Vinta Reethu
ES18BTECH11028
Compilers - 2
Indian Institute of Technology Hyderabad
Scribe Notes

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Mini Assignment - 3

Vinta Reethu
ES18BTECH11028

5th November 2020

1 YACC :

- Yacc reads the set of grammar rules provided by the user and generates an **LALR(1)** parser for it.
- LALR(1) is derived from the LR(1). LALR(1) takes 1 lookahead token.
- LALR(1) parser is less powerful than the LR(1) parser and more powerful than the SLR(1) parser.
- LALR(1) table will have 2 states i.e: GOTO,ACTION parts.
- In this parsing table the rows are merged, replacing the old names by the new merged names. Just like LR(1) conflicts may also occur in this.

2 Compiling program in YACC :

The input of YACC is set of grammar rules and the output is a C file.

- If you compile **'yacc filename.y'** it gives **y.tab.c** file. **y.tab.c** contains an output file.
- If you compile **'yacc -d filename.y'** it gives **y.tab.c and y.tab.h** files. **y.tab.h** contains definitions for tokens defined.
- If you compile **'yacc -d -v filename.y'** it gives **y.tab.c, y.tab.h and y.output** files. **y.output** contains information on parsing tables and shift,reduction rules.

3 Conflicts in YACC :

There are 3 possible types of conflicts that may occur. They are shift-shift, reduce-reduce and shift-reduce. Shift-shift conflict will **never** occur in parser.

- **Shift - Reduce Conflict :**
  If a state has both shift and reduce action associated with it, then the parser cannot decide between them. In a shift-reduce conflict, YACC's default is to shift.
- **Reduce - Reduce Conflict :**
  If a state has two reduce actions associated with it, then the parser cannot decide the action. In a reduce-reduce conflict, YACC's default is to reduce to the action which is appears first in the rules mentioned.

4 Running YACC on C,C++ programs :

- When the given C grammar is run on yacc a total of **350** states are created in the parsing table.
- **3** accept states are present in C grammar.
- When the given C++ grammar is run on yacc a total of **894** states are created in the parsing table.
- **3** accept states are present in C++ grammar.

5 Conflicts in C grammar :

When we run the given C grammar the terminal reports a warning message which says :

```
warning: 1 shift/reduce conflict [-Wconflicts-sr]
```

The above warning message states that there is a shift-reduce conflict in the given grammar. To see where exactly this conflict occur we can open **y.output** file. When we open y.output file the first line says this :

```
State 333 conflicts: 1 shift/reduce
```

Let us look at state 333 more clearly in our y.output file.

```
192 selection_statement: IF '(' expression ')' statement .
193                    | IF '(' expression ')' statement . ELSE statement

ELSE  shift, and go to state 343

ELSE      [reduce using rule 192 (selection_statement)]
$default  reduce using rule 192 (selection_statement)
```

There are two rules for selection_statement. If the lookahead token is ELSE the the parser has two choices. One is to shift the ELSE and go to 343 state. Other option is to reduce by using the rule 192. We know that parser prefers shift over reduce, hence default action is shift.

Example to observe this conflict:

```
if(Exp1)
    Statement 1
    if(Exp2)
        Statement 2
    else
        Statement 3
```

Let's say we are parsing the above code.
Consider the instance where if(Exp1)Statement 1 if (Exp2) Statement 2 are on the stack and the next input 'else' is seen the parser has two choices
1) **Reduce** : if (Exp2) Statement 2 → selection_statement.(by rule 192)
2) **Shift** the else and reduce it later by
if (Exp2) Statement2 else Statement 3 → selection_statement.
(by rule 193)
Since yacc prefers shift over reduce case-1 occurs.

6 Resolving Conflicts in C grammar:

- **Re-Writing the grammar(Removing the ambiguity):**
  It can clear been seen that the grammar is ambiguous. So we can write the grammar by adding some non-terminals without changing the meaning of the rule to remove ambiguity.

```
selection_statement: Matched_statement | Unmatched_statement
Matched_statement  : IF '(' expression ')' Matched_statement ELSE Matched_statement
                   | Switch Statement
Unmatched_statement: IF '(' expression ')' selection_statement
                   | IF '(' expression ')' Matched_statement ELSE
                     Unmatched_statement
```

- **Using Precedence :** Alternate solution to this is to give precedence the rules. We can give IF-ELSE more precedence than nrml IF statement. To do this we can do the following :

```
// As we go down, precidence increases
%nonassoc IF
%nonassoc ELSE
selection_statement : IF '(' expression ')' statement %prec IF
                    | IF '(' expression ')' statement ELSE statement
```

As we go down precedence increases therefore IF-ELSE has more precedence than IF statement. This will resolve the conflict.

Vinta Reethu
ES18BTECH11028
Compilers - 2
Indian Institute of Technology Hyderabad
Scribe Notes

7 Conflicts in C++ grammar :

The given C++ grammar has no conflicts.

8 Resolving Conflicts in C++ grammar:

There are no conflicts to solve in given C++ grammar.