Analyse descendante LL(k)

Mirabelle Nebut

Bureau 332 - M3
`mirabelle.nebut at lifl.fr`

2011-2012

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Analyse descendante

L'automate à pile sous-jacent :

- effectue uniquement des lectures et des expansions ;
- construit un arbre en ordre préfixe (idem aut. items) ;
- « part » de l'axiome (idem aut. items) ;
- construit une dérivation gauche (idem aut. items).

Différence avec l'automate des items

Deux différences fondamentales :

- analyse déterministe dite prédictive ;
- plus d'items ni de réductions explicites.

Analyse déterministe

À chaque expansion l'analyseur sait choisir une production.

Il ne revient jamais sur ce choix :

- en cas de succès le mot appartient au langage ;
- en cas d'échec on est sûr que mot n'appartient pas au langage.

Analyse prédictive

L'analyseur "prédit" quelle production utiliser...

... en analysant les k prochains symboles sous la tête de lecture.

Conséquences :

- ne fonctionne qu'avec certaines grammaires, dites LL(k) ;
- tête de lecture toujours définie : marqueur de fin de mot #.

NB : dans ce cours k=1, on regarde la tête de lecture et c'est tout.

Exemple à suivre dans le cours

Soit la grammaire $G = \{V_T, V_N, S, P\}$ avec :

- $V_T = \{a, b, d, e\}$;
- $V_N = \{S, A, B, D\}$;
- P contient les productions :

$$
\begin{aligned}
S &\to AB \mid Da \\
A &\to aAb \mid \epsilon \\
B &\to bB \mid \epsilon \\
D &\to dD \mid e
\end{aligned}
$$

Analyse prédictive LL(1), exemple - 1

$$
\begin{aligned}
S &\rightarrow AB \mid Da \\
A &\rightarrow aAb \mid \epsilon \\
B &\rightarrow bB \mid \epsilon \\
D &\rightarrow dD \mid e
\end{aligned}
$$

$abb\#$?

$S \Rightarrow ?$
$\triangle$

$a \;\; bb\#$
$\triangle$

Choix entre $S \rightarrow AB$ et $S \rightarrow Da$.
Prédiction : expansion par $S \rightarrow AB$

Analyse prédictive LL(1), exemple - 2

$$
\begin{aligned}
S &\to AB \mid Da \\
A &\to aAb \mid \epsilon \\
B &\to bB \mid \epsilon \\
D &\to dD \mid e
\end{aligned}
$$

$abb\#$?

$$
\begin{array}{l}
S \Rightarrow \underset{\triangle}{A}B \Rightarrow ?
\end{array}
$$

$$
\underset{\triangle}{a}\ bb\#
$$

Choix entre $A \to aAb$ et $A \to \epsilon$.
Prédiction : expansion par $A \to aAb$

Analyse prédictive LL(1), exemple - 3

$$
\begin{aligned}
S &\rightarrow AB \mid Da \\
A &\rightarrow aAb \mid \epsilon \\
B &\rightarrow bB \mid \epsilon \\
D &\rightarrow dD \mid e
\end{aligned}
$$

$abb\#$?

$$S \Rightarrow AB \Rightarrow \underset{\triangle}{a}AbB$$

$$\underset{\triangle}{a}\ bb\#$$

Lecture de a.

Analyse prédictive LL(1), exemple - 4

$$\begin{aligned} S &\rightarrow AB \mid Da \\ A &\rightarrow aAb \mid \epsilon \\ B &\rightarrow bB \mid \epsilon \\ D &\rightarrow dD \mid e \end{aligned}$$

$abb\#$?

$$S \Rightarrow AB \Rightarrow a\ \underset{\triangle}{A}bB \Rightarrow ?$$

$$a\ \underset{\triangle}{b}b\#$$

Choix entre $A \rightarrow aAb$ et $A \rightarrow \epsilon$.
Prédiction : expansion par $A \rightarrow \epsilon$

Analyse prédictive LL(1), exemple - 5

$$\begin{aligned} S &\to AB \mid Da \\ A &\to aAb \mid \epsilon \\ B &\to bB \mid \epsilon \\ D &\to dD \mid e \end{aligned}$$

$abb\#$?

$$S \Rightarrow AB \Rightarrow aAbB \Rightarrow abB \Rightarrow abb\ \underset{\triangle}{B} \Rightarrow ?$$

$$ab\underset{\triangle}{b}\ \#$$

Choix entre $B \to bB$ et $B \to \epsilon$.
Prédiction : expansion par $B \to \epsilon$, acceptation.

Se passer des items

Rappel : item de la forme $[X \to \alpha \bullet \beta]$:

- X est en cours de reconnaissance ;
- α a déjà été reconnu ;
- il reste à reconnaître β, le futur de l'item

Un analyseur LL :

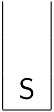
- ne mémorise pas qu'il est en train de reconnaître X ;
- ne mémorise pas qu'il a reconnu α ;
- considère uniquement β.

Se passer des items : conséquences

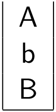
Plus besoin d'axiome supplémentaire.

Dans la pile :

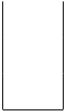
- plus d'items mais des mots étendus : mots de $(V_N \cup V_T)^*$;
- l'alphabet est $V_N \cup V_T$;
- le symbole de pile initiale est l'axiome.



pile initiale



une pile


pile vide (acceptation)

Exemple - les piles pour $abb\#$

$$S \to AB \mid Da \qquad B \to bB \mid \epsilon$$
$$A \to aAb \mid \epsilon \qquad D \to dD \mid e$$

$abb\#$?

		a						
		A	A					
	A	b	b	b		b		
S	B	B	B	B	B	B	B	
$abb\#$	$abb\#$	$abb\#$	$bb\#$	$bb\#$	$b\#$	$b\#$	$\#$	$\#$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

Comparer avec l'automates des items !
Dérivation gauche, arbre en ordre préfixe.

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Deux types de mise en œuvre possibles

Avec pile explicite :

- analyseur dit non-récursif ;
- encodage d'un automate à pile.

Avec pile implicite :

- analyseur = ensemble de fonctions récursives ;
- pile implicite résultant des appels récursifs ;
- on parle d'analyseur récursif : cf TP.

Dans les deux cas : une table d'analyse indique la production à utiliser.

Table d'analyse - exemple

	S	A	B	D
a	$S \rightarrow AB$	$A \rightarrow aAb$	erreur	erreur
b	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow bB$	erreur
d	$S \rightarrow Da$	erreur	erreur	$D \rightarrow dD$
e	$S \rightarrow Da$	erreur	erreur	$D \rightarrow e$
$\#$	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow \epsilon$	erreur

Table d'analyse LL(1)

Contient toute l'intelligence de l'analyseur syntaxique.

