Akash Tadwai
Department of Engineering Science
Indian Institute of Technology Hyderabad
Email: es18btech11019@iith.ac.in

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Mini-Assignment- III

Akash Tadwai

November 6, 2020

1 YACC Conflicts

There are 3 possible types of conflicts that can 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.

- **NOTE:** When YACC encounters a shift/reduce conflict, it shifts if the token in the input buffer has a greater precedence than the production of the handle on top of the stack and reduces if the production of the handle on top of the stack has a higher precedence than the token. If the production and the token have same precedence, it reduces if the production corresponding to the handle is left associative and shifts if they are right associative. (If the production and token in such a case of equal precedence occur and they both are non associative, YACC reports an error).

1.1 Running YACC on C and C++ Grammars

- **Compilation flags**:
  1. **-d :** If this option is used, the file y.tab.h is generated with the define statements that associate the yacc-assigned 'token codes' with the user-declared 'token names'. This allows source files other than y.tab.c to access the token codes.
  2. **-v**: If this option is given, the file y.output is prepared, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

- Running C grammar produces 350 states and a conflict in state 333 which is an if-else construct.
- Running the given C++ grammar produces 894 states and 0 unresolved conflicts. It was mentioned in the given grammar that 24 conflicts from 10 ambiguities are resolved by 8 %prec's, so that yacc reports 0 conflicts.

1.2 SR conflicts in C Grammar

There is only one Conflict in C grammar that is $if - else$ construct. The conflict is due to the ambigousness of the grammar. The output of yacc after running the C grammar is as follows,

CGrammar.y: warning: 1 shift/reduce conflict [-Wconflicts-sr]

Looking into the *y.output* file the first line states that, the there is a conflict in state 333, which is,

```
State 333

  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)
```

The conflict arises due to not specifying which "if" to match to the "else" statement, i.e, 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 can remove the conflict in one of the following ways:

- Prioritizing shift over reduce (YACC default). This is a good decision as designing Unambigous grammars may not be feasible in all cases. Or giving Precedence to if-else more than if using %prec. We can do it as follows,

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

- Designing Unambigous Grammar as follows,

$$
\begin{aligned}
stmt &\rightarrow matched - stmt \mid unmatched - stmt \\
matched - stmt &\rightarrow if \ (\ expr \ \ ) \ matched - stmt \ else \ matched - stmt \\
matched - stmt &\rightarrow non - alternative \ stmt \\
unmatched - stmt &\rightarrow if \ (\ expr \ \ ) \\
unmatched - stmt &\rightarrow if \ (\ expr \ \ ) \ matched - stmt \ else \ unmatched - stmt
\end{aligned}
$$

- A *matched statement* is
  - either an if-else statement containing no unmatched statements or
  - any statement which is not an if-else statement and not an if statement.
- Then an unmatched statement is: an if statement (with no else-part) or if-else statement where unmatched statements are allowed in the else-part (but not in the if-part).

2 YACC Grammars for Float Binary to Decimal

1. **Method-I**

```
N :   L.R {N.value = L.value + R.value}
    | {}

L :   L1 B {L.value = L1.value*2 + B.value}
    | B {L.value = B.value}

R :   B R1 {R.value = (R1.value+B.value)/2}
    | B    {R.value = B.value/2}

B :   ZERO {B.value=0}
    | ONE  {B.value=1}
```

2. **Method-II**

```
N :   L.R {N.value = L.val + R.val}
    | {}

L :   B L1 {L.len = L1.len +1
            L.val = B.val * 2^{L1.len}}
    | B {L.val = B.val
         L.len = 1}

R :   B R {R.val = (R.val+B.val)/2}
    | B   {R.val = B.val/2}

B :   ZERO {B.val=0}
    | ONE  {B.val=1}
```

3. Method-III

```
N   :  X1.X2 {N.val   = X1.val + X2.val/$2^{X2.len}$}

X   :  BX2   {X.len   = X1.len + 1
     |         X.val=B.val * $2^{X2.len}$ + X2.val}

B :    ZERO {B.val=0}
     | ONE  {B.val=1}
```

LaTeX generated document
************END************