Algorithmes pour l'analyse syntaxique

Tony Bourdier
Version 1.2 − 8 juillet 2008

Réduction inférieure *(suppression des non-terminaux improductifs)*

Rappel : Soit $G = (N, T, \rightarrow, X)$ une grammaire. Un non-terminal $A \in N$ est dit **improductif** s'il n'existe pas de mot $\alpha \in T^*$ tel que

$$A \overset{*}{\rightarrowtail} \alpha$$

Dans le cas contraire, A est dit **productif**.

Algorithme 1 Calcul des non-terminaux productifs

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
Sortie : Ensemble des non-terminaux de G productifs : $Prod$
Initialisation : $Prod \leftarrow \varnothing$
  pour tout $A \rightarrow \alpha$ où $\alpha \in T^*$ **faire**
    $Prod \leftarrow Prod \cup \{A\}$
  fin pour
  tantque $New \neq \varnothing$ **faire**
    $New \leftarrow \varnothing$
    pour tout $A \rightarrow \alpha$ tel que $A \notin Prod$ **faire**
      si $\alpha \in (T \cup Prod)^*$ **alors**
        $New \leftarrow New \cup \{A\}$
      fin si
    fin pour
    $Prod \leftarrow Prod \cup New$
  fin tantque

Exemple : Soit $G = (N = \{A, B, C\}, T = \{a, b\}, \rightarrow, A)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} A & \rightarrow & b\,B \mid a\,C \\ B & \rightarrow & b \\ C & \rightarrow & a\,C \end{array} \right\}$$

- Initialisation : $Prod \leftarrow \{B\}$ car $B \rightarrow b$ et $b \in T^*$
- Itération 1 : $Prod \leftarrow Prod \cup \{A\}$ car $A \rightarrow b\,B$ et $b \in T^*,\ B \in Prod$
- Itération 2 : $Prod \leftarrow Prod \cup \varnothing$
- fin

Donc $Prod = \{A, B\}$ donc le non-terminal C est improductif. On peut donc réduire la relation de production de G à l'ensemble suivant :

$$\left\{ \begin{array}{lcl} A & \rightarrow & b\,B \\ B & \rightarrow & b \end{array} \right\}$$

Réduction supérieure *(suppression des son-terminaux inaccessibles)*

Rappel : Soit $G = (N, T, \rightarrow, X)$ une grammaire. Un non-terminal $A \in N$ est dit **inaccessible** s'il n'existe pas $\alpha, \beta \in (N \cup T)^*$ tel que

$$X \overset{*}{\rightarrowtail} \alpha A \beta$$

Sinon, A est dit **accessible**.

Algorithme 2 Calcul des non-terminaux accessibles

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
Sortie : Ensemble des non-terminaux de G accessibles : Acc
Initialisation : $Acc \leftarrow \{X\}$
  tantque $New \neq \varnothing$ **faire**
    $New \leftarrow \varnothing$
    pour tout $A \rightarrow \alpha_1 B_1 \alpha_2 \ldots \alpha_n B_n \alpha_{n+1}$ tel que $A \in Acc$ et $\alpha_i \in T^*,\ B_i \in N$ **faire**
      pour tout $B_i \notin Acc$ **faire**
        $New \leftarrow New \cup B_i$
      fin pour
    fin pour
    $Acc \leftarrow Acc \cup New$
  fin tantque

Exemple : Soit $G = (N = \{X, Y, A, B, C, D\}, T = \{a, b, c, d\}, \rightarrow, X)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} X & \rightarrow & Y \\ Y & \rightarrow & Y\,D \ \mid\ Y\,a \ \mid\ b \\ A & \rightarrow & B \\ B & \rightarrow & B\,d \ \mid\ d \\ C & \rightarrow & c \\ D & \rightarrow & D\,C \end{array} \right\}$$

On a :
- Initialisation : $Acc \leftarrow \{X\}$
- Itération 1 : $New \leftarrow \{Y\}$
- Itération 2 : $New \leftarrow \{D\}$
- Itération 3 : $New \leftarrow \{C\}$
- Itération 4 : $New \leftarrow \varnothing$
- fin

On peut donc supprimer les non-terminaux A et B ainsi que toutes les règles les contenant.

Réduction

Rappel : Une grammaire est dite **réduite** si tous ses non-terminaux sont productifs et accessibles.

