Analyse Syntaxique - Annee 2010-2011
Feuille 4

Grammaires LR(0) et SLR(1).

Exercice 4.1
Une grammaire peut-elle être à la fois $LR(0)$ (respectivement $SLR(1)$) et ambiguë ? récursive gauche ? récursive droite ? Justifier.

Exercice 4.2
Soit la grammaire vu au TD3 :

$$\begin{array}{lcl} Expression & \rightarrow & LeftHandSide \text{ '='} RightHandSide \\ & & \mid RightHandSide \\ LeftHandSide & \rightarrow & \text{'*'} ID \\ & & \mid ID \\ RightHandSide & \rightarrow & Expression \\ & & \mid ID \\ & & \mid NUM \end{array}$$

et en annexe, le fichier produit par l'option -v de yacc.

1. Construire l'automate caractéristique des $LR(0)$-items, comme il est décrit dans ce fichier. G est-elle $LR(0)$? $SLR(1)$?
2. Pourquoi yacc ne signale-t-il pas de conflit dans l'état 1, alors qu'il signale un conflit en l'état 10 ?
3. G est-elle ambiguë ?
4. Comment transformer la grammaire G pour éviter le conflit de l'état 10 ?

Exercice 4.3
On considère les grammaires `G1` et `G2` suivantes :

```
grammaire G1:          |       grammaire G2:
                       |
S :  L                 |       S : L
                       |
L :  L ';' L           |       L : 'a' T
  |  A                 |
                       |       T : ';' L
A : 'a'                |         |
```

1. Dire pour chaque grammaire si elle est $LR(0)$, $SLR(1)$? Justifier votre réponse.

2. Si `G1` ou `G2` est $SLR(1)$ construire l'automate des $LR(0)$-items qui lui est associé, puis effectuer une analyse ascendante de la phrase : $a; a; a$ en utilisant une table à trois colonnes : ruban d'entrée, contenu de la pile, action.

Exercice 4.4

On considère la grammaire suivante :

```
L : 'id' ';' L
  | 'id' ';'
```

1. Décrire informellement le langage correspondant.
2. Montrer que cette grammaire n'est pas LR(0) mais qu'elle est SLR(1).
3. Proposer une grammaire équivalente qui est LR(0).
4. En modifiant légèrement cette grammaire, donner une grammaire équivalente qui n'est pas SLR(1).

Annexe

```
0  $accept : Expression $end
1  Expression : LeftHandSide ASSIGNOP RightHandSide
2             | RightHandSide
3  LeftHandSide : '*' ID
4               | ID
5  RightHandSide : Expression
6                | ID
7                | NUM


state 0
        $accept : . Expression $end  (0)

        ID  shift 1
        NUM  shift 2
        '*'  shift 3
        .  error

        Expression  goto 4
        LeftHandSide  goto 5
        RightHandSide  goto 6

state 1
        LeftHandSide : ID .  (4)
        RightHandSide : ID .  (6)

        $end  reduce 6
        ASSIGNOP  reduce 4
```

```
state 2
        RightHandSide : NUM .  (7)

        .  reduce 7

state 3
        LeftHandSide : '*' . ID  (3)

        ID  shift 7
        .  error

state 4
        $accept : Expression . $end  (0)
        RightHandSide : Expression .  (5)

        $end  accept
        .  reduce 5

state 5
        Expression : LeftHandSide . ASSIGNOP RightHandSide  (1)

        ASSIGNOP  shift 8
        .  error

state 6
        Expression : RightHandSide .  (2)

        .  reduce 2

state 7
        LeftHandSide : '*' ID .  (3)

        .  reduce 3

state 8
        Expression : LeftHandSide ASSIGNOP . RightHandSide  (1)

        ID  shift 1
        NUM  shift 2
        '*'  shift 3
        .  error

        Expression  goto 9
        LeftHandSide  goto 5
        RightHandSide  goto 10
```

```
state 9
        RightHandSide : Expression .  (5)

        .  reduce 5


10: reduce/reduce conflict (reduce 1, reduce 2) on $end
state 10
        Expression : LeftHandSide ASSIGNOP RightHandSide .  (1)
        Expression : RightHandSide .  (2)

        .  reduce 1


State 10 contains 1 reduce/reduce conflict.


6 terminals, 4 nonterminals
8 grammar rules, 11 states
```