Compiler Report

Group 27

Aymeric Baud **아이메릭 보드** (50221602)
Yuhui XU **유회 쉬** (50221565)

Participants:	Date: 06/2023
• Aymeric Baud 아이메릭 보드 (50221602) • Yuhui XU 유회 쉬 (50221565)	

Problem

Implement a bottom-up syntax analyzer based on CFG's table and SLR grammar as given by the project document.

Implement the syntax analyzer for a simplified Java programming language with the following context free grammar G.

Our Solution

We chose Python to construct our program:

- First of all, we had to find the connection and rules from CFG table
- Using the TLR table to redesign a table of action rules
-

Discard ambiguity from CFG Table

We got the CFG table as below:

CFG G:

01: CODE → VDECL CODE | FDECL CODE | CDECL CODE| ϵ
02: VDECL → vtype id semi | vtype ASSIGN semi
03: ASSIGN → id assign RHS
04: RHS → EXPR | literal | character | boolstr
05: EXPR → EXPR addsub EXPR | EXPR multdiv EXPR
06: EXPR → lparen EXPR rparen | id | num
07: FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
08: ARG → vtype id MOREARGS | ϵ
09: MOREARGS → comma vtype id MOREARGS | ϵ
10: BLOCK → STMT BLOCK | ϵ
11: STMT → VDECL | ASSIGN semi
12: STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE
13: STMT → while lparen COND rparen lbrace BLOCK rbrace
14: COND → COND comp COND | boolstr
15: ELSE → else lbrace BLOCK rbrace | ϵ
16: RETURN → return RHS semi
17: CDECL → class id lbrace ODECL rbrace

17: CDECL → class id lbrace ODECL rbrace

18: ODECL → VDECL ODECL | FDECL ODECL | ϵ

We had to modify some rules so that the algorithm could be better established:

CFG G:

00: S -> CODE

01: CODE → VDECL CODE | FDECL CODE | CDECL CODE| ϵ

02: VDECL → vtype id semi | vtype ASSIGN semi

03: ASSIGN → id assign RHS

04: RHS → EXPR | literal | character | boolstr

~~05: EXPR → EXPR addsub EXPR | EXPR multdiv EXPR~~

~~06: EXPR → lparen EXPR rparen | id | num~~

05: EXPR -> TERM addsub EXPR I TERM

06: TERM -> FACTOR multidiv TERM I FACTOR

07: FACTOR -> lparen EXPR rparen I id I nul

07: FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace

08: ARG → vtype id MOREARGS | ϵ

09: MOREARGS → comma vtype id MOREARGS | ϵ

10: BLOCK → STMT BLOCK | ϵ

11: STMT → VDECL | ASSIGN semi

12: STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE

13: STMT → while lparen COND rparen lbrace BLOCK rbrace

14: COND → COND comp COND | boolstr

15: COND -> FACTOR comp FACTOR I boolstr

15: ELSE → else lbrace BLOCK rbrace | ϵ

16: RETURN → return RHS semi

17: CDECL → class id lbrace ODECL rbrace

18: ODECL → VDECL ODECL | FDECL ODECL | ϵ

The program will begin from 00, and 05/06/15 are the place where we could get infinite loops because of their expression, our aim to modify these equivocal expressions is to run the check loop more efficiently.

SLR Table

We constructed a SLR parsing table for the non-ambiguous CFG through the following website:

S -> CODE
CODE -> VDECL CODE
CODE -> FDECL CODE
CODE -> CDECL CODE
CODE -> epsilon
VDECL -> vtype id semi
VDECL -> vtype ASSIGN semi
ASSIGN -> id assign RHS
RHS -> EXPR
RHS -> literal
RHS -> character
RHS -> boolstr
EXPR -> TERM addsub EXPR
EXPR -> TERM
TERM -> FACTOR multdiv TERM
TERM -> FACTOR
FACTOR -> lparen EXPR rparen
FACTOR -> id
FACTOR -> num
FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace

ARG -> vtype id MOREARGS
ARG -> epsilon
MOREARGS -> comma vtype id MOREARGS
MOREARGS -> epsilon
BLOCK -> STMT BLOCK
BLOCK -> epsilon
STMT -> VDECL
STMT -> ASSIGN semi
STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE
STMT -> while lparen COND rparen lbrace BLOCK rbrace
COND -> FACTOR comp FACTOR
COND -> boolstr
ELSE -> else lbrace BLOCK rbrace
ELSE -> epsilon
RETURN -> return RHS semi
CDECL -> class id lbrace ODECL rbrace
ODECL -> VDECL ODECL
ODECL -> FDECL ODECL
ODECL -> epsilon