Algorithme 3 Réduction

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
Sortie : Réduite de G
Effectuer la réduite inférieure de G
Effectuer la réduite supérieure de la grammaire résultante de l'étape précédente

Récursivité à gauche

Rappel : Une grammaire est dite **récursive gauche immédiate** si elle contient des règles de la forme

$$A \rightarrow A\alpha$$

où $A \in N$ et $\alpha \in (N \cup T)^*$

Algorithme 4 Dérécursivation gauche

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$
pour toute règle de la forme $A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \ldots \mid \beta_p$ **faire**
remplacer par $\left\{ \begin{array}{l} A \rightarrow \beta_1 A' \mid \ldots \mid \beta_p A' \\ A' \rightarrow \alpha_1 A' \mid \ldots \mid \alpha_n A' \mid \epsilon \end{array} \right.$
fin pour

Exemple : Soit $G = (\{E, T, F\}, \{i\}, \rightarrow, E)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (E) \mid i \end{array} \right\}$$

- Itération 1 : $E \rightarrow E\overbrace{+T}^{\alpha_1} \mid \overbrace{T}^{\beta_1}$ est remplacé par $\left\{ \begin{array}{lcl} E & \rightarrow & TE' \\ E' & \rightarrow & +TE' \mid \epsilon \end{array} \right.$
- Itération 2 : $T \rightarrow T\overbrace{*F}^{\alpha_1} \mid \overbrace{F}^{\beta_1}$ est remplacé par $\left\{ \begin{array}{lcl} T & \rightarrow & FT' \\ T' & \rightarrow & *FT' \mid \epsilon \end{array} \right.$
- fin

Remarque : On ne traitera pas cette année la récursivité indirecte (i.e. les règles de la forme $A \overset{+}{\rightarrowtail} A\alpha$*).*

Factorisation à gauche

Algorithme 5 Factorisation à gauche

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$

pour toute règle de la forme $A \rightarrow \alpha\beta_1 \mid \ldots \mid \alpha\beta_n \mid \gamma_1 \mid \ldots \mid \gamma_p$ **faire**

remplacer par $\left\{ \begin{array}{l} A \rightarrow \alpha A' \mid \gamma_1 \mid \ldots \mid \gamma_p \\ A' \rightarrow \beta_1 \mid \ldots \mid \beta_n \end{array} \right.$

fin pour

Exemple : Soit $G = (\{E, T\}, \{i\}, \rightarrow, E)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} E & \rightarrow & E+T \mid E+E \mid F \\ T & \rightarrow & T*(E) \mid T*i \mid T* \mid i \end{array} \right\}$$

- Itération 1 : $E \rightarrow E+T \mid E+E \mid F$ est remplacé par $\left\{ \begin{array}{lcl} E & \rightarrow & E+E' \mid F \\ E' & \rightarrow & T \mid E \end{array} \right.$
- Itération 2 : $T \rightarrow T*(E) \mid T*i \mid T* \mid i$ est remplacé par $\left\{ \begin{array}{lcl} T & \rightarrow & T*T' \mid i \\ T' & \rightarrow & (E) \mid i \mid \epsilon \end{array} \right.$
- fin

Production vide

Rappel : On note $Vide$ l'ensemble composé de tous les non-terminaux pouvant produire le vide :

$$Vide = \{\ A \in N \mid A \overset{*}{\rightarrowtail} \epsilon\ \}$$

On étend cette définition aux mots de N^* :

$$A = A_1 \ldots A_n \in Vide \quad \Leftrightarrow \quad A_1 \in Vide, \ldots, A_n \in Vide \text{ ou } A = \epsilon$$

Algorithme 6 Calcul de l'ensemble $Vide$

Entrée : Grammaire algébrique : $G = (N, T, \rightarrow, X)$

Sortie : Ensemble des non-terminaux pouvant produire le vide : $Vide$

pour tout $A \rightarrow \epsilon$ **faire**

$Vide \leftarrow Vide \cup \{A\}$

fin pour

tantque $New \neq \varnothing$ **faire**

$New \leftarrow \varnothing$

pour tout $A \rightarrow A_1 \ldots A_n$ tel que $A_1, \ldots, A_n \in Vide$ **faire**

$New \leftarrow New \cup \{A\}$

fin pour

$Vide \leftarrow Vide \cup New$

fin tantque

Exemple : Soit $G = (\{A, B, C, D\}, \{a, b, c\}, \rightarrow, A)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcllcl} A & \rightarrow & bB \mid CD \mid Bc & C & \rightarrow & a \mid \epsilon \\ B & \rightarrow & ab \mid cC & D & \rightarrow & \epsilon \end{array} \right\}$$

$Vide = \{C, D, A\}$

Calcul des premiers

Rappel : Pour tout mot $\omega \in (N \cup T)^*$, on appelle premiers de ω et on note $Premier(\omega)$ l'ensemble :

$$Premier(\omega) = \{a \in T \mid A \overset{*}{\rightarrowtail} a\beta \}$$

Algorithme 7 Calcul des premiers d'un mot

Entrée : Mot $\omega = \alpha_1\, \alpha_2\, \ldots\, \alpha_n \in (N \cup T)^*$
Sortie : $Premier(\omega)$

```
si α1 ∈ T alors
    Premier(ω) = {α1}
sinon
    Premier(ω) = ∅
    pour tout α1 → β faire
        Premier(ω) ← Premier(ω) ∪ Premier(β)
    fin pour
    pour i de 1 à n − 1 faire
        si α1, ..., αi ∈ Vide alors
            Premier(ω) ← Premier(ω) ∪ Premier(αi+1)
        fin si
    fin pour
fin si
```

