**Name: Chitwan Goel**
**Roll No. 210295**

# CS335 Assignment 2

## 1 Problem 1

Given grammar:

$$Function \rightarrow Type\ \textbf{id}\ (Arguments) \tag{1}$$
$$Type \rightarrow \textbf{id} \tag{2}$$
$$Type \rightarrow Type* \tag{3}$$
$$Arguments \rightarrow ArgList \tag{4}$$
$$Arguments \rightarrow \epsilon \tag{5}$$
$$ArgList \rightarrow Type\ \textbf{id}\ , ArgList \tag{6}$$
$$ArgList \rightarrow Type\ \textbf{id} \tag{7}$$

### 1.1 Why the grammar is not LL(1)?

The given grammar involves left recursion, for example, the productions $Type \rightarrow Type*$ is of the form $A \rightarrow A\alpha$. Also, the grammar is not left factored as it is clear from (6) and (7) productions. Thus, the given grammar is not LL(1).

### 1.2 Transformation to LL(1)

We follow the algorithm to remove indiret as well as direct recursion from the grammar. Let us fix the order of iteration of non-terminals as $Function$, $Type$, $Arguments$, $ArgList$. While processing $Type$, we see that there is direct recursion, so add a new non-terminal $Type'$ and transform as:

$$Type \rightarrow \textbf{id}\ Type'$$
$$Type' \rightarrow *Type' \mid \epsilon$$

Again in processing $ArgList$, there are productions of the form $A_i \rightarrow A_j\,\gamma$ so we transform them also as:

$$ArgList \rightarrow \textbf{id}\ Type'\ \textbf{id}$$

$$ArgList \rightarrow \textbf{id}\ Type'\ \textbf{id}\,,\ ArgList$$

Finally there is scope for left factoring in the above two productions, so we add a new non-terminal, say $B$ and do:

$$ArgList \rightarrow \textbf{id}\ Type'\ \textbf{id}\,B$$
$$B \rightarrow \epsilon \mid ,ArgList$$

Thus, the final transformed LL(1) grammar is:

$$Function \rightarrow Type\ \textbf{id}\ (Arguments) \tag{8}$$
$$Type \rightarrow \textbf{id}\ Type' \tag{9}$$
$$Type' \rightarrow *Type' \tag{10}$$
$$Type' \rightarrow \epsilon \tag{11}$$
$$Arguments \rightarrow ArgList \tag{12}$$
$$Arguments \rightarrow \epsilon \tag{13}$$
$$ArgList \rightarrow \textbf{id}\ \ Type'\ \ \textbf{id}\ B \tag{14}$$
$$B \rightarrow \epsilon \tag{15}$$
$$B \rightarrow ,\ ArgList \tag{16}$$

### 1.3 FIRST and FOLLOW sets

FIRST($Function$) = {**id**}
FIRST($Type$) = {**id**}
FIRST($Type'$) = {$*$, $\epsilon$}
FIRST($Arguments$) = {**id**, $\epsilon$}
FIRST($Arglist$) = {**id**}
FIRST($B$) = {, , $\epsilon$}

FOLLOW($Function$) = {\$}
FOLLOW($Type$) = {**id**}
FOLLOW($Type'$) = {**id**}
FOLLOW($Arguments$) = { ) }
FOLLOW($Arglist$) = { ) }
FOLLOW($B$) = { ) }

### 1.4 LL(1) Parsing Table

| **Non Terminal** | **id** | ( | ) | $*$ | , | \$ |
|---|---|---|---|---|---|---|
| $Function$ | $Function \rightarrow Type\ \textbf{id}\ (Arguments)$ | | | | | |
| $Type$ | $Type \rightarrow \textbf{id}\ Type'$ | | | | | |
| $Type'$ | $Type' \rightarrow \epsilon$ | | | $Type' \rightarrow *Type'$ | | |
| $Arguments$ | $Arguments \rightarrow ArgList$ | | $Arguments \rightarrow \epsilon$ | | | |
| $ArgList$ | $ArgList \rightarrow \textbf{id}\ \ Type'\ \ \textbf{id}\ B$ | | | | | |
| $B$ | | | $B \rightarrow \epsilon$ | | $B \rightarrow ,\ ArgList$ | |

