CHAPITRE 4 : ANALYSE SYNTAXIQUE DESCENDANTE

L'analyse descendante part de l'axiome et tente de créer une chaîne identique à la chaîne d'entrée par dérivations successives. Ainsi, l'opération de base est le remplacement d'un symbole non terminal par une de ses parties droites. Quand l'analyseur doit remplacer un non terminal ayant plus d'une partie droite, il doit pouvoir effectuer le ***bon choix***. Dans ce chapitre, une méthode descendante d'analyse syntaxique va être exposée, il s'agit de la méthode d'analyse prédictive non récursive.

4.1 Principe de la méthode d'analyse prédictive non récursive

L'analyse prédictive non récursive est une méthode descendante déterministe où la dérivation à appliquer pour un non terminal (partie droite à choisir) est recherchée dans une table d'analyse. Un analyseur syntaxique prédictif est dirigé par une table et fonctionne en tenant à jour une pile explicite, ainsi, il possède les éléments représentés par la figure 4.1.

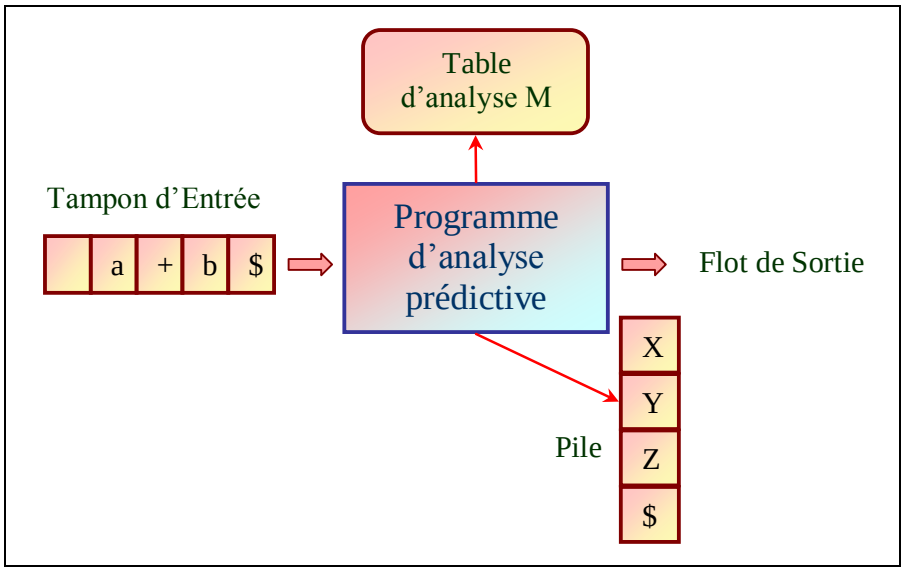

Figure 4.1. Modèle d'analyseur prédictif non récursif

- Le tampon d'entrée contient la chaîne à analyser suivie du symbole **$** qui est un marqueur de fin de chaîne.
- La pile contient une suite de symboles grammaticaux, avec $ marquant le fond de la pile. Initialement, l'axiome S se trouve au dessus de $.
- La table d'analyse M qui est une matrice où un élément M[X,a] correspond à un non terminal X, qu'on doit dériver, et à un terminal a (ou $), qu'on est en train de lire. Si on est arrivé à une étape où on doit dériver le non terminal **X** et qu'on lit le caractère **a** dans la chaîne d'entrée, on doit utiliser la règle de production qui se trouve dans M[X, a].
- Le programme d'analyse prédictive contrôle l'analyse syntaxique en considérant le symbole en sommet de pile X et le symbole d'entrée courant a. Ainsi, l'action de l'analyseur est déterminée par X et a, il y a plusieurs cas possibles (voir la section 4.4).

4.2 Fonctions Premier et Suivant

Les fonctions Premier et Suivant permettent d'effectuer la construction de la table M d'analyse prédictive en déterminant le contenu de ses entrées.