Definition

La table d'analyse *Table* est un tableau à deux dimensions tel que :

- chaque colonne est indicée par un non-terminal $\in V_N$;
- chaque ligne est indicée par un terminal $\in V_T$ ou $\#$;
- chaque case contient une production $\in P$ ou *erreur*.

On verra plus tard comment remplir cette table.

Interprétation de $Table[a, X]$

- si le terminal $a \in V_T$ est sous la tête de lecture ;
- et si le non-terminal en cours de traitement est $X \in V_N$;

alors on consulte $Table[a, X]$.

Si $Table[X, a]$ contient

- $X \to \gamma$ alors on choisit une expansion par cette production ;
- *erreur* alors erreur de syntaxe : X et a ne s'accordent pas.

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Analyseur descendant récursif

Principe :

- analyseur LL codé par un ensemble de fonctions ;
- ces fonctions s'appellent les unes les autres ;
- n'utilise pas de pile explicite : pile implicite des appels.

Codage des fonctions :

- une fonction `X()` par non-terminal $X \in V_N$ de la grammaire ;
- `X()` reconnaît un mot engendré par X ;
- la fonction `X()` code les productions $Table[X, y]$ de la table d'analyse, pour tout $y \in V_T \cup \#$.

Exemple

Écrire un analyseur syntaxique récursif LL(1) `Parser` pour G :

$$\begin{aligned} S &\rightarrow AB \mid Da \\ A &\rightarrow aAb \mid \epsilon \\ B &\rightarrow bB \mid \epsilon \\ D &\rightarrow dD \mid e \end{aligned}$$

À voir :

- écriture de `S()`, `A()`, `B()`, `D()` ;
- collaboration avec un analyseur lexical.

Collaboration avec un analyseur lexical

On reprend les conventions utilisées en TP :

- un an. lexical `anLex` de type `Scanner` (supposé donné) ;
- symboles de type `Symbole` ;
- codage entier du type des symboles dans `TypeSymboles` (noté `TS` dans les transparents) ;
- méthode `int getType()` de `Symbole` pour obtenir ce type ;
- méthode `Symbole next_token()` de `Scanner` :
  - avance la tête de lecture `teteLect` ;
  - retourne le symbole lu, de type `Symbole`.
- on remplace le marqueur # par `TS.EOF`.

Construction de l'analyseur syntaxique

```
...
public class Parser {

  // analyseur lexical
  private Scanner anLex;

  // symbole courant reçu de l'analyseur lexical
  private Symbole teteLect;

  public Parser (Scanner anLex) {
     this.anLex = anLex;
  }
...
```

Lancement de l'analyseur syntaxique

Dans la classe Parser :

```
public void analyser() throws ScannerException, ParserExcep

  // positionnement tete de lecture
  this.teteLect = (Symbole) this.anLex.next_token();

  this.S(); // je veux reconnaître l'axiome

  // et uniquement l'axiome
  if (this.teteLect.getType() != TS.EOF)
      throw new ParserException();
}
```

Code de S()

La tête de lecture est déjà positionnée sur le symbole de prédiction.

	S
a	$S \to AB$
b	$S \to AB$
d	$S \to Da$
e	$S \to Da$
$\#$	$S \to AB$

```
private void S() throws ... {
  if (this.teteLect.getType() == TS.a)
  ... // S -> AB
  else if (this.teteLect.getType() == TS.b)
  ... // S -> AB
  else if (this.teteLect.getType() == TS.d)
  ... // S -> Da
  else if (this.teteLect.getType() == TS.e)
  ... // S -> Da
  else if (this.teteLect.getType() == TS.EOF)
  ... // S -> AB
  else throw new ParserException();
}
```

Code de `S()`

On factorise.

	S
a	$S \rightarrow AB$
b	$S \rightarrow AB$
d	$S \rightarrow Da$
e	$S \rightarrow Da$
#	$S \rightarrow AB$

```
private void S() throws ... {
  if (this.teteLect.getType() == TS.a
   || this.teteLect.getType() == TS.b
   || this.teteLect.getType() == TS.EOF)
  ... // S -> AB
  else if (this.teteLect.getType() == TS.d
       || this.teteLect.getType() == TS.e)
  ... // S -> Da
  else throw new ParserException();
}
```

Code des productions de `S()`

Code pour $S \rightarrow AB$:

- je veux reconnaître A puis B ;
- $\Rightarrow$ `A() ; B() ;`

Code pour $S \rightarrow Da$:

- je veux reconnaître D...
- ... puis vérifier que a est bien sous la tête de lecture ;
- et consommer a.

Terminaux : vérification et consommation

Méthode `consommer` dédiée à la gestion des terminaux :

```
private void consommer(int type)
                throws ScannerException, ParserException {
  if (this.teteLect.getType() == type)
    this.teteLect = (Symbole) this.anLex.next_token();
  else throw new ParserException();
}
```

Code pour $S \rightarrow Da$: `D(); this.consommer(TS.a);`.

Code final de `S()`

	S
a	$S \rightarrow AB$
b	$S \rightarrow AB$
d	$S \rightarrow Da$
e	$S \rightarrow Da$
$\#$	$S \rightarrow AB$

```
private void S() throws ... {
  if (this.teteLect.getType() == TS.a
   || this.teteLect.getType() == TS.b
   || this.teteLect.getType() == TS.EOF) {
  A(); B(); // S -> AB
  } else if (this.teteLect.getType() == TS.d
     || this.teteLect.getType() == TS.e) {
  D(); this.consommer(TS.a); // S -> Da
  } else throw new ParserException();
}
```

Quand `S()` termine, pour un mot accepté, la tête de lecture est sur `TS.EOF`.

Code de `A()`

La tête de lecture est déjà positionnée sur le symbole de prédiction.

	A
a	$A \to aAb$
b	$A \to \epsilon$
d	erreur
e	erreur
$\#$	$A \to \epsilon$

```
private void A() throws ... {
  if (this.teteLect.getType() == TS.a)
  ... // A -> aAb
  else if (this.teteLect.getType() == TS.b
        || this.teteLect.getType() == TS.EOF)
  ... // A ->
  else // erreur
     throw new ParserException();
}
```

Code des productions de `A()`

Code pour $A \to aAb$:

```
this.consommer(TS.a); A(); this.consommer(TS.b);
```

Code pour $A \to \epsilon$:

- le mot vide est immédiatement reconnu ;
- sans toucher à la tête de lecture ;
- on ne fait rien.

