‘Syntax Directed Translator Design’

FOR

Dr. Ramprasad S. Joshi
(Department of Computer Science)

BY

Group 3

Name of the Student	ID Number
Neel Shashikant Bhandari	***2018B1A70084G***
Dev Churiwala	***2018B1A70602G***
Taher Yunus Lilywala	***2018B1A70609G***
Sonakshi Gupta	***2018B1A70614G***
Nilay Pradeep Rajderkar	***2018B3A70725G***

In the partial fulfillment of the requirements of
CS F363: COMPILER CONSTRUCTION

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
27 April 2022

Table of Contents

Changes in the Pattern Action table:

Pattern	**Action**
START	Start Game
MOVE	Move statement
LEFT	Move Left
RIGHT	Move Right
ROTATE	Rotate Block
CL	Rotate Clockwise
ACL	Rotate Anticlockwise
PAUSE	Pause Game
RESTART	Restart Game
DROP	Drop the block fast
BLOCK	Choose Blocks mode
EASY	Blocks Easy

Pattern	**Action**
MED	Blocks Medium
HARD	Blocks Hard
SPEED	Set Block Speed
MODE	Set Game Mode
CB	Mode Colorblind
CC	In-game sound control
NEXTQ	Upcoming blocks
ON	Toggle On
OFF	Toggle Off
SENSITIVITY	Select Movement Sensitivity
NUMBER	Literal Numeric String

Syntax Directed Translation Scheme and Accompanying LR Automaton:

SLY uses a parsing technique known as LR-parsing or shift-reduce parsing. LR parsing is a bottom up technique that tries to recognize the right-hand-side of various grammar rules. Whenever a valid right-hand-side is found in the input, the appropriate action method is triggered and the grammar symbols on right hand side are replaced by the grammar symbol on the left-hand-side.

Consolidated Grammar:

Main Commands

```
command     : DROP
            | droplr
            | RESTART
            | PAUSE
            | START
            | nextq
            | sensitivity
            | speed
            | block
            | rotate
            | mode
             move
```

Higher Order Grammar

```
move        : MOVE move_param

mode        : MODE mode_param

rotate      : ROTATE rotate_param

block       : BLOCK block_param

speed       : SPEED NUMBER

sensitivity : SENSITIVITY NUMBER

droplr      : DROP move_param
```

```
nextq          : NEXTQ nextq_param
```

Parameters

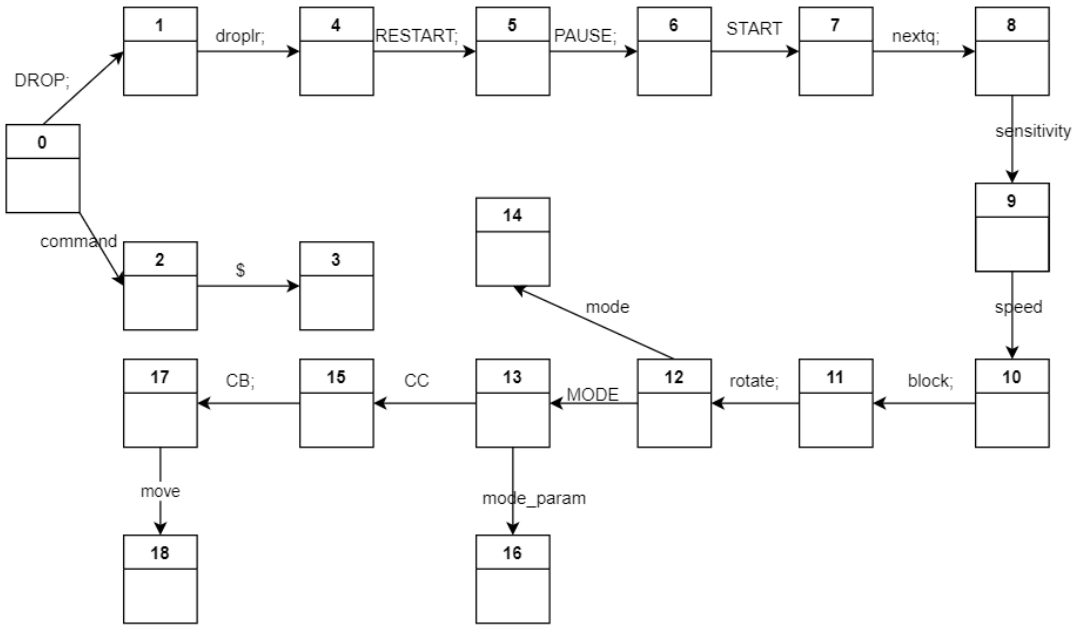
```
move_param     : RIGHT
               | LEFT

rotate_param   : ACL
               | CL

block_param    : HARD
               | MED
               | EASY

mode_param     : CC
               | CB

nextq_param    : OFF
               | ON
```

State Table:

State	Item set
0	{S' ::= • command $, command ::= • DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; move}
1	{command ::= DROP; • droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; move}
2	{S' ::= command • $}
3	{S' ::= command $ • }
4	{command ::= DROP; droplr; • RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; move}
5	{command ::= DROP; droplr; RESTART; • PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; move}
6	{command ::= DROP; droplr; RESTART; PAUSE; • START; nextq; sensitivity; speed; block; rotate; mode; move}
7	{command ::= DROP; droplr; RESTART; PAUSE; START; • nextq; sensitivity; speed; block; rotate; mode; move}
8	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; • sensitivity; speed; block; rotate; mode; move}
9	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; • speed; block; rotate; mode; move}
10	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; • block; rotate; mode; move}
11	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; • rotate; mode; move}
12	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; • mode; move}
13	pplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; • move, move ::
14	{move ::= MOVE • move_param, move_param ::= • RIGHT; LEFT}
15	{command ::= DROP; droplr; RESTART; PAUSE; START; nextq; sensitivity; speed; block; rotate; mode; move • }
16	{move_param ::= RIGHT; • LEFT}
17	{move ::= MOVE move_param • }
18	{move_param ::= RIGHT; LEFT • }

Transition Table: The Transition Table was too large to fit into the document, so it is available here to view as a Google Sheet.

LR Automaton:



An Explanation of Challenges and Parser Conflicts and Error Productions:

A parser constructed from a grammar augmented by error productions detects common and anticipated errors when an error production is used during parsing. The parser can then generate appropriate error diagnostics about the erroneous construct that has been recognized in the input. However, as of now, our design doesn't employ error productions.

```
def error(self, p):
    """
    Syntax error handler.
    Checks for syntax errors i.e. incorrect token and prints location and incorrect token.
    """
    if not p:
        print("Tetris > Syntax Error Encountered.")
        return NULL

    print(f"Tetris > Line {p.lineno}.{p.index+1}: Syntax Error: \"{p.value}\"")
```

```
Tetris > SPEED 4
Parser > ('command', ('speed', 4))
Tetris > SPEED
Tetris > Syntax Error Encountered.
```

We have constructed an error rule in the parser as a syntax error handler. This will check for
syntax errors i.e. incorrect token and prints location and incorrect token. We can identify in our game programming language where a syntax error and corresponding token occurs if any do exist.

Test Cases:

End-to-end tetris game engine programming toolchain:

Workflow Diagram: