Tutorial 3 LALR(1) Parsing, etc.

Consider the CUP grammar in Appendix 1.

(a) Use the LALR(1) state information in Appendix 2. Construct the action table and goto table from these states.

(b) Perform an LALR(1) parse of the valid input in Appendix 3. Display the information in the same style as chapter 2 of the lecture notes.

(c) Draw the parse tree corresponding to the LALR(1) parse. The root node should correspond to the rule "$START ::= Program $". I would prefer that nodes correspond to grammar rules, rather than terminal and nonterminal symbols.

(d) Draw the abstract syntax tree built by the actions in the CUP grammar.

Appendix 1 CUP grammar for question 1

```
terminal ERROR, LEFT, RIGHT, LEFTBRACE, RIGHTBRACE, ASSIGN, SEMICOLON, COMMA,
CLASS;
terminal String NUMBER;
terminal String IDENT;

nonterminal ProgramNode Program;
nonterminal DeclListNode DeclList;
nonterminal DeclNode Decl;
nonterminal TypeNode Type;
nonterminal DeclrListNode DeclrList;
nonterminal DeclrNode Declr;
nonterminal ExprNode Expr;
nonterminal VariableNode Variable;

start with Program;

Program::=
          DeclList:declList
          {:
          RESULT = new ProgramNode( declList );
          :}
     ;

DeclList::=
          {:
          RESULT = new DeclListNode();
          :}
     |
          DeclList:declList Decl:decl
          {:
          declList.addElement( decl );
          RESULT = declList;
          :}
     ;
```

```
Decl::=
          Type:type DeclrList:declrList SEMICOLON
          {:
          RESULT = new VariableDeclNode( type, declrList );
          :}
     |
          CLASS IDENT:ident LEFTBRACE DeclList:declList RIGHTBRACE
          {:
          RESULT = new ClassDeclNode( ident, declList );
          :}
     |
          error SEMICOLON
          {:
          RESULT = new ErrorDeclNode( "DeclError...;" );
          :}
     |
          error RIGHTBRACE
          {:
          RESULT = new ErrorDeclNode( "DeclError...}" );
          :}
     ;

Type::=
          IDENT:ident
          {:
          RESULT = new TypeIdentNode( ident );
          :}
     ;

DeclrList::=
          Declr:declr
          {:
          RESULT = new DeclrListNode( declr );
          :}
     |
          DeclrList:declrList COMMA Declr:declr
          {:
          declrList.addElement( declr );
          RESULT = declrList;
          :}
     ;

Declr::=
          IDENT:ident ASSIGN Expr:expr
          {:
          RESULT = new InitDeclrNode( ident, expr );
          :}
     |
          IDENT:ident
          {:
          RESULT = new UninitDeclrNode( ident );
          :}
     |
          error
          {:
          RESULT = new ErrorDeclrNode( "DeclrError..." );
          :}
     ;
```

```
Expr::=
          LEFT Expr:expr RIGHT
          {:
          RESULT = expr;
          :}
     |
          NUMBER:value
          {:
          RESULT = new NumberNode( Integer.parseInt( value ) );
          :}
     |
          Variable:variable
          {:
          RESULT = new VariableExprNode( variable );
          :}
     ;

Variable::=
          IDENT:ident
          {:
          RESULT = new VariableNode( ident );
          :}
     ;
```