Table 1: LL(1) Parsing Table

# 2 Problem 2

Given Grammar (in augmented form):

$$
\begin{aligned}
0 \quad & S' \to S \\
1 \quad & S \to LM \\
2 \quad & S \to Lp \\
3 \quad & S \to qLr \\
4 \quad & S \to sr \\
5 \quad & S \to qsp \\
6 \quad & L \to aMb \\
7 \quad & L \to s \\
8 \quad & L \to t \\
9 \quad & M \to t
\end{aligned}
$$

## 2.1 FIRST and FOLLOW sets

FIRST$(S)$ = $\{q, s, a, t\}$
FIRST$(L)$ = $\{s, a, t\}$
FIRST$(M)$ = $\{t\}$
FOLLOW$(S)$ = $\{\$\}$
FOLLOW$(L)$ = $\{t, p, r\}$
FOLLOW$(M)$ = $\{\$, b\}$

## 2.2 LR(0) Canonical Collection

We will start with the state $I_0$ and compute LR(0) collection. In all these states, the kernel items will be in blue color and the non-kernel items in black.
$I_0 = \{S' \to \bullet S,\ S \to \bullet LM,\ S \to \bullet Lp,\ S \to \bullet qLr,\ S \to \bullet sr,\ S \to \bullet qsp,\ L \to \bullet aMb,\ L \to \bullet s,\ L \to \bullet t\}$
$GOTO(I_0, S) = \{S' \to S\bullet\} = I_1$
$GOTO(I_0, L) = \{S \to L \bullet M,\ S \to L \bullet p,\ M \to \bullet t\} = I_2$
$GOTO(I_0, q) = \{S \to q \bullet Lr,\ S \to q \bullet sp,\ L \to \bullet aMb,\ L \to \bullet s,\ L \to \bullet t\} = I_3$
$GOTO(I_0, s) = \{S \to s \bullet r,\ L \to s\bullet\} = I_4$
$GOTO(I_0, a) = \{L \to a \bullet Mb,\ M \to \bullet t\} = I_5$
$GOTO(I_0, t) = \{L \to t\bullet\} = I_6$
$GOTO(I_2, M) = \{S \to LM\bullet\} = I_7$
$GOTO(I_2, p) = \{S \to Lp\bullet\} = I_8$
$GOTO(I_2, t) = \{M \to t\bullet\} = I_9$
$GOTO(I_3, L) = \{S \to qL \bullet r\} = I_{10}$
$GOTO(I_3, s) = \{S \to qs \bullet p,\ L \to s\bullet\} = I_{11}$
$GOTO(I_3, a) = \{L \to a \bullet Mb,\ M \to \bullet t\} = I_5$
$GOTO(I_3, t) = \{L \to t\bullet\} = I_6$
$GOTO(I_4, r) = \{S \to sr\bullet\} = I_{12}$
$GOTO(I_5, M) = \{L \to aM \bullet b\} = I_{13}$
$GOTO(I_5, t) = \{M \to t\bullet\} = I_9$

$GOTO(I_{10}, r) = \{S \to qLr\bullet\} = I_{14}$
$GOTO(I_{11}, p) = \{S \to qsp\bullet\} = I_{15}$
$GOTO(I_{13}, b) = \{L \to aMb\bullet\} = I_{16}$

### 2.3 SLR Parsing Table

| **State** | ACTION | | | | | | | | GOTO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | p | q | r | s | a | b | t | $ | S | L | M |
| 0 | | s3 | | s4 | s5 | | s6 | | 1 | 2 | |
| 1 | | | | | | | | accept | | | |
| 2 | s8 | | | | | | s9 | | | | 7 |
| 3 | | | | s11 | s5 | | s6 | | | 10 | |
| 4 | r7 | | r7/s12 | | | | r7 | | | | |
| 5 | | | | | | | s9 | | | | 13 |
| 6 | r8 | | r8 | | | | r8 | | | | |
| 7 | | | | | | | | r1 | | | |
| 8 | | | | | | | | r2 | | | |
| 9 | | | | | | r9 | | r9 | | | |
| 10 | | | s14 | | | | | | | | |
| 11 | s15/r7 | | r7 | | | | r7 | | | | |
| 12 | | | | | | | | r4 | | | |
| 13 | | | | | | s16 | | | | | |
| 14 | | | | | | | | r3 | | | |
| 15 | | | | | | | | r5 | | | |
| 16 | r6 | | r6 | | | | r6 | | | | |