4.2.1 Fonction Premier (Début ou First)

Les symboles premiers d'une chaîne α sont les terminaux (y compris ε) pouvant se trouver au début de α (commencer α) directement ou après plusieurs dérivations.

Premier (α) = {$x \in V_T / \alpha \xrightarrow{*} x\beta$, $\alpha, \beta \in (V_T U V_N)^*$} sachant que : Si $\alpha \xrightarrow{*} \varepsilon$ alors $\varepsilon \in$ Premier (α)

Exemple 1

Soient les règles de productions suivantes pour une grammaire donnée :

$S \rightarrow Ba$
$B \rightarrow cP \mid bP \mid P \mid \varepsilon$
$P \rightarrow dS$

On se propose de déterminer les premiers du symbole S.

On a: $S \rightarrow Ba \rightarrow cPa$ donc: $c \in$ Prem(S)
On a: $S \rightarrow Ba \rightarrow bPa$ donc: $b \in$ Prem(S)
On a: $S \rightarrow Ba \rightarrow Pa \rightarrow dSa$ donc: $d \in$ Prem(S)
On a: $S \rightarrow Ba \rightarrow \varepsilon a \rightarrow a$ donc: $a \in$ Prem(S)

Nous pouvons en déduire que **Prem(S)= {a, b, c, d}**

Pour calculer Prem(X), il faut appliquer les trois règles suivantes jusqu'à ce qu'aucun terminal (ni ε) ne puisse être ajouté à Prem(X).

Détermination de Prem(X)

① Si $X \in V_T$ alors Prem(X)={X}

② Si $X \rightarrow \varepsilon$ est une production alors ajouter ε à Prem(X)

③ Si $X \in V_N$ et $X \rightarrow Y_1Y_2Y_3\ldots Y_k$ est une production avec $Y_i \in (V_T U V_N)$ alors :

- Ajouter les éléments de Prem(Y_1) sauf ε à Prem(X)
- S'il existe j (j $\in$ {2, 3,…k}) tel que : pour tout i (i =1..j-1), on a $\varepsilon \in$ Prem(Y_i)
  Alors Ajouter les éléments de Prem(Y_j) sauf ε à Prem(X)
- Si pour tout i (i =1..k), on a $\varepsilon \in$ Prem(Y_i) Alors Ajouter ε à Prem(X)

Exemple 2

Soient les règles de productions suivantes pour une grammaire donnée :

$S \rightarrow ABCe$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid cB \mid \varepsilon$
$C \rightarrow de \mid da \mid dA$

On se propose de déterminer les premiers de S, A B et C en appliquant les règles énoncées précédemment :

Prem(C) = {d}
Prem(B) = {b, c, ε}
Prem(A) = {a, ε}
Prem(S) = {a} U {b, c} U {d} = {a, b, c, d}

4.2.2 Fonction Suivant (Follow)

Pour chaque non terminal A, Suivant(A) définit l'ensemble des terminaux qui peuvent apparaître immédiatement à la droite de A dans une chaîne. Ceci signifie que :

$$Suivant\ (A) = \{x \in V_T / S \overset{*}{\rightarrow} \alpha A x \beta,\ \alpha, \beta \in (V_T U V_N)^*\}$$

Exemple 1

Soient les règles de productions suivantes pour une grammaire donnée :

$S \rightarrow aAB \mid aAd$

$B \rightarrow bB \mid c$

On se propose de déterminer les suivants du symbole A. On constate, d'après la première règle de production, que A peut être suivi par B ou d. D'après la deuxième règle, B peut commencer par b ou c. On peut donc déduire que : **Suiv(A)= {b, c, d}**

Pour calculer les suivants pour tous les non terminaux, il faut appliquer les trois règles suivantes jusqu'à ce qu'aucun terminal ne puisse être ajouté aux ensembles Suiv.

Détermination des Suivants

① Ajouter \$ à Suiv(S) où S est l'axiome.