Code final de `A()`

	A
a	$A \rightarrow aAb$
b	$A \rightarrow \epsilon$
d	erreur
e	erreur
$\#$	$A \rightarrow \epsilon$

```
private void A() throws ... {
  if (this.teteLect.getType() == TS.a) {
  // A -> aAb
  this.consommer(TS.a); A();
  this.consommer(TS.b);
  } else if (this.teteLect.getType() == TS.b
          || this.teteLect.getType() == TS.EOF
    // rien, A ->
  } else // erreur
      throw new ParserException();
}
```

Quand `A()` termine, pour un mot accepté, la tête de lecture est positionnée pour reconnaître un B ou lire un b.

Code final de B()

	B
a	erreur
b	$B \rightarrow bB$
d	erreur
e	erreur
$\#$	$B \rightarrow \epsilon$

```
private void B() throws ... {
  if (this.teteLect.getType() == TS.b) {
  // B -> bB
  this.consommer(TS.b); B();
  } else if (this.teteLect.getType() == TS.EOF
    // rien, B ->
  } else // erreur
      throw new ParserException();
}
```

Code final de D()

	D
a	erreur
b	erreur
d	$D \to dD$
e	$D \to e$
#	erreur

```
private void D() throws ... {
  if (this.teteLect.getType() == TS.d) {
  // D -> dD
  this.consommer(TS.d); D();
  } else if (this.teteLect.getType() == TS.e)
      // D -> e
      this.consommer(TS.e);
  else // erreur
    throw new ParserException();
}
```

Exemple d'exécution

Reconnaître $abb\#$?

Mise en pratique : TP3 (INIT)

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Exemple - les piles pour *abb#*

$$S \rightarrow AB \mid Da$$
$$A \rightarrow aAb \mid \epsilon$$
$$B \rightarrow bB \mid \epsilon$$
$$D \rightarrow dD \mid e$$

abb# ?

		a						
		A	A					
	A	b	b	b		b		
S	B	B	B	B	B	B	B	
abb#	*abb#*	*abb#*	*bb#*	*bb#*	*b#*	*b#*	#	#
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

Comment ça se généralise ?

Principes - 1

- La configuration initiale est $(m\#, S)$;
- La configuration finale est $(\#, \quad)$: acceptation par pile vide.

On traite systématiquement le sommet de pile.

Principes - transition de lecture

Si le sommet de pile est un terminal $a \in V_T$:

- on contrôle que a est bien sous la tête de lecture (sinon échec) ;
- on le consomme ;
- on le dépile.

Lecture de a :

$$(am, z_1 \ldots z_n a) \quad \vdash \quad (m, z_1 \ldots z_n)$$

Principes - transition d'expansion

Si le sommet de pile est un non terminal $X \in V_N$...

... et que la tête de lecture est $y \in V_T \cup \{\#\}$...

si $Table[X, y]$ contient $X \to X_1 \dots X_n$:

- on dépile X ;
- on empile à sa place $X_1 \dots X_n$, avec X_1 au sommet.

$$(m, z_1 \dots z_n X) \quad \vdash \quad (m, z_1 \dots z_n X_n \dots X_1)$$

sinon erreur.

Transition d'expansion : remarque

Expansion par $X \rightarrow X_1 \ldots X_n$:

$$(m, z_1 \ldots z_n X) \quad \vdash \quad (m, z_1 \ldots z_n X_n \ldots X_1)$$

- X_1 sera traité en premier.
- on assure ainsi la construction d'une dérivation gauche ;

Construction de l'arbre syntaxique : ordre préfixe

Transition d'expansion par $X \to X_1 \ldots X_n$:

- X est le « prochain » nœud à traiter dans l'arbre (pour l'ordre préfixe) ;
- on lui rajoute les fils $X_1 \ldots X_n$ de la gauche vers la droite ;
- le prochain nœud à traiter devient X_1.

Transition de a-lecture :

- a est le « prochain » nœud à traiter dans l'arbre (pour l'ordre préfixe) ;
- on vérifie que a concorde bien avec la tête de lecture ;
- et on passe au nœud suivant.

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

- Principes
- Table d'analyse LL
- Analyseur récursif
- Analyseur non-récursif
- Construction de la table d'analyse
  - Ensembles *Premier*
  - Ensemble des ϵ-prod
  - Ensembles *Suivant*
  - Remplissage de la table d'analyse
- Caractérisation d'une grammaire LL(1)
- Quand une grammaire n'est pas LL(1)
  - Factorisation à gauche
  - Suppression de la récursivité à gauche

Outils pour l'analyse prédictive, intuition - 1

Comment choisir entre $S \rightarrow AB$ et $S \rightarrow Da$?

Supposons que je sache que :

- AB ne permet de dériver que des mots préfixés par a ou par b ;
- $AB \Rightarrow^* au$ et $AB \Rightarrow^* bu$, pour $u \in {V_T}^*$;
- Da ne permet de dériver que des mots préfixés par d ou par e ;
- $Da \Rightarrow^* du$ et $Da \Rightarrow^* eu$, pour $u \in {V_T}^*$.

Outils pour l'analyse prédictive, intuition - 1

Maintenant je sais (partiellement) choisir entre $S \rightarrow AB$ et $S \rightarrow Da$:

- si tête lecture $\in \{a, b\}$: choisir $S \rightarrow AB$;
- si tête lecture $\in \{d, e\}$: choisir $S \rightarrow Da$.

	S
a	$S \rightarrow AB$
b	$S \rightarrow AB$
d	$S \rightarrow Da$
e	$S \rightarrow Da$
$\#$	?

Ensemble *Premier* - définition

On dit que $Premier(AB) = \{a, b\}$ et $Premier(Da) = \{d, e\}$.

Pour $\alpha \in (V_T \cup V_N)^+$, $Premier(\alpha)$ contient l'ensemble des terminaux de V_T susceptibles de commencer un mot de ${V_T}^+$ dérivé de α.

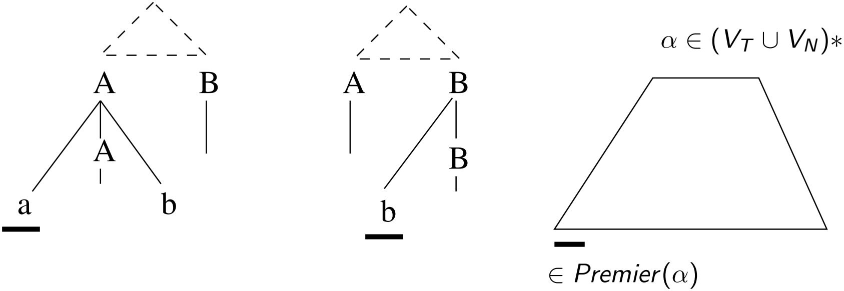
Si $\alpha = \epsilon$, cet ensemble est vide.