By using a table from internet (http://jsmachines.sourceforge.net/machines/slr.html - which had been indicate in the project document), we built a new rule table from CFG and SLR table as below:

State	ACTION																				GOTO															
	vtype	id	Semi	assign	literal	character	boolstr	addsub	multdiv	lparen	rparen	num	lbrace	rbrace	comma	if	while	comp	else	return	class	$	S	CODE	VDECL	ASSIGN	RHS	EXPR	TERM	FACTOR	FDECL	ARG	MOREARGS	BLOCK	STMT	COND	ELSE	RETURN	CDECL	ODECL
0	S5																			S6	R4		1	2						3								4	
1																					acc																			
2	S5																			S6	R4		7	2						3								4	
3	S5																			S6	R4		8	2						3								4	
4	S5																			S6	R4		9	2						3								4	
5		S10																							11															
6		S12																																						
7																					R1																			
8																					R2																			
9																					R3																			
10			S13	S15						S14																														
11			S16																																					
12													S17																										
13	R5	R5												R5		R5	R5			R5	R5	R5																		
14	S19										R21																					18								
15		S28			S22	S23	S24			S27		S29														20	21	25	26											
16	R6	R6												R6		R6	R6			R6	R6	R6																		
17	S5													R38											31						32								30	
18											S33																												
19		S34																																					
20			R7																																				
21			R8																																				
22			R9																																				
23			R10																																				
24			R11																																				
25			R13					S35			R13																												
26			R15					R15	S36		R15																												
27		S28								S27		S29																											
28			R17					R17	R17		R17							R17																					
29			R18					R18	R18		R18							R18																					
30														S38																									
31	S5													R38											31						32								39	
32	S5													R38											31						32								40	
33													S41																										
34											R23				S43																		42							
35		S28								S27		S29															44	25	26											
36		S28								S27		S29																45	26											
37											S46																												
38	R35																			R35	R35																		
39														R36																									
40														R37																									
41	S53	S54												R25		S51	S52			R25					49	50								47	48					
42											R20																												
43	S55																																						
44			R12								R12																												
45			R14					R14			R14																												
46			R16					R16	R16		R16							R16																					
47																				S57																			
48	S53	S54												R25		S51	S52			R25					49	50								58	48					
49	R26	R26												R26		R26	R26			R26																			
50			S59																																				
51										S60																												
52										S61																												
53		S62																							11														
54				S15																																			
55		S63																																				
56														S64																									
57		S28			S22	S23	S24			S27		S29														65	21	25	26											
58														R24						R24																			
59	R27	R27												R27		R27	R27			R27																			
60		S28						S68		S27		S29																	67								66			
61		S28						S68		S27		S29																	67								69			
62			S13	S15																																			
63											R23				S43																		70						
64	R19													R19							R19	R19																	
65			S71																																				
66											S72																												
67																		S73																					
68											R31																												
69											S74																												
70											R22																												
71														R34																									
72													S75																										
73		S28								S27		S29																	76											
74													S77																										
75	S53	S54												R25		S51	S52			R25					49	50								78	48					
76											R30																												
77	S53	S54												R25		S51	S52			R25					49	50								79	48					
78														S80																									
79														S81																									
80	R33	R33												R33		R33	R33		S83	R33																	82		
81	R29	R29												R29		R29	R29			R29																			
82	R28	R28												R28		R28	R28			R28																			

83													S84																											
84	S53	S54												R25		S51	S52			R25					49	50								85	48					
85														S86																										
86	R32	R32												R32		R32	R32			R32																				

Implementation

1: Definition of SLR table rule:

```
LINKED_GROUP = [
    ['S', 'CODE'],
    ['CODE', 'VDECL CODE'],
    ['CODE', 'FDECL CODE'],
    ['CODE', 'CDECL CODE'],
    ['CODE', 'epsilon'],
    ['VDECL', 'vtype id semi'],
    ['VDECL', 'vtype ASSIGN semi'],
    ['ASSIGN', 'id assign RHS'],
    ['RHS', 'EXPR'],
    ['RHS', 'literal'],
    ['RHS', 'character'],
    ['RHS', 'boolstr'],
    ['EXPR', 'TERM addsub EXPR'],
    ['EXPR', 'TERM'],
    ['TERM', 'FACTOR multdiv TERM'],
    ['TERM', 'FACTOR'],
    ['FACTOR', 'lparen EXPR rparen'],
    ['FACTOR', 'id'],
    ['FACTOR', 'num'],
    ['FDECL', 'vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace'],
    ['ARG', 'vtype id MOREARGS'],
    ['ARG', 'epsilon'],
    ['MOREARGS', 'comma vtype id MOREARGS'],
    ['MOREARGS', 'epsilon'],
    ['BLOCK', 'STMT BLOCK'],
    ['BLOCK', 'epsilon'],
    ['STMT', 'VDECL'],
    ['STMT', 'ASSIGN semi'],
    ['STMT', 'if lparen COND rparen lbrace BLOCK rbrace ELSE'],
    ['STMT', 'while lparen COND rparen lbrace BLOCK rbrace'],
    ['COND', 'FACTOR comp FACTOR'],
    ['COND', 'boolstr'],
    ['ELSE', 'else lbrace BLOCK rbrace'],
    ["ELSE', 'epsilon"],
    ['RETURN', 'return RHS semi'],
    ['CDECL', 'class id lbrace ODECL rbrace'],
    ['ODECL', 'VDECL ODECL'],
    ['ODECL', 'FDECL ODECL'],
    ['ODECL', 'epsilon']
]
```