② Pour chaque production $A \rightarrow \alpha B \beta$, le contenu de Prem($\beta$) sauf ε est ajouté à Suiv(B)

③ S'il existe une production $A \rightarrow \alpha B$ ou une production $A \rightarrow \alpha B \beta$ telle que Prem(β) contient ε ($\beta \overset{*}{\rightarrow} \varepsilon$)
Alors les éléments de Suiv(A) sont ajoutés à Suiv(B)

Exemple 2

Soient les règles de productions suivantes pour une grammaire donnée :

$S \rightarrow aSb \mid cd \mid SAe$

$A \rightarrow aAdB \mid \varepsilon$

$B \rightarrow bb$

On se propose de déterminer les suivants de S, A et B en appliquant les règles énoncées précédemment :

Suiv(S) = {\$, b, a, e}
Suiv(A) = {e, d}
Suiv(B) = {e, d}

4.3 Construction de la table d'analyse prédictive

L'idée principale de l'algorithme de construction de la table d'analyse prédictive M est que, dans le cas où on a $\mathbf{A \rightarrow \alpha}$ et $\mathbf{a \in Prem(\alpha)}$ alors l'analyseur développera **A** en α chaque fois que le symbole d'entrée courant est **a**.

Un problème se pose quand $\alpha \rightarrow \varepsilon$ ou $\alpha \overset{*}{\rightarrow} \varepsilon$, dans ce cas, il faudra développer **A** en α si le symbole d'entrée courant est dans Suiv(A) ou si le \$ a été atteint et que \$ $\in$ Suiv(A).

Etapes de l'algorithme de construction de la table d'analyse prédictive M

① Pour chaque production $A \rightarrow \alpha$ de la grammaire, exécuter les étapes ② et ③

② Pour chaque terminal $a \in Prem(\alpha)$, ajouter $A \rightarrow \alpha$ à M[A, a]

③ Si $\varepsilon \in Prem(\alpha)$ Alors ajouter $A \rightarrow \alpha$ à M[A, b] pour chaque $b \in Suiv(A)$
Si $\varepsilon \in Prem(\alpha)$ et $\$ \in Suiv(A)$ Alors ajouter $A \rightarrow \alpha$ à M[A, $]

④ Affecter *Erreur* aux entrées qui restent vides dans la table

4.4 Programme d'analyse prédictive

Le programme d'analyse prédictive contrôle l'analyse syntaxique en considérant le symbole en sommet de pile X et le caractère courant **a** dans le tampon d'entrée (chaîne à analyser). Ainsi, l'action de l'analyseur est déterminée par X et a, il y a trois cas possibles comme le montre l'algorithme suivant.

Algorithme d'analyse prédictive

1er cas : Si X=a=$ Alors l'analyseur s'arrête : analyse réussie, chaîne acceptée

2ème cas : Si X est un terminal
Si $X=a \neq \$$ Alors dépiler X et avancer dans la chaîne à analyser
Si $X \neq a$ Alors Erreur

3ème cas : Si X est un non terminal Alors utiliser M[X, a]
Si M[X, a] contient **une** production $X \rightarrow Y_1Y_2Y_3...Y_N$
Alors
Dépiler X
Empiler Y_N puis Y_{N-1} puis ...Y_2 puis Y_1
Emettre en sortie la production $X \rightarrow Y_1Y_2Y_3...Y_N$
Si M[X, a] est vide Alors Erreur

4.5 Exemples d'application de l'analyse prédictive

4.5.1 Exemple 1

Soient les règles de productions suivantes pour une grammaire donnée dont l'axiome est E :

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

On se propose de construire la table d'analyse prédictive puis d'analyser les chaînes id+id*id et id++id en utilisant la méthode d'analyse prédictive non récursive.

Pour cela, on commence d'abord par la détermination les premiers et les suivants pour chaque symbole non terminal de la grammaire afin de faciliter la construction de la table d'analyse. Par la suite, le programme d'analyse va être appliqué pour analyser chacune des chaînes demandées.

Détermination des premiers et des suivants

Après l'application des algorithmes présentés dans la section 4.2 (pages 36-37), on obtient :

	Prem	Suiv
E	(, id	$,)
E'	+, ε	$,)
T	(, id	+, $,)
T'	*, ε	+, $,)
F	(, id	*, +, $,)

Construction de la table d'analyse M

Après l'application de l'algorithme présenté dans la section 4.3 (pour chacune des productions A→ α de la grammaire), on obtient la table M dont les lignes correspondent aux symboles non terminaux (respecter l'ordre d'apparition dans les parties gauches des règles de production) et les colonnes correspondent aux symboles terminaux en plus du $ (pour les terminaux, respecter l'ordre d'apparition dans les parties droites des règles de production):

Productions de la grammaire

E → TE' prem (TE')= { (, id }
E' → +TE' prem (+TE')= { + }
E' → ε suiv (E')= { $,) }
T → FT' prem (FT')= { (, id }
T' → *FT' prem (*FT')= { * }
T' → ε suiv (T')= { +, $,) }
F → (E) prem ((E))= { (}
F → id prem (id)= { id }

	+	*	(	)	id	$
E			E→TE'		E→TE'	
E'	E' →+TE'			E'→ε		E'→ε
T			T→FT'		T→FT'	
T'	T'→ε	T'→*FT'		T'→ε		T'→ε
F			F→(E)		F→id	

*Analyse de la chaîne id+id*id*

Après l'application de l'algorithme d'analyse prédictive présenté dans la section 4.4 (page 38), on obtient les transitions effectuées par un analyseur prédictif sur la chaîne d'entrée id+id*id, comme suit:

Pile	Entrée	Sortie
$E	id+id*id$	**E→TE'**
$E' T	id+id*id$	**T→FT'**
$E' T' F	id+id*id$	**F→id**
$E' T' id	id+id*id$	Comparer
$E' T'	+id*id$	**T'→ε**
$E'	+id*id$	**E' →+TE'**
$E' T +	+id*id$	Comparer
$E' T	id*id$	**T→FT'**
$E' T' F	id*id$	**F→id**
$E' T' id	id*id$	Comparer
$E' T'	*id$	**T'→*FT'**
$E' T' F *	*id$	Comparer
$E' T' F	id$	**F→id**
$E' T' id	id$	Comparer
$E' T'	$	**T'→ε**
$E'	$	**E'→ε**
$	$	Analyse réussie, chaîne acceptée

L'arbre syntaxique de la chaîne id+id*id peut être déduit directement à partir de la colonne des sorties (productions en gras) comme le montre la figure 4.2.

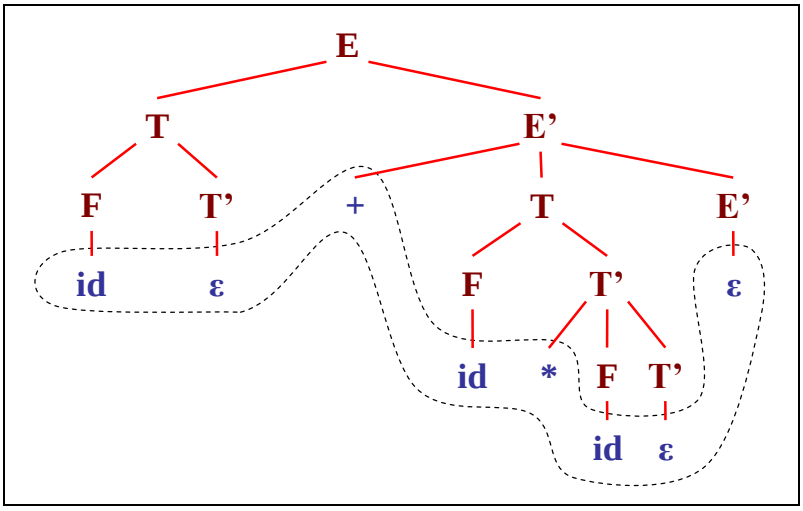

Figure 4.2. Arbre syntaxique obtenu par l'analyseur prédictif pour la chaîne id+id*id

Analyse de la chaîne id++id

Après l'application de l'algorithme d'analyse prédictive présenté dans la section 4.4 (page 38), on obtient les transitions effectuées par un analyseur prédictif sur la chaîne d'entrée id++id, comme suit:

Pile	Entrée	Sortie
$E	id++id$	E→TE'
$E' T	id++id$	T→FT'
$E' T' F	id++id$$	F→id
$E' T' id	id++id$	Comparer
$E' T'	++id$	T'→ε
$E'	++id$	E' →+TE'
$E' T +	++id$	Comparer
$E' T	+id$	Erreur, chaîne rejetée

4.5.2 Exemple 2

Soient les règles de productions suivantes pour une grammaire donnée dont l'axiome est S, on veut construire la table d'analyse prédictive pour cette grammaire:

S → aAb

A → cd | c

Détermination des premiers et des suivants

Après l'application des algorithmes présentés dans la section 4.2, on obtient :

	Prem	Suiv
S	a	$
A	c	b

Construction de la table d'analyse M

Après l'application de l'algorithme présenté dans la section 4.3 (pour chacune des productions A→ α de la grammaire), on obtient la table M suivante :

Productions de la grammaire

S → aAb prem (aAb)= { a }

A → cd prem (cd)= { c }

A → c prem (c)= { c }

	a	b	c	d	$
S	S → aAb				
A			A→ cd A → c		

On remarque que M[A, c] est une case contenant deux règles de production, donc il y a deux développements possibles : A par c ou A par cd. Dans ce genre de situations (entrées dans M définies de façon multiple), on ne peut pas utiliser la méthode d'analyse prédictive et on dit que **la grammaire n'est pas LL(1)**.

4.6 Grammaire LL(1)

L'algorithme d'analyse prédictive n'est pas applicable si la table d'analyse contient des entrées multiples (plusieurs productions pour une même case M[X, a]) car on ne peut pas savoir quelle production appliquer.

***Définition** : On appelle grammaire LL(1) une grammaire pour laquelle la table d'analyse prédictive ne contient aucune case définie de façon multiple, chaque case de la table contient **au plus** une règle de production.*

L : **L**eft to right scanning (on parcourt ou on analyse la chaîne en entrée de la gauche vers la droite)

L : **L**eftmost derivation (on utilise les dérivations gauches)

1 : on utilise **un seul** symbole d'entrée de prévision à chaque étape nécessitant la prise d'une décision d'action d'analyse.

4.6.1 Conditions pour qu'une grammaire soit LL(1)

On peut montrer formellement qu'une grammaire est LL(1), si et seulement si, à chaque fois que l'on a une production de la forme $A \rightarrow \alpha|\beta$, les trois conditions suivantes sont vérifiées :

C1 : Pour aucun terminal *a*, α et β ne se dérivent toutes les deux en des chaînes commençant par *a*. Autrement dit : **Prem(α) $\cap$ Prem(β) = $\varnothing$**.

C2 : Une des chaînes (α ou β) au plus peut se dériver en la chaîne vide. Autrement dit : **($\beta \rightarrow^* \varepsilon$) ou bien ($\alpha \rightarrow^* \varepsilon$) mais pas les deux à la fois**.

C3 : Si ($\beta \rightarrow^* \varepsilon$), α ne se dérive pas en une chaîne commençant par un terminal de Suiv(A). Autrement dit : **Si ($\beta \rightarrow^* \varepsilon$) alors Prem($\alpha$) $\cap$ Suiv(A) = $\varnothing$**.

Exemple 1

On se propose de montrer si la grammaire G1 ayant les règles de productions suivantes est LL(1) (c'est la grammaire de l'exemple 2, section 4.5.2).

S $\rightarrow$ aAb
A $\rightarrow$ cd | c

Pour la production A $\rightarrow$ cd | c, on constate que Prem(cd) $\cap$ Prem(c) = {c} $\cap$ {c} = {c} $\neq \varnothing$, ceci signifie que la condition C1 n'est pas vérifiée donc la grammaire **G1 n'est pas LL(1)**.

Exemple 2

Soient les règles de productions suivantes pour une grammaire G2 dont l'axiome est E (c'est la grammaire de l'exemple 1, section 4.5.1):

E $\rightarrow$ TE'
E' $\rightarrow$ +TE' | ε
T $\rightarrow$ FT'
T' $\rightarrow$ *FT' | ε
F $\rightarrow$ (E) | id

On se propose de montrer si G2 est une grammaire LL(1).

Pour la production E′ → +TE′ | ε

C1 : Prem(+TE′) ∩ Prem(ε) = {+} ∩ {ε} = ∅ vérifiée

C2 : vérifiée car +TE′ ne peut pas se dériver en ε

C3 : Prem(+TE′) ∩ Suiv(E′) = {+} ∩ {$,)} = ∅ vérifiée

Pour la production T′ → *FT′ | ε

C1 : Prem(*FT′) ∩ Prem(ε) = {*} ∩ {ε} = ∅ vérifiée

C2 : vérifiée car *FT′ ne peut pas se dériver en ε

C3 : Prem(*FT′) ∩ Suiv(T′) = {*} ∩ {+, $,)} = ∅ vérifiée

Pour la production F → (E) | id

C1 : Prem((E)) ∩ Prem(id) = {(} ∩ {id} = ∅ vérifiée

C2 : vérifiée car (E) →* ε et id→* ε sont toutes les deux impossibles

C3 : non applicable (à cause de C2) vérifiée

Conclusion : Les trois conditions (C1, C2 et C3) sont vérifiées pour toutes les productions de la forme A→α|β, donc la grammaire **G2 est LL(1)**.

4.6.2 Transformations pour qu'une grammaire devienne LL(1)

> **Théorème** : Une grammaire **ambiguë**, ou **récursive à gauche** ou **non factorisée à gauche**, **n'est pas** une grammaire **LL(1)**.

Ainsi, si la table d'analyse prédictive d'une grammaire comporte des cases définies de façon multiple, cette grammaire n'est pas LL(1) mais on peut essayer de la transformer en une grammaire LL(1) en effectuant les transformations suivantes :

1. La rendre non ambiguë. Pour cela, il n'existe malheureusement pas de méthode : une grammaire ambiguë est une grammaire mal conçue. Il faut tenter de refaire la conception de la grammaire (tenir compte de la priorité des opérateurs, associer le premier else au dernier if…).
2. Eliminer la récursivité à gauche si cela est nécessaire (voir section 4.6.2.1).
3. Factoriser les règles de production à gauche si cela est nécessaire (voir section 4.6.2.2).

Par la suite, on peut construire la table d'analyse prédictive de la grammaire obtenue après transformations. Si aucune case n'est définie de manière multiple, alors la grammaire obtenue est LL(1) et la méthode d'analyse prédictive peut être appliquée.

Malheureusement, il existe des grammaires pour lesquelles il n'y a pas de transformation les rendant LL(1), dans ce cas, la méthode d'analyse prédictive ne peut être appliquée et il faudra utiliser une autre méthode d'analyse.

4.6.2.1 Elimination de la récursivité à gauche

Définition : Une grammaire est récursive à gauche si elle contient un non terminal A tel que $\mathbf{A} \overset{*}{\rightarrow} \mathbf{A}\alpha$ (α étant une chaîne quelconque).

Les méthodes d'analyse descendante ne peuvent pas fonctionner avec une grammaire récursive à gauche, il faut donc la transformer en une grammaire équivalente en éliminant la récursivité à gauche.