Definition

Soit une grammaire algébrique. On définit :

$$\begin{aligned} Premier \;:\; (V_T \cup V_N)^* &\rightarrow \mathcal{P}(V_T) \\ \alpha &\mapsto \{a \in V_T \,|\, \alpha \Rightarrow^* au, u \in {V_T}^*\} \end{aligned}$$

Les *Premier* sur les arbres syntaxiques - notation



Calcul des *Premier* - 0

Soit $\alpha \in (V_N \cup V_T)^*$:

cas

$\alpha = \epsilon$: ?

$\alpha = a$, $a \in V_T$: ?

$\alpha = a\beta$, $a \in V_T$, $\beta \in (V_N \cup V_T)^*$: ?

$\alpha = X$, $X \in V_N$: ?

$\alpha = X\beta$, $X \in V_N$, $\beta \in (V_N \cup V_T)^*$: ?

fcas

Calcul des *Premier* - 1

Soit $\alpha \in (V_N \cup V_T)^*$:

cas

$\alpha = \epsilon : \emptyset$

$\alpha = a,\ a \in V_T : \{a\}$

$\alpha = a\beta,\ a \in V_T,\ \beta \in (V_N \cup V_T)^* : \{a\}$

$\alpha = X,\ X \in V_N :\quad ?$

$\alpha = X\beta,\ X \in V_N,\ \beta \in (V_N \cup V_T)^* :\quad ?$

fcas

Calcul de $Premier(X)$, $X \in V_N$ - exemple

Si l'ensemble des productions de membre gauche S est :

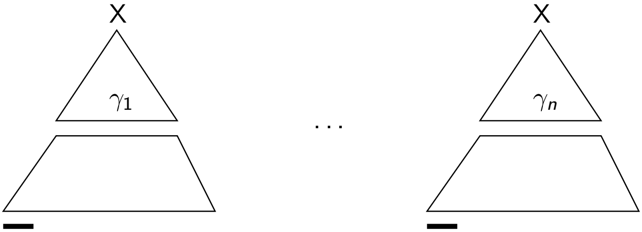
$$S \rightarrow AB \mid Da$$

alors on a :

$$Premier(S) = Premier(AB) \cup Premier(Da)$$

Calcul de $Premier(X)$, $X \in V_N$ - généralisation

Si la grammaire contient les productions de membre gauche X :
$X \to \gamma_1 \mid \ldots \mid \gamma_n$



$\in Premier(\gamma_1)$
$\in Premier(X)$

$\in Premier(\gamma_n)$
$\in Premier(X)$

$$Premier(X) = \bigcup \{Premier(\gamma_i) \mid X \to \gamma_i \in P\}$$

Calcul des *Premier* - 2

Soit $\alpha \in (V_N \cup V_T)^*$:

cas

$$\alpha = \epsilon : \emptyset$$

$$\alpha = a,\ a \in V_T : \{a\}$$

$$\alpha = a\beta,\ a \in V_T,\ \beta \in (V_N \cup V_T)^* : \{a\}$$

$$\alpha = X,\ X \in V_N : \bigcup\{Premier(\gamma_i) \mid X \rightarrow \gamma_i \in P\}$$

$$\alpha = X\beta,\ X \in V_N,\ \beta \in (V_N \cup V_T)^* : \quad ?$$

fcas

Calcul des *Premier*, $\alpha = X\beta$, $X \in V_N$

Deux cas selon que X peut « s'effacer » ou non :

$$X \Rightarrow^* \epsilon\,?$$

Si $X \Rightarrow^* \epsilon$ on dit que X est ϵ-productif : $X \in \epsilon\text{-}Prod$

Calcul des *Premier*, $\alpha = X\beta$, $X \notin \epsilon$-Prod - exemple

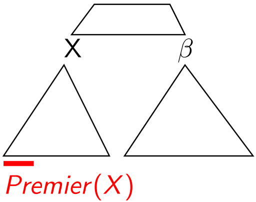
Par exemple $D \notin \epsilon$-productif :

$$D \rightarrow dD \mid e$$

alors $Premier(Da) = Premier(D)$

Calcul des *Premier*, $\alpha = X\beta$, $X \notin \epsilon$-Prod - généralisation

$$X \not\Rightarrow^* \epsilon$$



$$Premier(\alpha) = Premier(X)$$

Calcul des *Premier*, $\alpha = X\beta$, $X \in \epsilon$-Prod - exemple

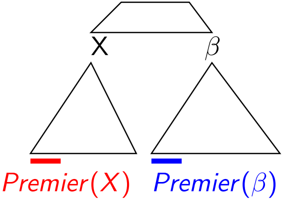
Par exemple $A \in \epsilon$-productif :

$$A \rightarrow \epsilon \mid aAb$$

$$Premier(AB) = Premier(A) \cup Premier(B)$$
$$Premier(ABS) = Premier(A) \cup Premier(BS)$$

Calcul des *Premier*, $\alpha = X\beta$, $X \in V_N$ - généralisation

$$X \Rightarrow^* \epsilon$$



$$Premier(\alpha) =$$
$$Premier(X) \cup Premier(\beta)$$

Calcul des *Premier*

Soit $\alpha \in (V_N \cup V_T)^*$:

cas

$\alpha = \epsilon : \emptyset$

$\alpha = a,\ a \in V_T : \{a\}$

$\alpha = a\beta,\ a \in V_T,\ \beta \in (V_N \cup V_T)^* : \{a\}$

$\alpha = X,\ X \in V_N : \bigcup\{Premier(\gamma_i) \mid X \to \gamma_i \in P\}$

$\alpha = X\beta,\ X \in V_N \setminus \epsilon\text{-}Prod,\ \beta \in (V_N \cup V_T)^* : Premier(X)$

$\alpha = X\beta,\ X \in V_N \cap \epsilon\text{-}Prod,\ \beta \in (V_N \cup V_T)^* : Premier(X) \cup Premier(\beta)$

fcas

Calcul effectif des ensembles *Premier*

On procède en deux étapes :

1. on pose un système d'équations pour *Premier* ;
2. on calcule par itération de point fixe les plus petits ensembles qui satisfont ces équations.

Pour le moment on suppose donné ϵ-*Prod*, l'ensemble des ϵ-productifs.