2: Creating of SLR parse table

```
SLR_PARSE = [
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 1, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'$': 'ACC', },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 7, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 8, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 9, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'id': 'S10', 'ASSIGN': 11, },
    {'id': 'S12', },
    {'$': 'R1', },
    {'$': 'R2', },
    {'$': 'R3', },
    {'semi': 'S13', 'assign': 'S15', 'lparen': 'S14', },
    {'semi': 'S16', },
    {'lbrace': 'S17', },
    {'vtype': 'R5', 'id': 'R5', 'rbrace': 'R5', 'if': 'R5', 'while': 'R5', 'return': 'R5', 'class': 'R5', '$': 'R5', },
    {'vtype': 'S19', 'rparen': 'R21', 'ARG': 18, },
    {'id': 'S28', 'literal': 'S22', 'character': 'S23', 'boolstr': 'S24', 'lparen': 'S27', 'num': 'S29', 'RHS': 20, 'EXPR': 21, 'TERM': 25, 'FACTOR': 26, },
    {'vtype': 'R6', 'id': 'R6', 'rbrace': 'R6', 'if': 'R6', 'while': 'R6', 'return': 'R6', 'class': 'R6', '$': 'R6', },
    {'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 30, },
    {'rparen': 'S33', },
    {'id': 'S34', },
    {'semi': 'R7', },
    {'semi': 'R8', },
    {'semi': 'R9', },
    {'semi': 'R10', },
    {'semi': 'R11', },
    {'semi': 'R13', 'addsub': 'S35', 'rparen': 'R13', },
    {'semi': 'R15', 'addsub': 'R15', 'multdiv': 'S36', 'rparen': 'R15', },
    {'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'EXPR': 37, 'TERM': 25, 'FACTOR': 26, },
    {'semi': 'R17', 'addsub': 'R17', 'multdiv': 'R17', 'rparen': 'R17', 'comp': 'R17', },
```

```
{'semi': 'R18', 'addsub': 'R18', 'multdiv': 'R18', 'rparen': 'R18', 'comp': 'R18', },
{'rbrace': 'S38', },
{'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 39, },
{'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 40, },
{'lbrace': 'S41', },
{'rparen': 'R23', 'comma': 'S43', 'MOREARGS': 42, },
{'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'EXPR': 44, 'TERM': 25, 'FACTOR': 26, },
{'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'TERM': 45, 'FACTOR': 26, },
{'rparen': 'S46', },
{'vtype': 'R35', 'class': 'R35', '$': 'R35', },
{'rbrace': 'R36', },
{'rbrace': 'R37', },
{'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49, 'ASSIGN': 50, 'BLOCK': 47, 'STMT': 48, },
{'rparen': 'R20', },
{'vtype': 'S55', },
{'semi': 'R12', 'rparen': 'R12', },
{'semi': 'R14', 'addsub': 'R14', 'rparen': 'R14', },
{'semi': 'R16', 'addsub': 'R16', 'multdiv': 'R16', 'rparen': 'R16', 'comp': 'R16', },
{'return': 'S57', 'RETURN': 56, },
```



```
{'return': 'S57', 'RETURN': 56, },
{'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49, 'ASSIGN': 50, 'BLOCK': 58, 'STMT': 48, },
{'vtype': 'R26', 'id': 'R26', 'rbrace': 'R26', 'if': 'R26', 'while': 'R26', 'return': 'R26', },
{'semi': 'S59', },
{'lparen': 'S60', },
{'lparen': 'S61', },
{'id': 'S62', 'ASSIGN': 11, },
{'assign': 'S15', },
{'id': 'S63', },
{'rbrace': 'S64', },
{'id': 'S28', 'literal': 'S22', 'character': 'S23', 'boolstr': 'S24', 'lparen': 'S27', 'num': 'S29', 'RHS': 65, 'EXPR': 21, 'TERM': 25, 'FACTOR': 26, },
{'rbrace': 'R24', 'return': 'R24', },
{'vtype': 'R27', 'id': 'R27', 'rbrace': 'R27', 'if': 'R27', 'while': 'R27', 'return': 'R27', },
{'id': 'S28', 'boolstr': 'S68', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 67, 'COND': 66, },
{'id': 'S28', 'boolstr': 'S68', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 67, 'COND': 69, },
{'semi': 'S13', 'assign': 'S15', },
{'rparen': 'R23', 'comma': 'S43', 'MOREARGS': 70, },
{'vtype': 'R19', 'rbrace': 'R19', 'class': 'R19', '$': 'R19', },
{'semi': 'S71', },
{'rparen': 'S72', },
{'comp': 'S73', },
{'rparen': 'R31', },
{'rparen': 'S74', },
{'rparen': 'R22', },
{'rbrace': 'R34', },
{'lbrace': 'S75', },
{'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 76, },
{'lbrace': 'S77', },
{'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49, 'ASSIGN': 50, 'BLOCK': 78, 'STMT': 48, },
{'rparen': 'R30', },
{'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49, 'ASSIGN': 50, 'BLOCK': 79, 'STMT': 48, },
{'rbrace': 'S80', },
{'rbrace': 'S81', },
{'vtype': 'R33', 'id': 'R33', 'rbrace': 'R33', 'if': 'R33', 'while': 'R33', 'else': 'S83', 'return': 'R33', 'ELSE': 82, },
{'vtype': 'R29', 'id': 'R29', 'rbrace': 'R29', 'if': 'R29', 'while': 'R29', 'return': 'R29', },
{'vtype': 'R28', 'id': 'R28', 'rbrace': 'R28', 'if': 'R28', 'while': 'R28', 'return': 'R28', },
{'lbrace': 'S84', },
{'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49, 'ASSIGN': 50, 'BLOCK': 85, 'STMT': 48, },
{'rbrace': 'S86', },
{'vtype': 'R32', 'id': 'R32', 'rbrace': 'R32', 'if': 'R32', 'while': 'R32', 'return': 'R32', },
]
```


3: SLR Stack

```python
def syntax_parse(symbols: list, index = 0) -> Tuple[Status, str]:
    if not symbols:
        return Status.ERROR, "no symbols found"

    if len(symbols) == 1:
        return Status.SUCCESS, "ok"
```

Created syntax_parse function, we check whether the list of symbols is empty or return ok when their's only one symbol at first to saving the time in certain situations.

```python
    stack = [0] # State stack
    error_line = 1

    while True:
```

```
        next_one = symbols[index]
        current = stack[-1] # get the last element from list
```

Here we defined stack, error_line, current and next_one variable. error_line is for memorizing the error line to throw a error message at the end. current is the last element from stack which will be used in the check loop.