Elimination de la récursivité à gauche immédiate

Dans le **cas le plus simple**, il s'agit d'une récursivité à gauche immédiate avec la présence d'un non terminal A et une production de la forme $\mathbf{A \rightarrow A\alpha}$. Pour éliminer la récursivité à gauche immédiate, il faut remplacer toute production de la forme :

$A \rightarrow A\alpha | \beta$ par : $\mathbf{A \rightarrow \beta A'}$

$\mathbf{A' \rightarrow \alpha A' | \varepsilon}$

D'une **manière plus générale**, pour éliminer la récursivité à gauche immédiate, il faut procéder comme suit :

1. Regrouper les A-productions de la façon suivante : $\mathbf{A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_M | \beta_1 | \beta_2 | \ldots | \beta_N}$
2. Remplacer les A-productions par : $\mathbf{A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_N A'}$

   $\mathbf{A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_M A' | \varepsilon}$

Exemple 1

Soit la grammaire G_1 ayant les règles de production suivantes :

E → E+T | T
T → T*F | F
F → (E) | id

Après l'élimination de la récursivité à gauche, on obtient une grammaire G'_1 équivalente à G_1, dont les règles de production sont :

E → TE'
E' → +TE' | ε
T → FT'
T' → *FT' | ε
F → (E) | id

Exemple 2

Soit la grammaire G_2 ayant les règles de production suivantes :

S → ScA | B
A → Aa | ε
B → Bb | d | e

Après l'élimination de la récursivité à gauche, on obtient une grammaire G'_2 équivalente à G_2, dont les règles de production sont :

S → BS'
S' → cAS' | ε
A → A'
A' → aA' | ε
B → dB' | eB'
B' → bB' | ε

Il est aussi possible que la récursivité gauche ne soit pas immédiate, mais qu'on ne puisse la détecter qu'après avoir effectué des dérivations. D'une manière encore plus générale, pour éliminer la récursivité à gauche, on peut appliquer l'algorithme suivant à condition que la grammaire soit propre, autrement dit, ne contienne pas de production $A \rightarrow \varepsilon$:

Algorithme Elimination de récursivité gauche d'une grammaire propre;
Début
  Ordonner les non terminaux $A_1, A_2, \ldots A_N$;
  Pour i=1 à N **Faire**
  Début
    Pour k=1 à i-1 **Faire**
    Début
      Remplacer chaque production de la forme
      $A_i \rightarrow A_k\alpha | \lambda$ où $A_k \rightarrow \beta_1 | \beta_2 | \ldots | \beta_P$
      par $A_i \rightarrow \beta_1\alpha | \beta_2\alpha | \ldots | \beta_P\alpha | \lambda$
    Fin;
    Eliminer les récursivités à gauche immédiate des productions A_i
  Fin ;
Fin ;

Exemple 3

Soit la grammaire G_3 ayant les règles de production suivantes :

$Z \rightarrow Aa \mid z$
$A \rightarrow Ac \mid Zd$

On commence par ordonner Z et A (A_1=Z, A_2=A).
Pour i=1, pas de récursivité immédiate dans $Z \rightarrow Aa \mid z$
Pour i=2 et k=1, on obtient $A \rightarrow Ac \mid Aad \mid zd$
on élimine la récursivité immédiate :
$A \rightarrow zdA'$
$A' \rightarrow cA' \mid adA' \mid \varepsilon$

On obtient ainsi une grammaire G'_3 équivalente à G_3, non récursive à gauche et dont les règles de production sont :

$Z \rightarrow Aa \mid z$
$A \rightarrow zdA'$
$A' \rightarrow cA' \mid adA' \mid \varepsilon$

Exemple 4

Soit la grammaire G_4 ayant les règles de production suivantes :

$S \rightarrow Aa \mid b$
$A \rightarrow Ac \mid Sd \mid BA \mid c$
$B \rightarrow SSc \mid a$