Exemple

$$S \rightarrow AB \mid Da$$
$$A \rightarrow aAb \mid \epsilon$$
$$B \rightarrow bB \mid \epsilon$$
$$D \rightarrow dD \mid e$$

$$\epsilon\text{-}Prod = \{A, B, S\}$$

$$Premier(S) = ?$$
$$Premier(A) = ?$$
$$Premier(B) = ?$$
$$Premier(D) = ?$$

$\alpha = \epsilon : \emptyset$

$\alpha = a,\ a \in V_T : \{a\}$

$\alpha = a\beta,\ a \in V_T,\ \beta \in (V_N \cup V_T)^* : \{a\}$

$\alpha = X,\ X \in V_N :$
$\bigcup\{Premier(\gamma_i) \mid X \rightarrow \gamma_i \in P\}$

$\alpha = X\beta,\ X \in V_N \setminus \epsilon\text{-}Prod,\ \beta \in (V_N \cup V_T)^* :$
$Premier(X)$

$\alpha = X\beta,\ X \in V_N \cap \epsilon\text{-}Prod,\ \beta \in (V_N \cup V_T)^* :$
$Premier(X) \cup Premier(\beta)$

Exemple

$$Premier(S) = Premier(A) \cup Premier(B) \cup Premier(D)$$
$$Premier(A) = \{a\}$$
$$Premier(B) = \{b\}$$
$$Premier(D) = \{d, e\}$$

D'où

$$Premier(S) = \{a, b, d, e\} \qquad Premier(aAb) = \{a\}$$
$$Premier(A) = \{a\} \qquad Premier(bB) = \{b\}$$
$$Premier(B) = \{b\} \qquad Premier(dD) = \{d\}$$
$$Premier(D) = \{d, e\} \qquad Premier(e) = \{e\}$$
$$Premier(AB) = \{a, b\} \qquad Premier(\epsilon) = \emptyset$$
$$Premier(Da) = \{d, e\}$$

Exemple : remplissage de la table

$A \rightarrow aAb$ et $Premier(aAb) = \{a\}$
$A \rightarrow \epsilon$ et $Premier(\epsilon) = \emptyset$

	S	A	B	D
a	$S \rightarrow AB$	$A \rightarrow aAb$	erreur	erreur
b	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow bB$	erreur
d	$S \rightarrow Da$	erreur	erreur	$D \rightarrow dD$
e	$S \rightarrow Da$	erreur	erreur	$D \rightarrow e$
#	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow \epsilon$	erreur

Exemple : remplissage de la table

$S \rightarrow AB$ et $Premier(AB) = \{a, b\}$
$S \rightarrow Da$ et $Premier(Da) = \{d, e\}$

	S	A	B	D
a	$S \rightarrow AB$	$A \rightarrow aAb$	erreur	erreur
b	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow bB$	erreur
d	$S \rightarrow Da$	erreur	erreur	$D \rightarrow dD$
e	$S \rightarrow Da$	erreur	erreur	$D \rightarrow e$
$\#$	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow \epsilon$	erreur

Remarque : résolution du système

$$
\begin{aligned}
&Premier(S) = Premier(A) \cup Premier(B) \cup Premier(D)\\
&Premier(A) = \{a\}\\
&Premier(B) = \{b\}\\
&Premier(D) = \{d, e\}
\end{aligned}
$$

Se résoud sans itération de point fixe : système d'équations non récursif.

Ce n'est pas toujours le cas.

Résolution du système : autre exemple

$$S \to S_1 S_2 \mid a$$
$$S_1 \to S \mid b$$
$$S_2 \to c$$

$$Premier(S) = Premier(S_1) \cup \{a\}$$
$$Premier(S_1) = Premier(S) \cup \{b\}$$
$$Premier(S_2) = \{c\}$$

iter	$Premier(S)$	$Premier(S_1)$	$Premier(S_2)$
0	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{a\}$	$\{b\}$	$\{c\}$
2	$\{a, b\}$	$\{b, a\}$	$\{c\}$
3	$\{a, b\}$	$\{b, a\}$	$\{c\}$

stabilisation

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse

- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)

- Factorisation à gauche
- Suppression de la récursivité à gauche

Définition des ϵ-prod

Definition

Un non terminal $X \in V_N$ est dit ϵ-productif si $X \Rightarrow^* \epsilon$.

L'ensemble des ϵ-productif est ϵ-*Prod*.

X est ϵ-productif si la grammaire contient la production :

- $X \rightarrow \epsilon$;
- ou $X \rightarrow Y_1 Y_2 \dots Y_n$ telle que l'ensemble des non-terminaux $\{Y_1, Y_2, \dots, Y_n\} \subseteq V_N$ ne contient que des non-terminaux ϵ-productifs.

Algorithme de calcul similaire à celui qui calcule les productifs.

Exemple

$$
\begin{aligned}
S &\rightarrow AB \mid Da \\
A &\rightarrow aAb \mid \epsilon \\
B &\rightarrow bB \mid \epsilon \\
D &\rightarrow dD \mid e
\end{aligned}
$$

ϵ-$Prod$?

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Outils pour l'analyse prédictive, intuition - 2

Pour choisir entre $S \to AB$ et $S \to Da$:

- si tête lecture $\in \{a, b\}$: choisir $S \to AB$;
- si tête lecture $\in \{d, e\}$: choisir $S \to Da$.

Et si la tête de lecture est $\#$? $\# \notin Premier(AB) \cup Premier(Da)$.

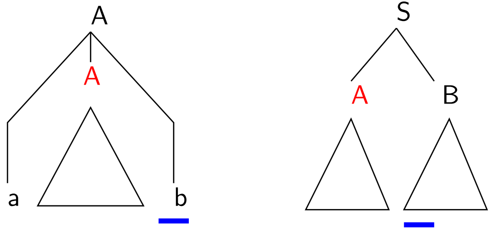
Comment choisir entre $A \to aAb$ et $A \to \epsilon$? $Premier(\epsilon) = \emptyset$

$\Rightarrow$ les ensembles *Premier* ne suffisent pas.

Ensembles *Suivant*, intuition

Quand appliquer $A \rightarrow \epsilon$?

Quand la tête de lect. correspond à un terminal qui peut suivre A.



$b \in Suivant(A)$

$Suivant(A) \supseteq Premier(B)$

Ensembles *Suivant*, intuition

On a donc $b \in Suivant(A)$

	A
a	$A \rightarrow aAb$
b	$A \rightarrow \epsilon$
d	?
e	?
$\#$	?

Ensembles *Suivant*, intuition

Pour $\alpha \in (V_T \cup V_N)^+$, $Suivant(\alpha)$ contient l'ensemble des terminaux de V_T susceptibles de suivre α dans un mot de ${V_T}^+$ dérivé de l'axiome S.

Si $\alpha = \epsilon$, cet ensemble est vide.

Par convention, $Suivant(S) \supseteq \{\#\}$.

Ensembles *Suivant*, définitions équivalentes

Definition

Soit une grammaire algébrique d'axiome S. On définit :

$$\begin{array}{rcl} Suivant : (V_T \cup V_N)^* & \rightarrow & \mathcal{P}(V_T) \\ \alpha & \mapsto & \{a \in V_T \mid S \Rightarrow^* \beta \alpha a \gamma, \\ & & \quad \text{pour } \beta, \gamma \in (V_N \cup V_T)^*\} \end{array}$$

Definition

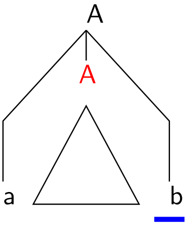
$$\begin{array}{rcl} Suivant : (V_T \cup V_N)^* & \rightarrow & \mathcal{P}(V_T) \\ \alpha & \mapsto & \{a \in Premier(\gamma) \mid S \Rightarrow^* \beta \alpha \gamma, \\ & & \quad \text{pour } \beta, \gamma \in (V_N \cup V_T)^*\} \end{array}$$

Calcul des *Suivant* - 1

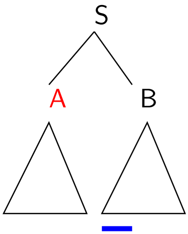
Pour calculer $Suivant(X)$, on regarde les productions dans lesquelles X apparaît en partie droite (différent du calcul des *Premier*).

Pour $Suivant(A)$:

$A \rightarrow aAb$



$S \rightarrow AB$



Calcul des *Suivant* - 2

Soit $P_X \subseteq P$ l'ensemble des productions p dans lesquelles X apparaît en partie droite :

$$Suivant(X) = \bigcup_{p \in P_X} Suivant_p(X)$$

Ex : $P_A = \{S \rightarrow AB, A \rightarrow aAb\}$
$Suivant(A) = Suivant(A)_{S \rightarrow AB} \cup Suivant(A)_{A \rightarrow aAb}$

Calcul des *Suivant*, cas de l'axiome

Pour l'axiome, on ajoute le marqueur de fin de mot :

$$Suivant(S) = \{\#\} \cup \bigcup_{p \in P_S} Suivant_p(S)$$

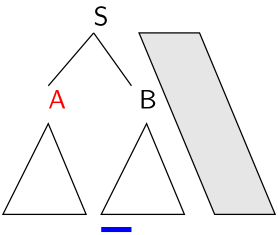
Ex : pour $X \rightarrow aXb \mid \epsilon$

$P_X = \{X \rightarrow aXb\}$
$Suivant(X) = \{\#\} \cup Suivant(X)_{X \rightarrow aXb}$

Calcul des *Suivant* - 3 - exemple

$Suivant(A)_{S \rightarrow AB}$? (exemple précédent)

Cas déjà vu :

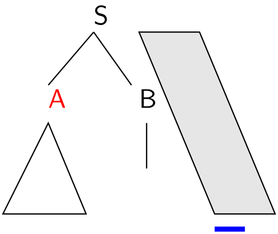


$$Suivant(A) \supseteq Premier(B)$$

Calcul des *Suivant* - 3 - exemple

$Suivant(A)_{S \to AB}$? (exemple précédent)

mais B est ϵ-*Prod* !



$$Suivant(A) \supseteq Suivant(S)$$

Calcul des *Suivant* - 3 - exemple

$$Suivant(A)_{S \rightarrow AB} = Premier(B) \cup Suivant(S)$$

Calcul des *Suivant* - 4

Quand une production est de la forme $\ldots \rightarrow X\alpha$:

- pour calculer $Suivant(X)$;
- Il faut pouvoir dire si $\alpha \in (V_N \cup V_T)^*$ est ϵ-productif ou pas.

Definition

$\alpha \in (V_N \cup V_T)^*$ est ϵ-productif si $\alpha \Rightarrow^* \epsilon$. On définit la fonction :

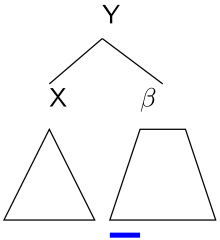
$$\begin{array}{rcl} Eps \;:\; (V_N \cup V_T)^* & \rightarrow & \{vrai, faux\} \\ \alpha & \mapsto & \alpha \text{ est } \epsilon\text{-productif} \end{array}$$

On verra après comment calculer *Eps*.

Calcul des *Suivant* - 5

Pour calculer $Suivant_p(X)$ avec :

$$p = \; Y \rightarrow \alpha X \beta \text{ et } Eps(\beta) = faux, \; \alpha, \beta \in (V_N \cup V_T)^*$$



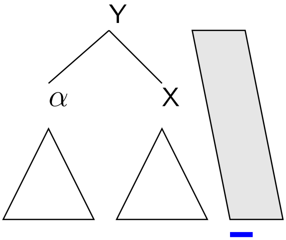
$$Suivant_p(X) = Premier(\beta)$$

Ex : pour $Y \rightarrow Xb$, $Suivant(X)_{Y \rightarrow Xb} = \{b\}$.

Calcul des *Suivant* - 6

Pour calculer $Suivant_p(X)$ avec :

$$p = \; Y \rightarrow \alpha X$$

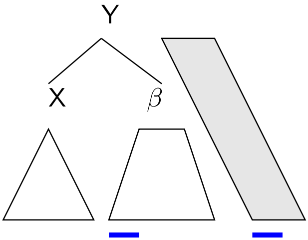


$$Suivant_p(X) = Suivant(Y)$$

Calcul des *Suivant* - 7

Pour calculer $Suivant_p(X)$ avec :

$$p = \; Y \rightarrow \alpha X \beta \text{ et } Eps(\beta) = vrai, \; \alpha, \beta \in (V_N \cup V_T)^*$$



$Suivant_p(X) =$
$Premier(\beta) \cup Suivant(Y)$

Ex : pour $S \rightarrow AB$, $Suivant(A)_{S \rightarrow AB} = Premier(B) \cup Suivant(S)$.

Remarque - 1

Si X apparaît plusieurs fois en partie droite d'une production, il faut prendre en compte toutes ses occurrences dans le calcul de $Suivant(X)$.

Ex : $Y \rightarrow XaXaXc$

$Suivant_{Y \rightarrow XaXaXc}(X) = \{a, c\}$

Remarque - 2

Pourquoi pas $Suivant(X)_{Y \to X\beta} = Premier(\beta) \cup Suivant(\beta)$?