Appendix 2 LALR(1) State Information

```
===== Rules =====
[0] $START ::= Program EOF
[1] Program ::= DeclList
[2] DeclList ::=
[3] DeclList ::= DeclList Decl
[4] Decl ::= Type DeclrList SEMICOLON
[5] Decl ::= CLASS IDENT LEFTBRACE DeclList RIGHTBRACE
[6] Decl ::= error SEMICOLON
[7] Decl ::= error RIGHTBRACE
[8] Type ::= IDENT
[9] DeclrList ::= Declr
[10] DeclrList ::= DeclrList COMMA Declr
[11] Declr ::= IDENT ASSIGN Expr
[12] Declr ::= IDENT
[13] Declr ::= error
[14] Expr ::= LEFT Expr RIGHT
[15] Expr ::= NUMBER
[16] Expr ::= Variable
[17] Variable ::= IDENT

-------- ACTION_TABLE --------
From state #0
     EOF:REDUCE(rule 2) error:REDUCE(rule 2) CLASS:REDUCE(rule 2)
     IDENT:REDUCE(rule 2)
From state #1
     EOF:SHIFT(state 29)
From state #2
     EOF:REDUCE(rule 1) error:SHIFT(state 4) CLASS:SHIFT(state 5)
     IDENT:SHIFT(state 6)
From state #3
     error:SHIFT(state 14) IDENT:SHIFT(state 15)
From state #4
     RIGHTBRACE:SHIFT(state 13) SEMICOLON:SHIFT(state 12)
From state #5
     IDENT:SHIFT(state 8)
```

```
From state #6
     error:REDUCE(rule 8) IDENT:REDUCE(rule 8)
From state #7
     EOF:REDUCE(rule 3) error:REDUCE(rule 3) RIGHTBRACE:REDUCE(rule 3)
     CLASS:REDUCE(rule 3) IDENT:REDUCE(rule 3)
From state #8
     LEFTBRACE:SHIFT(state 9)
From state #9
     error:REDUCE(rule 2) RIGHTBRACE:REDUCE(rule 2) CLASS:REDUCE(rule 2)
     IDENT:REDUCE(rule 2)
From state #10
     error:SHIFT(state 4) RIGHTBRACE:SHIFT(state 11) CLASS:SHIFT(state 5)
     IDENT:SHIFT(state 6)
From state #11
     EOF:REDUCE(rule 5) error:REDUCE(rule 5) RIGHTBRACE:REDUCE(rule 5)
     CLASS:REDUCE(rule 5) IDENT:REDUCE(rule 5)
From state #12
     EOF:REDUCE(rule 6) error:REDUCE(rule 6) RIGHTBRACE:REDUCE(rule 6)
     CLASS:REDUCE(rule 6) IDENT:REDUCE(rule 6)
From state #13
     EOF:REDUCE(rule 7) error:REDUCE(rule 7) RIGHTBRACE:REDUCE(rule 7)
     CLASS:REDUCE(rule 7) IDENT:REDUCE(rule 7)
From state #14
     SEMICOLON:REDUCE(rule 13) COMMA:REDUCE(rule 13)
From state #15
     ASSIGN:SHIFT(state 21) SEMICOLON:REDUCE(rule 12) COMMA:REDUCE(rule 12)
From state #16
     SEMICOLON:REDUCE(rule 9) COMMA:REDUCE(rule 9)
From state #17
     SEMICOLON:SHIFT(state 19) COMMA:SHIFT(state 18)
From state #18
     error:SHIFT(state 14) IDENT:SHIFT(state 15)
From state #19
     EOF:REDUCE(rule 4) error:REDUCE(rule 4) RIGHTBRACE:REDUCE(rule 4)
     CLASS:REDUCE(rule 4) IDENT:REDUCE(rule 4)
From state #20
     SEMICOLON:REDUCE(rule 10) COMMA:REDUCE(rule 10)
From state #21
     LEFT:SHIFT(state 22) NUMBER:SHIFT(state 25) IDENT:SHIFT(state 26)
From state #22
     LEFT:SHIFT(state 22) NUMBER:SHIFT(state 25) IDENT:SHIFT(state 26)
From state #23
     RIGHT:REDUCE(rule 16) SEMICOLON:REDUCE(rule 16) COMMA:REDUCE(rule 16)
From state #24
     SEMICOLON:REDUCE(rule 11) COMMA:REDUCE(rule 11)
From state #25
     RIGHT:REDUCE(rule 15) SEMICOLON:REDUCE(rule 15) COMMA:REDUCE(rule 15)
From state #26
     RIGHT:REDUCE(rule 17) SEMICOLON:REDUCE(rule 17) COMMA:REDUCE(rule 17)
From state #27
     RIGHT:SHIFT(state 28)
From state #28
     RIGHT:REDUCE(rule 14) SEMICOLON:REDUCE(rule 14) COMMA:REDUCE(rule 14)
From state #29
     EOF:REDUCE(rule 0)
------------------------------
```

```
-------- REDUCE_TABLE --------
From state #0:
     Program:GOTO(1)
     DeclList:GOTO(2)
From state #1:
From state #2:
     Decl:GOTO(7)
     Type:GOTO(3)
From state #3:
     DeclrList:GOTO(17)
     Declr:GOTO(16)
From state #4:
From state #5:
From state #6:
From state #7:
From state #8:
From state #9:
     DeclList:GOTO(10)
From state #10:
     Decl:GOTO(7)
     Type:GOTO(3)
From state #11:
From state #12:
From state #13:
From state #14:
From state #15:
From state #16:
From state #17:
From state #18:
     Declr:GOTO(20)
From state #19:
From state #20:
From state #21:
     Expr:GOTO(24)
     Variable:GOTO(23)
From state #22:
     Expr:GOTO(27)
     Variable:GOTO(23)
From state #23:
From state #24:
From state #25:
From state #26:
From state #27:
From state #28:
From state #29:
-----------------------------
```

Appendix 3 Valid input for question 1(b), (c), (d)

```
class A {
     int a, b = 3;
     int c = ( a );
     }
```

Bruce Hutton