amU Faculté des sciences Aix Marseille Université

Année 2024-25

Domaine Sciences et Technologies

Licence mathématiques-informatique

Compilation : TD 3

Codes UE : SIN6U64L

TD 3 : Analyse SLR

1 Introduction

L'objectif de ce TD est de construire des analyseurs SLR à partir des grammaires suivantes :

— G_1 : parenthèses équilibrées

1. $P \to$ `(` P `)` P
2. $P \to \varepsilon$

— G_2 : déclarations simplifiées pour un langage de programmation : `dv` = déclaration de variable, `df` = déclaration de fonction

1. $P \to V$ `;` F
2. $V \to$ `dv` V'
3. $V' \to$ `,` `dv` V'
4. $V' \to \varepsilon$
5. $F \to$ `df` F
6. $F \to \varepsilon$

Suivez les étapes ci-dessous pour chacune des grammaires G_1 et G_2.

2 Augmentation de la grammaire

Créez une grammaire augmentée G' en rajoutant un nouvel axiome S dont la seule production est $S \to P$ vers l'ancien axiome P de la grammaire originale G.

3 Articles/FERMETURE

Un *article* est une production avec un marqueur spécial $\bullet$ en partie droite de la production. Ce marqueur indique la partie du manche qui a déjà été observée sur la pile.

Construisez l'ensemble d'articles initial Fermeture($\{S \to \bullet P\}$). La fonction Fermeture(I) prend un ensemble d'articles et, pour tout article $A \to \alpha \bullet B\gamma$ dans Fermeture(I), ajoute $B \to \bullet\beta$ pour toute règle $B \to \beta$ de la grammaire.

4 Fonction ALLER_A(I, X)

À partir d'un ensemble d'articles I, et d'un symbole X terminal ou non terminal de la grammaire, la fonction Aller_a(I, X) est la Fermeture de l'ensemble de tous les articles $A \to \alpha X \bullet \beta$ tels que $A \to \alpha \bullet X\beta$ est

dans I. C'est-à-dire, la fonction ALLER_A(I, X) fait avancer le marqueur $\bullet$ pour tous les articles de I dont le prochain symbole après le marqueur est X.

Calculez la collection d'ensembles d'articles de la grammaire à l'aide de la fonction ALLER_A(I, X) à partir de l'ensemble initial déjà calculé, pour tous les symboles X après le marquer $\bullet$. Itérez ce processus jusqu'à ce qu'aucun nouvel ensemble d'articles ne soit ajouté à la collection.

5 Automate $LR(0)$

Dessinez l'automate $LR(0)$ de la grammaire de la façon suivante :

- Les états sont les ensembles d'articles I calculés précédemment
- Les transitions sont données par les valeurs de ALLER_A(I, X)
- L'état initial est FERMETURE$(\{S \rightarrow \bullet P\}$
- Une transition étiquetée \$ depuis l'état FERMETURE$(\{S \rightarrow P \bullet\})$ mène à l'état `accept`

6 PREMIER(X)/SUIVANT(X)

Pour chaque symbole non terminal X de la grammaire, calculez PREMIER(X) et SUIVANT(X).

Pour calculer PREMIER(X), appliquer les règles suivantes jusqu'à ce qu'aucun symbole ne puisse être ajouté à PREMIER(X).

1. Si $X \rightarrow \varepsilon$ est une production de la grammaire, on ajoute ε à PREMIER(X).
2. Si $X \rightarrow Y_1 \dots Y_k \in P$, mettre a dans PREMIER(X) s'il existe i tel que a est dans PREMIER(Y_i) et que ε est dans tous les PREMIER$(Y_1) \dots$ PREMIER(Y_{i-1}). Si $\varepsilon \in$ PREMIER$(Y_j) \forall j, 1 \leq j \leq k$, alors on ajoute ε à PREMIER(X) [1].

Pour calculer SUIVANT(X), appliquer les règles suivantes jusqu'à ce qu'aucun symbole ne puisse être ajouté à SUIVANT(X).

1. Mettre \$ dans SUIVANT(S).
2. si $A \rightarrow \alpha X \beta$, le contenu de PREMIER$(\beta)$, excepté ε, est ajouté à SUIVANT(X).
3. s'il existe une règle $A \rightarrow \alpha X$ ou une règle $A \rightarrow \alpha X \beta$ telle que $\varepsilon \in$ PREMIER(β) (c'est-à -dire $\beta \stackrel{*}{\Rightarrow} \varepsilon$), les éléments de SUIVANT(A) sont ajoutés à SUIVANT(X).

7 Construction de la table SLR

Construisez la table d'analyse SLR pour la grammaire comme suit.

1. Construire $C = \{I_0, I_1, \dots, I_n\}$ la collection d'ensemble d'articles $LR(0)$ pour G'
2. L'état i est construit à partir de I_i. Les actions d'analyse syntaxique pour l'état i sont déterminées comme suit :

[1] PREMIER$(a) = \{a\}$ pour a terminal.

1. Si $A \to \alpha \bullet a\beta$ est dans I_i et si $\text{ALLER\_A}(I_i, a) = I_j$, alors $\text{ACTION}[i, a] = dj$. Dans ce cas, a doit être un terminal.
2. Si $A \to \alpha\bullet$ est dans I_i, alors $\text{ACTION}[i, a] = rj$ où j est le numéro de la règle $A \to \alpha$ pour tout $a \in \text{SUIVANT}(A)$, à l'exception de S'
3. Si $S' \to S\bullet$ est dans I_i, alors $\text{ACTION}[i, \$] = acc$

   Si un conflit entre différentes actions résulte de ces règles, la grammaire n'est pas SLR. L'algorithme ne permet pas dans ce cas la construction d'un analyseur syntaxique.
3. Les transitions de transfert $\text{GOTO}[i, A]$ pour l'état i sont construites pour tout non-terminal A comme suit : si $\text{ALLER\_A}(I_i, A) = I_j$ alors $\text{GOTO}[i, A] = j$
4. Toutes les entrées non remplies par les règles 2 et 3 sont positionnées à `err` (cellules vides)
5. L'état initial est celui construit à partir de l'ensemble d'items contenant $S' \to \bullet S$

8 Analyse LR

L'analyseur se sert de la table SLR pour déterminer la prochaine action en fonction de l'état au sommet de la pile i et du prochain symbole sous la tête de lecture a de la façon suivante :

— Si $\text{ACTION}[i, a] = dj$, où j est un état. L'analyseur effectue un décalage : il empile j et consomme une unité lexicale
— Si $\text{ACTION}[i, a] = rk$, où k est le numéro de la règle $A \to \beta$. L'analyseur effectue une réduction :
  — il dépile $|\beta|$ symboles de la pile
  — l'état l est maintenant au sommet de la pile
  — il empile l'état m, qui correspond à l'entrée $\text{GOTO}[l, A]$
— Si $\text{ACTION}[i, a] =$ `acc`: l'analyseur accepte l'entrée
— Si $\text{ACTION}[i, a] =$ `err`: l'analyseur signale une erreur

Utilisez la table d'analyse pour simuler l'analyse des mots suivants :

— Pour G_1 : $w_1 =$ `( ( ) )` et $w_2 =$ `( ) )`
— Pour G_2 : $w_1 =$ `dv ; df`, $w_2 =$ `dv ; dv` et $w_3 =$ `dv , dv ;`

Détaillez l'état de la pile, de la bande de lecture et l'action de l'analyseur à chaque étape.