Parce que $Suivant(Y) \subseteq Suivant(\beta)$.

Ex :

$S \to Y \mid Z$	$Suivant(S) = \{\#\}$
$Y \to X_1 X_2$	$Suivant(Y) = Suivant(S) = \{\#\}$
$X_1 \to a$	$Suivant(Z) = Suivant(S) = \{\#\}$
$X_2 \to \epsilon \mid b$	$Suivant(X_2) = \{c\} \cup Suivant(Y) = \{c, \#\}$
$Z \to X_2 c$	$Suivant(X_1) = Premier(X_2) \cup Suivant(Y)$

Calcul des *Suivant*, récapitulons !

$\# \in Suivant(axiome)$

Soit $P_X \subseteq P$ l'ensemble des productions p dans lesquelles X apparaît en partie droite :

$$Suivant(X) = \bigcup_{p \in P_X} Suivant_p(X)$$

avec : $Suivant_p(X) =$

cas

$p = \; Y \rightarrow \alpha X$: $Suivant(Y)$

$p = \; Y \rightarrow \alpha X \beta$ et $Eps(\beta) = faux$: $Premier(\beta)$

$p = \; Y \rightarrow \alpha X \beta$ et $Eps(\beta) = vrai$: $Premier(\beta) \cup Suivant(Y)$

fincas

Calcul effectif des ensembles *Suivant*

On procède en deux étapes :

- on pose un système d'équations pour *Suivant* ;
- on calcule par itération de point fixe les plus petits ensembles qui satisfont ces équations.
- avec initialement $\mathit{Suivant}(S) = \{\#\}$, et pour les autres non-terminaux $\mathit{Suivant}(X) = \emptyset$.

Exemple

$$
\begin{aligned}
S &\rightarrow AB \mid Da \\
A &\rightarrow aAb \mid \epsilon \\
B &\rightarrow bB \mid \epsilon \\
D &\rightarrow dD \mid e
\end{aligned}
$$

$Eps(B) = \text{vrai}$

Suivant ?

$\# \in Suivant(axiome)$

$Suivant(X) = \bigcup_{p \in P_X} Suivant_p(X)$

avec : $Suivant_p(X) =$

cas

$p = \; Y \rightarrow \alpha X \; : \; Suivant(Y)$

$p = \; Y \rightarrow \alpha X \beta$ et $Eps(\beta) = faux \; : \; Premier(\beta)$

$p = \; Y \rightarrow \alpha X \beta$ et $Eps(\beta) = vrai \; : \; Premier(\beta) \cup Suivant(Y)$

fincas

Exemple de remplissage de table

$$A \rightarrow \epsilon \text{ et } Suivant(A) = \{b, \#\}$$

	S	A	B	D
a	$S \rightarrow AB$	$A \rightarrow aAb$	erreur	erreur
b	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow bB$	erreur
d	$S \rightarrow Da$	erreur	erreur	$D \rightarrow dD$
e	$S \rightarrow Da$	erreur	erreur	$D \rightarrow e$
$\#$	$S \rightarrow AB$	$A \rightarrow \epsilon$	$B \rightarrow \epsilon$	erreur

Exemple de remplissage de table

$S \to AB$ et $Suivant(S) = \{\#\}$

	S	A	B	D
a	$S \to AB$	$A \to aAb$	erreur	erreur
b	$S \to AB$	$A \to \epsilon$	$B \to bB$	erreur
d	$S \to Da$	erreur	erreur	$D \to dD$
e	$S \to Da$	erreur	erreur	$D \to e$
$\#$	$S \to AB$	$A \to \epsilon$	$B \to \epsilon$	erreur

Calcul des ϵ-productifs

On connaît déjà ϵ-*Prod*, ens. des non-terminaux ϵ-productifs.

Pour calculer $Eps(\alpha)$:

$Eps(\alpha) =$

cas

$\alpha = \epsilon$: *vrai*

$\alpha = X_1 \dots X_n, n \geq 1$ avec $\{X_1, \dots, X_n\} \subseteq V_N$ et $\{X_1, \dots, X_n\} \subseteq \epsilon\text{-}Prod$: *vrai*

autre : *faux* // α `contient un terminal`

fincas

Exemple

Sachant que $\epsilon\text{-}Prod = \{A, B, S\}$:

$Eps(Da)$?

$Eps(AB)$?

$Eps(\epsilon)$?

$Eps(B)$?

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Table d'analyse : au préalable

On calcule :

- les ϵ-productifs ;
- les ensembles *Premier* ;
- les ensembles *Suivant*.

Remplissage de la table

Entrée : une gram. alg. G, ses ensembles *Premier* et *Suivant*
Sortie : la table d'analyse *Table*

```
pour toute production X → γ ∈ P
faire pour tout a ∈ Premier(γ)
      faire ajouter X → γ à Table[X, a] fait
      si Eps(γ) = vrai alors pour tout b ∈ Suivant(X)
                             faire Table[X, b] = X → γ
                             fait
      finsi
fait
```

Ajouter *erreur* dans les entrées de *Table* restées vides

Exemple

$S \to AB$:

- $Premier(AB) = \{a, b\}$;
- $Eps(AB) = vrai$;
- $Suivant(S) = \{\#\}$.

$S \to Da$:

- $Premier(Da) = \{d, e\}$;
- $Eps(Da) = faux$.

Rien à compléter par *erreur*.

	S
a	$S \to AB$
b	$S \to AB$
d	$S \to Da$
e	$S \to Da$
$\#$	$S \to AB$

Exemple

$A \to aAb$:

- $Premier(aAb) = \{a\}$;
- $Eps(aAb) = faux$.

$A \to \epsilon$:

- $Premier(\epsilon) = \emptyset$;
- $Eps(\epsilon) = vrai$;
- $Suivant(A) = \{b, \#\}$.

On complète par *erreur*.

	A
a	$A \to aAb$
b	$A \to \epsilon$
d	erreur
e	erreur
$\#$	$A \to \epsilon$

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Analyseur LL(1)

Un analyseur LL(1) est déterministe et piloté par le sommet de pile :

- si terminal a : lecture de a (ou erreur) ;
- si non terminal X avec a sous la tête de lecture : expansion selon $Table[X, a]$.

Et si $Table[X, a]$ contient plus d'une production ?

Non-déterminisme :

- la grammaire n'est pas LL(1) ;
- on ne peut pas appliquer une analyse LL(1).

Caractérisation d'une grammaire LL(1)

Caractérisation par table d'analyse : une grammaire est LL(1) si chaque case contient exactement une production ou erreur.