In the SLR table, we see multiple cells having both shift and reduce choice causing shift-reduce conflicts. Thus, the grammar is not SLR(1).

## 2.4 LR(1) Collection

$I_0 = \{[S' \to \bullet S, \$], [S \to \bullet LM, \$], [S \to \bullet Lp, \$], [S \to \bullet qLr, \$], [S \to \bullet sr, \$], [S \to \bullet qsp, \$], [L \to \bullet aMb, t/p], [L \to \bullet s, t/p], [L \to \bullet t, t/p]\}$
$GOTO(I_0, S) = \{[S' \to S\bullet, \$]\} = I_1$
$GOTO(I_0, L) = \{[S \to L \bullet M, \$], [S \to L \bullet p, \$], [M \to \bullet t, \$]\} = I_2$
$GOTO(I_0, q) = \{[S \to q \bullet Lr, \$], [S \to q \bullet sp, \$], [L \to \bullet aMb, r], [L \to \bullet s, r], [L \to \bullet t, r]\} = I_3$
$GOTO(I_0, s) = \{[S \to s \bullet r, \$], [L \to s\bullet, t/p]\} = I_4$
$GOTO(I_0, a) = \{[L \to a \bullet Mb, t/p], [M \to \bullet t, b]\} = I_5$
$GOTO(I_0, t) = \{[L \to t\bullet, t/p]\} = I_6$
$GOTO(I_2, M) = \{[S \to LM\bullet, \$]\} = I_7$
$GOTO(I_2, p) = \{[S \to Lp\bullet, \$]\} = I_8$
$GOTO(I_2, t) = \{[M \to t\bullet, \$]\} = I_9$
$GOTO(I_3, L) = \{[S \to qL \bullet r, \$]\} = I_{10}$
$GOTO(I_3, s) = \{[S \to qs \bullet p, \$], [L \to s\bullet, r]\} = I_{11}$
$GOTO(I_3, a) = \{[L \to a \bullet Mb, r], [M \to \bullet t, b]\} = I_{12}$
$GOTO(I_3, t) = \{[L \to t\bullet, r]\} = I_{13}$
$GOTO(I_4, r) = \{[S \to sr\bullet, \$]\} = I_{14}$
$GOTO(I_5, M) = \{[L \to aM \bullet b, t/p]\} = I_{15}$
$GOTO(I_5, t) = \{[M \to t\bullet, b]\} = I_{16}$
$GOTO(I_{10}, r) = \{[S \to qLr\bullet, \$]\} = I_{17}$
$GOTO(I_{11}, p) = \{[S \to qsp\bullet, \$]\} = I_{18}$
$GOTO(I_{12}, M) = \{[L \to aM \bullet b, r]\} = I_{19}$
$GOTO(I_{12}, t) = \{[M \to t\bullet, b]\} = I_{16}$
$GOTO(I_{15}, b) = \{[L \to aMb\bullet, t/p]\} = I_{20}$
$GOTO(I_{19}, b) = \{[L \to aMb\bullet, r]\} = I_{21}$

## 2.5 LALR(1) Collection

The states whose cores are same are merged into a single state.