Exemple : Soit $G = (\{S, A, B, D\}, \{a, b, c, d\}, \rightarrow, S)$ avec $\rightarrow$ définie par :

$$\left\{ \begin{array}{lcl} S & \rightarrow & DA \mid B \\ A & \rightarrow & aA \mid Dc \mid \epsilon \\ B & \rightarrow & bB \mid \epsilon \\ D & \rightarrow & dD \mid \epsilon \end{array} \right\}$$

- $Premier(D) = \{d\}$
- $Premier(B) = \{b\}$
- $Premier(A) = \{a, c, d\}$:

$$\begin{array}{rcl} Premier(A) & = & Premier(aA) \cup Premier(Dc) \\ & = & Premier(a) \cup Premier(D) \cup \underbrace{Premier(c)}_{\text{car } D \in Vide} \\ & = & \{a\} \cup \{d\} \cup \{c\} = \{a, c, d\} \end{array}$$

- $Premier(S) = \{a, b, c, d\}$:

$$\begin{array}{rcl} Premier(S) & = & Premier(DA) \cup Premier(B) \\ & = & Premier(D) \cup \underbrace{Premier(A)}_{\text{car } D \in Vide} \cup Premier(B) \\ & = & \{a, b, c, d\} \end{array}$$

Calcul des suivants

Rappel : Soit $G = (N, T, \rightarrow, X)$ une grammaire algébrique. Pour tout non-terminal $E \in N$, on appelle suivants de E et on note $Suivant(E)$ l'ensemble :

$$Suivant(E) = \{a \in T \mid S \rightarrow \alpha E a \beta \} \cup \{ \$ \mid X \overset{*}{\rightarrowtail} \alpha E \}$$

Algorithme 8 Calcul des suivants d'un non-terminal

Entrée : Non-terminal E
Sortie : $Suivant(E)$
  pour tout $A \rightarrow \alpha E \beta$ *(cas direct)* **faire**
    $Suivant(E) \leftarrow Suivant(E) \cup Premier(\beta)$
  fin pour
  pour tout $A \rightarrow \alpha E$ ou $A \rightarrow \alpha E \beta$ tel que $\beta \in Vide$ *(cas final)* **faire**
    $Suivant(E) \leftarrow Suivant(E) \cup Suivant(A)$
  fin pour
  si E est l'axiome **alors**
    $Suivant(E) \leftarrow Suivant(E) \cup \{\$\}$
  fin si

Exemple : Soit $G = (N = \{E, E', T, T', F\}, T = \{(, i,), +, *\}, \rightarrow, E)$ une grammaire algébrique dont la relation de production est définie par :

$$\begin{cases} E & \rightarrow & TE' \\ E' & \rightarrow & +TE' \mid \epsilon \\ T & \rightarrow & FT' \\ T' & \rightarrow & *FT' \mid \epsilon \\ F & \rightarrow & (E) \mid i \end{cases}$$

On a $Vide = \{E', T'\}$
- $Suivant(E) = \{\$,)\}$
- $Suivant(E') = Suivant(E) = \{\$,)\}$
- $Suivant(T) = Premier(E') \cup Suivant(E) \cup Suivant(E') = \{+, \$,)\}$
- $Suivant(T') = Suivant(T) = \{+, \$,)\}$
- $Suivant(F) = Premier(T') \cup Suivant(T) \cup Suivant(T') = \{*, +, \$,)\}$

Calcul des symboles directeurs

Algorithme 9 Calcul des symboles directeurs d'une règle

Entrée : Règle $A \to \alpha$
Sortie : $SD(A \to \alpha)$

```
SD(A → α) ← Premier(α)
si α ∈ Vide alors
    SD(A → α) ← SD(A → α) ∪ Suivant(A)
fin si
```

Exemple : Soit $G = (N = \{E, E', T, T', F\}, T = \{(, i,), +, *\}, \to, E)$ une grammaire algébrique dont la relation de production est définie par :

$$\begin{cases} E & \to & TE' \\ E' & \to & +TE' \mid \epsilon \\ T & \to & FT' \\ T' & \to & *FT' \mid \epsilon \\ F & \to & (E) \mid i \end{cases}$$

On a $Vide = \{E', T'\}$

- $SD(E \to TE') = Premier(TE') = \{(, i\}$
- $SD(E' \to +TE') = Premier(+TE') = \{+\}$
- $SD(T \to FT') = Premier(FT') = \{(, i\}$
- $SD(T' \to *FT') = \{*\}$
- $SD(F \to i) = Premier(i) = \{i\}$
- $SD(F \to (E)) = Premier((E)) = \{(\}$
- $SD(E' \to \epsilon) = Suivant(E') = \{\$,)\}$
- $SD(T' \to \epsilon) = Suivant(T') = \{+, \$,)\}$

Construction de la table d'analyse LL(1)

Algorithme 10 Construction de la table d'analyse LL(1)

Entrée : Grammaire algébrique : $G = (N, T, \to, X)$
Sortie : Table d'analyse : Tab

```
/* Chaque ligne correpond à un non-terminal */
/* Chaque colonne correspond à un terminal */
pour tout A → α faire
    pour tout d ∈ SD(A → α) faire
        Tab(A, d) ← "A → α"
    fin pour
fin pour
```