Caractérisation « par contre-exemple » : une grammaire n'est pas LL(1) s'il existe 2 productions $X \rightarrow \alpha$ et $X \rightarrow \beta$ telles que :

1. soit $\mathit{Premier}(\alpha) \cap \mathit{Premier}(\beta) \neq \emptyset$;
   Ex : $S \rightarrow aS \mid A,\ A \rightarrow a$
2. soit $\mathit{Eps}(\alpha) = \mathit{vrai}$ et $\mathit{Premier}(\beta) \cap \mathit{Suivant}(X) \neq \emptyset$;
   Ex : $S \rightarrow aS \mid Ab,\ A \rightarrow \epsilon \mid b$
3. soit $\mathit{Eps}(\alpha) = \mathit{vrai}$ et $\mathit{Eps}(\beta) = \mathit{vrai}$ (la grammaire est ambiguë)
   Ex : $S \rightarrow A \mid B,\ A \rightarrow \epsilon,\ B \rightarrow \epsilon$

LL(1) et ambiguïté

Une grammaire LL(1) n'est pas ambiguë.

Une grammaire ambiguë n'est pas LL(1).

Cas classiques non LL(1)

Dans les cas suivants, la grammaire n'est pas LL(1) :

- ambiguïté ;
- récursivité gauche : $A \to Aa \mid \epsilon$;
  - intuitivement récursivité infinie de `A()`.
- non factorisation gauche : $S \to aA \mid aB$

On peut parfois arranger ça :

- factorisation à gauche ;
- suppression de la récursivité gauche.

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Factorisation à gauche : exemple - 1

Les listes d'identificateur de INIT :

$$\textit{listeIdent} \rightarrow \text{IDENT} \mid \text{IDENT SEPVAR } \textit{listeIdent}$$

On factorise IDENT et on obtient :

$\textit{listeIdent} \rightarrow \text{IDENT } \textit{suiteListeIdent}$
$\textit{suiteListeIdent} \rightarrow \epsilon \mid \textit{SEPVAR listeIdent}$

Factorisation à gauche : exemple - 2

$X \rightarrow ab \mid abbX \mid abbbX$

Factorisation de ab : on prend le plus grand préfixe commun.
$X \rightarrow abY$
$Y \rightarrow \epsilon \mid bX \mid bbX$

Puis à nouveau factorisation de b.

Factorisation à gauche - algorithme

On remplace les règles de la forme :

$$X \to \alpha\beta_1 \mid \ldots \mid \alpha\beta_n \mid \gamma_1 \mid \ldots \mid \gamma_m$$

où

- $\alpha \in (V_T \cup V_N)^+$ et $\beta_i, \gamma_j \in (V_T \cup V_N)*$;
- le préfixe commun α est choisi le plus grand possible ;
- α n'est pas préfixe de γ_j.

par les règles :

$$X \to \alpha X' \mid \gamma_1 \mid \ldots \mid \gamma_m$$
$$X' \to \beta_1 \mid \ldots \mid \beta_n$$

où X' est un nouveau non-terminal.
On réitère ce processus tant que nécessaire.

Principes

Table d'analyse LL

Analyseur récursif

Analyseur non-récursif

Construction de la table d'analyse
- Ensembles *Premier*
- Ensemble des ϵ-prod
- Ensembles *Suivant*
- Remplissage de la table d'analyse

Caractérisation d'une grammaire LL(1)

Quand une grammaire n'est pas LL(1)
- Factorisation à gauche
- Suppression de la récursivité à gauche

Suppression de la récursivité à gauche

Récursivité gauche :

- immédiate : production $A \rightarrow A\alpha$, $\alpha \in (V_T \cup V_N)^+$;
- générale : il existe une dérivation $A \Rightarrow^* A\alpha$, $\alpha \in (V_T \cup V_N)^+$.

Il est possible de supprimer les deux cas.

On ne verra que la récursivité immédiate.

Suppression de la récursivité gauche immédiate

On remplace les règles de la forme

$$X \to X\alpha_1 \mid \ldots \mid X\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$$

où

- $\alpha_i \in (V_T \cup V_N)^+$ et $\beta_j \in (V_T \cup V_N)*$;
- les β_j ne commencent pas par X.

par les règles :

$$X \to \beta_1 X' \mid \ldots \mid \beta_m X'$$
$$X' \to \alpha_1 X' \mid \ldots \mid \alpha_n X' \mid \epsilon$$

où X' est un nouveau non-terminal.

Suppression de la récursivité gauche : exemple

Grammaire non ambiguë
des expressions arithmétiques :

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow \texttt{i} \mid (E)$$

$$X \rightarrow \; X\alpha_1 \mid \ldots \mid X\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$$

Après suppression de la rec gauche :

$$E \rightarrow TE'$$
$$E' \rightarrow +TE' \mid \epsilon$$
$$T \rightarrow FT'$$
$$T' \rightarrow *FT' \mid \epsilon$$
$$F \rightarrow \texttt{i} \mid (E)$$

$$X \rightarrow \beta_1 X' \mid \ldots \mid \beta_m X'$$
$$X' \rightarrow \alpha_1 X' \mid \ldots \mid \alpha_n X' \mid \epsilon$$

Parfois ça ne suffit pas

La grammaire $(\{a, b\}, \{S, A\}, S, P)$ avec

$$P = \{S \to aSb \,|\, A,\;\; A \to aA \,|\, \epsilon\}$$

- n'est pas ambiguë ;
- n'est pas récursive gauche ;
- est factorisée à gauche ;

mais elle n'est pas LL(1).

Au delà des grammaires LL(1) - 1

La grammaire $(\{a, b\}, \{A, B\}, A, P)$ avec

$$P = \{A \to abB \mid \epsilon,\ \ B \to Aaa \mid b\}$$

n'est pas LL(1).

En effet $Eps(A) = vrai$ et $a \in Premier(A) \cap Suivant(A)$.

Mais cette grammaire est LL(2) :
$Premier_2(A) = \{ab\}$
$Premier_2(B) = \{ab, aa, b\}$
$Suivant_2(A) = \{aa, \#\}$

Au delà des grammaires LL(1) - 2

$$A \rightarrow abB \mid \epsilon$$
$$B \rightarrow Aaa \mid b$$

	aa	ab	b	$\#$
A	$A \rightarrow \epsilon$	$A \rightarrow abB$	erreur	$A \rightarrow \epsilon$
B	$B \rightarrow Aaa$	$B \rightarrow Aaa$	$B \rightarrow b$	erreur

Table d'analyse LL(2)