$I_0 = \{[S' \to \bullet S, \$], [S \to \bullet LM, \$], [S \to \bullet Lp, \$], [S \to \bullet qLr, \$], [S \to \bullet sr, \$], [S \to \bullet qsp, \$], [L \to \bullet aMb, t/p], [L \to \bullet s, t/p], [L \to \bullet t, t/p]\}$
$GOTO(I_0, S) = \{[S' \to S\bullet, \$]\} = I_1$
$GOTO(I_0, L) = \{[S \to L \bullet M, \$], [S \to L \bullet p, \$], [M \to \bullet t\$]\} = I_2$
$GOTO(I_0, q) = \{[S \to q \bullet Lr, \$], [S \to q \bullet sp, \$], [L \to \bullet aMb, r], [L \to \bullet s, r], [L \to \bullet t, r]\} = I_3$
$GOTO(I_0, s) = \{[S \to s \bullet r, \$], [L \to s\bullet, t/p]\} = I_4$
$GOTO(I_0, a) = \{[L \to a \bullet Mb, t/p/r], [M \to \bullet t, b]\} = I_{5-12}$
$GOTO(I_0, t) = \{[L \to t\bullet, t/p/r]\} = I_{6-13}$
$GOTO(I_2, M) = \{[S \to LM\bullet, \$]\} = I_7$
$GOTO(I_2, p) = \{[S \to Lp\bullet, \$]\} = I_8$
$GOTO(I_2, t) = \{[M \to t\bullet, \$/b]\} = I_{9-16}$
$GOTO(I_3, L) = \{[S \to qL \bullet r, \$]\} = I_{10}$
$GOTO(I_3, s) = \{[S \to qs \bullet p, \$], [L \to s\bullet, r]\} = I_{11}$
$GOTO(I_3, a) = \{[L \to a \bullet Mb, t/p/r], [M \to \bullet t, b]\} = I_{5-12}$
$GOTO(I_3, t) = \{[L \to t\bullet, t/p/r]\} = I_{6-13}$
$GOTO(I_4, r) = \{[S \to sr\bullet, \$]\} = I_{14}$

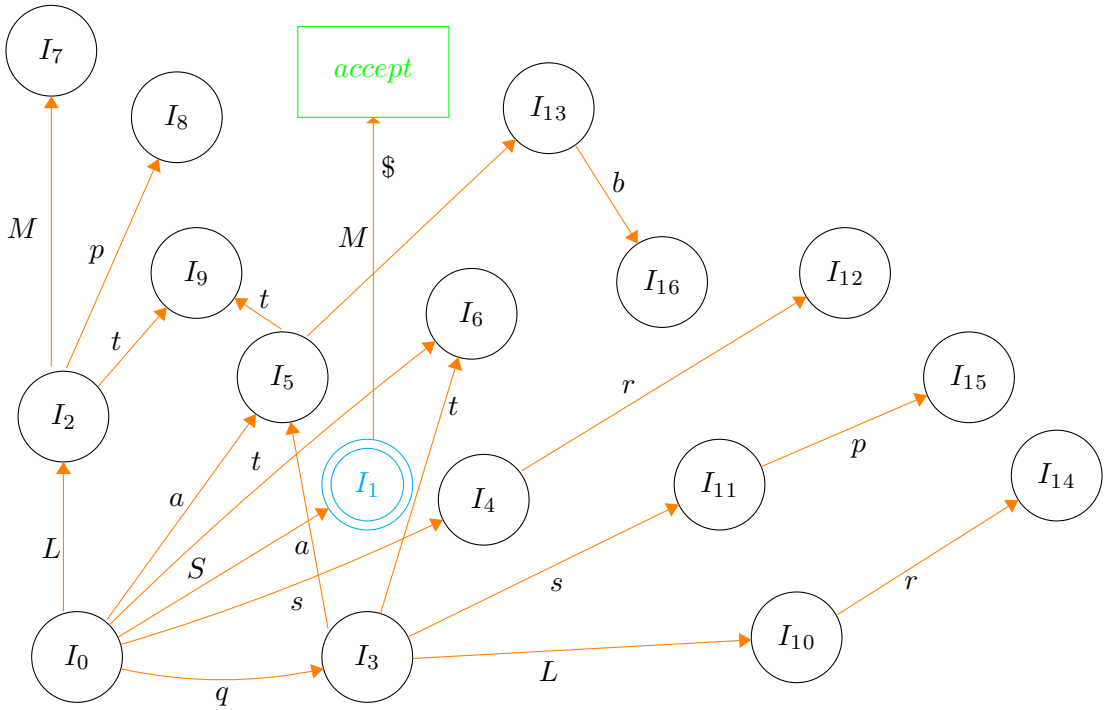
$GOTO(I_{5-12}, M) = \{[L \rightarrow aM \bullet b,\ t/p/r\,]\} = I_{15-19}$
$GOTO(I_5, t) = \{[M \rightarrow t\bullet,\ b\,]\} = I_{9-16}$
$GOTO(I_{10}, r) = \{[S \rightarrow qLr\bullet, \$]\} = I_{17}$
$GOTO(I_{11}, p) = \{[S \rightarrow qsp\bullet, \$]\} = I_{18}$
$GOTO(I_{15-19}, b) = \{[L \rightarrow aMb\bullet,\ t/p/r\,]\} = I_{20-21}$

