \beta$ (ε ou la prod)

.Si ε est dans premier β ou β = ε

Mettre Suivant(A) dans suivant (B)

Table d'analyse LL1

⊙ Accéder une case vide

→ Erreur

⊙ Deux dérivation possible

→ Grammaire n'est pas LL1

(il faut une et une seul dérivation)

exemple

exo1 (TD3) : G Grammaire ayant règle production

$S \to SS+ \mid SS^* \mid a$

① Donner Dérivation gauche

$S \Rightarrow SS^*$

$\Rightarrow SS + S^*$

$\Rightarrow aS + S^*$

$\Rightarrow aa + S^*$

$\Rightarrow aa + a^*$

② Donner Dérivation droite

$aa + a^*$

$S \Rightarrow SS^*$

$\Rightarrow S a^*$

$\Rightarrow SS + a^*$

$\Rightarrow S a + a^*$

$\Rightarrow aa + a^*$