4: Shift

When the first character of the command of SLR parse table gives us is S, it means we need to switch

```
178             if (action_type == 'S'):
179                 index += 1
180                 error_line += 1
181                 stack.append(action_value)
```

Here is the error management, here return the error status and the error message which made by the 'error_line' and the error content as information

The print status function helps us to output the error or succeed message as soon as we return a value from the syntax parse function.

```
171             if next_one not in SLR_PARSE[current].keys():
172                 error = f"symbol {next_one}, line {error_line}"
173                 return Status.ERROR, error
```

```
143 class Status(Enum):
144     SUCCESS = 0,
145     ERROR = 1
146 
147 def print_status(status: Status, message: str) -> None:
148     print(f"{status.name.lower()}: {message}")
```

5: Reduce and TOGO

```
        elif (action_type == 'R'):
            non_termi, termi = LINKED_GROUP[action_value][:2]
            termi_length = len(str(termi).split(" "))

            for i in range(termi_length):
```

```
            if termi != 'epsilon':
                stack.pop()
                symbols.pop(index - i - 1)

        if SLR_PARSE[stack[-1]].get(non_termi) is None:
            error = f"symbol {next_one}, line {error_line}"
            return Status.ERROR, error
        stack.append(SLR_PARSE[stack[-1]][non_termi])
        if termi != 'epsilon':
            index -= termi_length - 1
        else:
            index += 1

        symbols.insert(index - 1, non_termi)
```

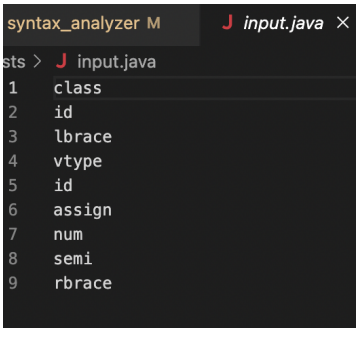
This part is to the process to reduce the terminal or the non-terminal situation to non-terminal.

6: Input process

```
150    def get_lines(file_path: str) -> List:
151        with open(file_path) as file:
152            return [line.rstrip() for line in file]
```

Experiment

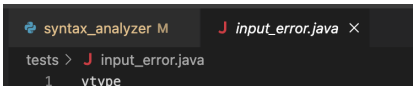
1 - Test file which should give us a positive result:

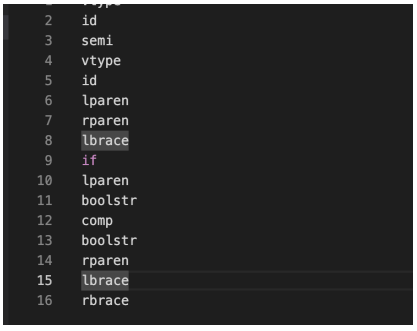


Result that our program returns:

```
clara@ClaradeMacBook-Air syntax-analyzer % python3 syntax_analyzer tests/input.java
success: OK!
```

2 - Test file which should give us an error message:

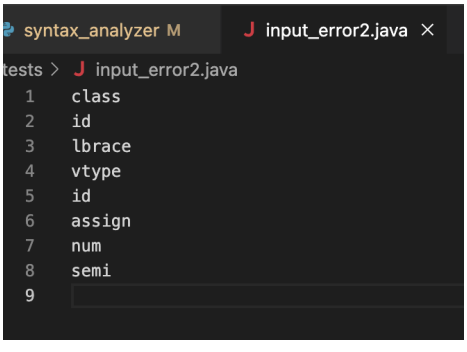




Result that our program returns:

```
clara@ClaradeMacBook-Air syntax-analyzer % python3 syntax_analyzer tests/input_error.java
error: symbol comp, line 12
```

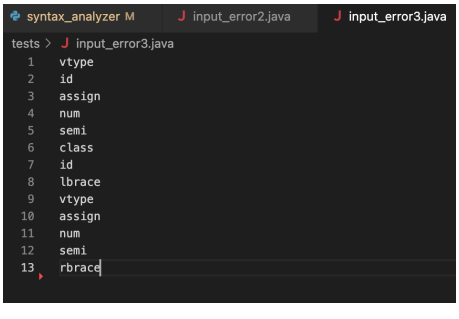
3 - Test file which should give us an error message as there's a empty line at the end of the file:



Result that our program returns: (command $ because we add '$' at the end of the each file to mark that this is the last character that we need to read)

```
clara@ClaradeMacBook-Air syntax-analyzer % python3 syntax_analyzer tests/input_error2.java
error: symbol $, line 9
```

4 - Test file which should give us an error message:



Result that our program returns

```
clara@ClaradeMacBook-Air syntax-analyzer % python3 syntax_analyzer tests/input_error3.java
error: symbol assign, line 10
```