### 2.6 LALR(1) Parsing Table

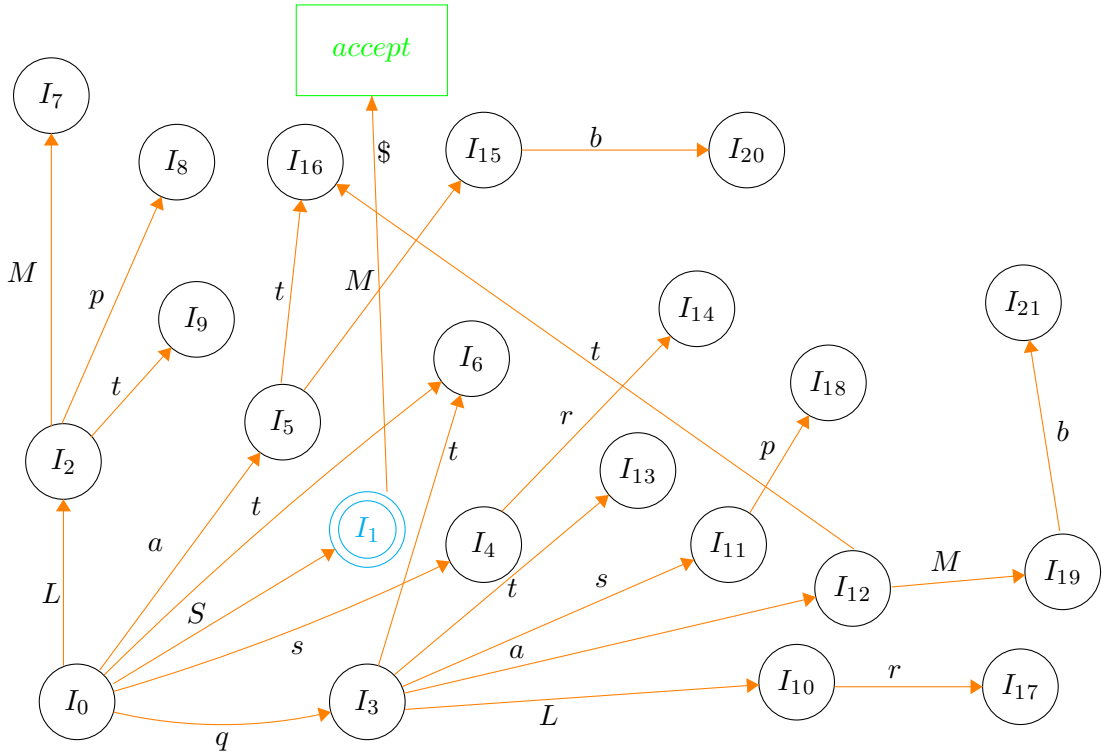
| **State** | ACTION | | | | | | | | GOTO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | p | q | r | s | a | b | t | $ | S | L | M |
| 0 | | s3 | | s4 | s5 | | s6 | | 1 | 2 | |
| 1 | | | | | | | | accept | | | |
| 2 | s8 | | | | | | s9-16 | | | | 7 |
| 3 | | | | s11 | s5-12 | | s6-13 | | | 10 | |
| 4 | r7 | | s14 | | | | r7 | | | | |
| 5-12 | | | | | | | s9-16 | | | | 15-19 |
| 6-13 | r8 | | r8 | | | | r8 | | | | |
| 7 | | | | | | | | r1 | | | |
| 8 | | | | | | | | r2 | | | |
| 9-16 | | | | | | | | r9 | | | |
| 10 | | | s17 | | | | | | | | |
| 11 | s18 | | r7 | | | | r7 | | | | |
| 14 | | | | | | | | r4 | | | |
| 15-19 | | | | | | s20-21 | | | | | |
| 17 | | | | | | | | r3 | | | |
| 18 | | | | | | | | r5 | | | |
| 20-21 | r6 | | r6 | | | | r6 | | | | |

In the LALR(1) table, every cell has a unique entry. Thus, the grammar is LALR(1).

## 2.7 LR(0) Automaton




## 2.8 LR(1) Automaton



# 3 Problem 3

## 3.1 File Structure

Inside the directory *problem3*, there are 2 files *lexer.l* and *parser.y* along with a script *run.sh*

## 3.2 Compilation Instructions

All commands to be executed inside *problem3* directory

```
$ bison -d parser.y
$ flex lexer.l
$ g++ parser.tab.c parser.tab.h lex.yy.c -o parser
$ ./parser < $[path_to_input_file]
```

Alternatively, the bash script *run.sh* can be executed in the following mannner to print the output on the screen.

```
$ chmod +x run.sh
$ ./run.sh $[path_to_input_file]
```

NOTE: The errors are printed to stderr. So, if directing the output to some file, please direct everything to it.

## 3.3 Error handling

- The parser will stop at the point it detects the first error. It will first print the error message and then the statistics of the input it has parsed till that point.
- The following error cases have been detected-
    1. The marks of singleselect question should be either 1 or 2 and multiselect can be in the range 2-8. Error will be thrown for any input violating these ranges of marks. The error message will contain the tag (<singleselect> or <multiselect>) along with the line number at which that question started.
    2. The choices for any question must be 3 or 4. For any cases violating this condition, error will be thrown showing the tag and the line number of start of the question.
    3. The number of correct answers have to be 1 in singleselect and <= choices in multiselect. They can be 0 also in multiselect. Error similar to previous cases will be thrown in other cases.
    4. <singleselect> or <multiselect> tags not following their schema (marks not enclosed in quotes or other) will be thrown as invalid syntax error at that particular line. Whether they are following the hierarchy of tags or not will not be considered and the error will be invalid syntax.
    5. It is assumed that the number of <quiz> tags can only be 1. Error will be thrown at the point the 2nd quiz tag is detected in between the opening and closing of the first quiz tag.
    6. Before the first <quiz> is opened and after it is closed, everything in the input is ignored, whether it is syntactically correct or not.
    7. If any closing tag is found at places where the current open tag does not correspond to it, stray closing < /tag> is thrown.
    8. Opening <singleselect> or <multiselect> : If such a tag is detected inside a <choice> or <correct> tag, then invalid hierarchy error will be thrown. Also, if such a tag is detected inside a <singleselect> or <multiselect> tag, then missing closing < /singleselect> or < /multiselect> is thrown.

9. Opening <choice> or <correct> : If such a tag is detected inside a <choice> or <correct> tag, then missing closing <choice> or <correct> is thrown. Also, if such a tag is detected outside any question, invalid hierarchy error is thrown.
10. If some tag is opened till the end of program, missing closing tag error will be thrown.

### 3.4 Corner cases

1. The marks for a question, count of total and categorical (marks-wise or single/multiselect-wise) questions will be increased only when a question is completely detected, i.e., opening and closing tags both in proper hierarchy. However, even if there are errors in the count of marks, choice or correct options, the fields are incremented (said on Piazza).

2. Choices, correct count is incremented whenever a choice (or correct) block is completed, i.e., opening and closing tags both.