On commence par ordonner S, A, B (A_1=S, A_2=A, A_3=B).
Pour i=1, pas de récursivité immédiate dans $S \rightarrow Aa \mid b$

Pour i=2 et k=1, on obtient $A \rightarrow Ac \mid Aad \mid bd \mid BA \mid c$
on élimine la récursivité immédiate
$A \rightarrow bdA' \mid BAA' \mid cA'$
$A' \rightarrow cA' \mid adA' \mid \varepsilon$

Pour i=3 et k=1, on obtient : B → AaSc | bSc | a

et k=2 donne : B → bdA'aSc | BAA'aSc | cA'aSc | bSc | a

on élimine la récursivité immédiate

B → bdA'aScB' | cA'aScB' | bScB' | aB'

B' → AA'aScB' | ε

On obtient ainsi une grammaire G'_4 équivalente à G_4, non récursive à gauche et dont les règles de production sont :

S → Aa | b

A → bdA' | BAA' | cA'

A' → cA' | adA' | ε

B → bdA'aScB' | cA'aScB' | bScB' | aB'

B' → AA'aScB' | ε

Rappel (grammaire propre)

Une grammaire est dite propre si elle ne contient aucune production A → ε . Pour rendre une grammaire propre, il faut rajouter une production dans laquelle le A est remplacé par ε , ceci pour chaque A apparaissant en partie droite d'une production, et pour chaque A d'une production A → ε .

Exemple (grammaire propre)

Soit la grammaire G_5 ayant les règles de production suivantes, on veut rendre cette grammaire propre :

S → aTb | aU

T → bTaTA | ε

U → aU | b

On obtient ainsi une grammaire propre G'_5 équivalente à G_5, dont les règles de production sont :

S → aTb | ab | aU

T → bTaTA | baTA | bTaA | baA

U → aU | b

4.6.2.2 Factorisation à gauche

Cette transformation est un moyen pour obtenir une grammaire convenant à l'analyse prédictive. L'idée de la factorisation à gauche est de réécrire les A-productions du type $\mathbf{A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_N \mid \lambda}$ de sorte à reporter la décision (quelle production choisir ?) jusqu'à ce que suffisamment de texte soit lu pour faire le bon choix.

Pour effectuer la factorisation à gauche, il faut remplacer :

$\mathbf{A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_N \mid \lambda}$ par $\mathbf{A \rightarrow \alpha A' \mid \lambda}$

$\mathbf{A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_N}$

où $\alpha \in (V_T \cup V_N)^+$ et λ représente toutes les alternatives qui ne commencent pas par α.

Remarque :

Pour chaque non terminal A, il faut essayer de trouver le plus long préfixe α commun à deux ou plusieurs des parties droites des A-productions. Il faut refaire l'opération de factorisation jusqu'à ne plus trouver de préfixes communs.

Exemple 1

Soit la grammaire G_1 ayant les règles de production suivantes :

$S \rightarrow aC \mid abD \mid aBc$

Après factorisation à gauche, on obtient une grammaire G'_1 équivalente à G_1, dont les règles de production sont :

$S \rightarrow aS'$

$S' \rightarrow C \mid bD \mid Bc$

Exemple 2

Soit la grammaire G_2 ayant les règles de production suivantes :

$S \rightarrow aEbS \mid aEbSeB \mid a$

$E \rightarrow bcB \mid bca$

$B \rightarrow ba$

Après une première factorisation à gauche, on obtient :

$S \rightarrow aEbSS' \mid a$

$S' \rightarrow \varepsilon \mid eB$

$E \rightarrow bcE'$

$E' \rightarrow B \mid a$

$B \rightarrow ba$

Finalement, la grammaire G'_2 factorisée à gauche et équivalente à G_2, aura pour règles de production :

$S \rightarrow aS''$

$S'' \rightarrow EbSS' \mid \varepsilon$

$S' \rightarrow \varepsilon \mid eB$

$E \rightarrow bcE'$

$E' \rightarrow B \mid a$

$B \rightarrow ba$