Travaux Dirigés de Traduction n°4

Licence d'informatique

———

Analyse syntaxique ascendante 2

Le but de ce TD est d'étudier l'analyse ascendante LALR(1).

▸ Exercice 1. *Grammaire de la somme (associativité droite)*

Soit la grammaire

$$\begin{array}{lcl} S & \longrightarrow & E\texttt{<eof>} \\ E & \longrightarrow & T\texttt{<plus>}E \mid T \\ T & \longrightarrow & \texttt{<val>} \end{array}$$

Construire la table d'analyse LR(1) de cette grammaire.

▸ Exercice 2. *Quelle analyse ?*

Soit la grammaire

$$\begin{array}{lcl} S & \longrightarrow & E\texttt{<eof>} \\ E & \longrightarrow & \texttt{<lpar>}E\texttt{<rpar>} \mid \texttt{<star>}E \mid \texttt{<val>} \end{array}$$

Construire les tables d'analyse LR(0) et LR(1) de cette grammaire.

▸ Exercice 3. *LALR(1) et LR(1)*

Soit la grammaire

$$\begin{array}{lcl} S & \longrightarrow & Aa \mid bAc \mid Bc \mid bBa \\ A & \longrightarrow & d \\ B & \longrightarrow & d \end{array}$$

Calculer les tables d'analyse LR(1) et LALR(1) de cette grammaire.

▸ **Exercice 4.** ***Un premier programme bison***

Dans cet exemple, on a écrit le lexer "à la main". Il sera normalement créé via `flef`.

```
%{
  #include<stdio.h>
  #include<ctype.h>
    int yylval ;
    int yyerror(char *msg);
    int yylex();
%}

%token NB PLUS FIN
%left PLUS
%start AXIOME
%%
AXIOME : EXP FIN { printf("%d\n", $1 );return 0}
EXP    : NB { $$ = $1 }
       | EXP PLUS EXP { $$ = $1 + $3 }
       ;
%%

int main( void ) {
  yyparse() ;
  return 0;
}

int yyerror(char *msg) {
  printf("\n%s\n", msg);
  return 0;
}

int yylex(void) {
  int car ;
  car = getchar() ;
  if ( car == EOF ) return 0;
  if ( isdigit(car) ) {
    yylval = car - '0';
    return NB;
  }
  switch ( car ) {
  case '+' : return PLUS;
  case '=': return FIN;
  }
  return -1;
```

```
}
```

Quelle est la grammaire appliquée dans ce parseur ? Calculer la table d'analyse LALR(1) de cette grammaire. Comment son ambiguïté est-elle levée ?