

Term Project Report

SLR Parser

과 목 Subject	24-1 COMPILER 03 컴파일러 03 분반
담당 교수 Professor	김호수
소속 Department	중앙대학교 소프트웨어대학 소프트웨어학부
팀 Team	Team 23
팀원 Team Member	20204946 이규성 20202203 박호근

Table of Contents

<i>Abstract / 개요</i>	<i>- 3 -</i>
<i>NON-Ambiguous CFG / 모호하지 않은 CFG 제작.....</i>	<i>- 4 -</i>
Change Things 기존 CFG 대비 변경사항	- 5 -
<i>SLR parsing table.....</i>	<i>- 6 -</i>
<i>All Implementation</i>	<i>- 17 -</i>
main.py	- 17 -
data.py	- 21 -
html_to_txt.py	- 22 -
<i>Test Cases.....</i>	<i>- 23 -</i>
Random Test Case	- 24 -
Example_maker.py.....	- 24 -
Static Test Case	- 28 -
<i>Executable Binary File 실행 가능 파일 사용법.....</i>	<i>- 32 -</i>

Abstract | 개요

해당 보고서는 24-1 CAU CSE Compiler 03분반의 Term Project, SLR Parser를 만드는 과정에 대해 기술합니다. 프로젝트 문서에서 기술된 CFG를 다음의 사이트에서 활용할 수 있도록 문법을 분해 및 변경하여 사용하였습니다. (<https://jsmachines.sourceforge.net/machines/slr.html>)

해당 사이트에서 만들어진 Table을 파싱하여, 데이터로 활용할 수 있도록 변경하는 data.py와, 그 데이터를 활용해 실제 SLR parsing을 진행하는 main.py 두 개의 스크립트로 구성됩니다. 또한 파싱을 위해 사용되는 raw data인 parse table html 스크립트와, reduce할 때 정보로 활용할 grammar.txt에 우리 팀이 수정한 CFG가 존재합니다.

마지막으로, 테스트를 위해 random case를 만들어내는 example_maker.py 가 존재하고, random_test.sh를 통해 랜덤 케이스를 만들고, 그것을 main.py가 잘 수행하는 지 확인하는 쉘 스크립트가 존재합니다. Static한 케이스들에 대해 반복적인 테스트를 위해 static 케이스들 역시 테스트하는 static_test.sh도 존재합니다.

해당 파이썬 및 쉘 스크립트는 리눅스 환경에서 동작하며, 자세한 개발환경 및 실행을 위해 필요한 내용은 우리 팀의 Github Repository의 README.md를 참고해주시길 바랍니다.

<https://github.com/Tastypotato245/24-1-CAU-CSE-Compiler-Term-Project-SLR-parser>

NON-Ambiguous CFG | 모호하지 않은 CFG 제작

"는 입실론을 의미함.

```
CODE -> CODE_D
CODE_D -> FDECL CODE_D
CODE_D -> VDECL CODE_D
CODE_D -> ''
VDECL -> VTYPE id ;
VDECL -> VTYPE ASSIGN ;
ASSIGN -> id = RHS
RHS -> EXPR
RHS -> literal
RHS -> character
RHS -> BOOLSTR
EXPR -> EXPR_D ADDSUB EXPR
EXPR -> EXPR_D
EXPR_D -> EXPR_DD MULDIV EXPR_D
EXPR_D -> EXPR_DD
EXPR_DD -> ( EXPR )
EXPR_DD -> id
EXPR_DD -> num
FDECL -> VTYPE id ( ARG ) { BLOCK RETURN }
ARG -> VTYPE id MOREARGS
ARG -> ''
MOREARGS -> , VTYPE id MOREARGS
MOREARGS -> ''
BLOCK -> STMT BLOCK
BLOCK -> ''
STMT -> VDECL
STMT -> ASSIGN ;
STMT -> if ( COND ) { BLOCK } ELSE
STMT -> while ( COND ) { BLOCK }
COND -> BOOLSTR COMP COND
COND -> BOOLSTR
ELSE -> else { BLOCK }
ELSE -> ''
RETURN -> return RHS ;
ADDSUB -> +
ADDSUB -> -
MULDIV -> *
MULDIV -> /
COMP -> ==
COMP -> !=
BOOLSTR -> true
BOOLSTR -> false
VTYPE -> int
VTYPE -> float
VTYPE -> char
```

Change Things | 기존 CFG 대비 변경사항

1. Start Symbol 부분 변경
 - a. 기존의 문법을 SLR 테이블을 생성해주는 사이트에서 이용하기 위해, 가장 첫 번째 Start Symbol을 하나로 통일했습니다.
 - b. 또한 | 로 구분된 규칙을 모두 한 라인 당 하나의 생성 규칙이 되도록 분리하였습니다.
2. Ambiguous 해결 (1)
 - a. EXPR의 경우, ADDSUB와 MULTDIV로 변경될 수 있는 부분을, EXPR_D 와 EXPR_DD 라는 심볼을 추가해 Depth를 주어, Ambiguous Tree가 나오지 않도록 수정하였습니다.
3. Ambiguous 해결 (2)
 - a. COND의 경우, COND가 항상 COMP 뒤에 등장하고, COND자체가 BOOLSTR로 변경 될 수 있도록 하여, 기존의 문법과 동일한 언어를 인식하되, 모호성을 없앴습니다.
4. Terminal 일부 추가 및 변경
 - a. VTPYE이나, ADDSUB, MULTDIV, COMP, BOOLSTR의 경우, 실제 terminal 기호로 보여 지는 것이 좋아 추가적으로 문법을 완성했습니다.
 - b. 또한 SEMI나 PARAN, BRACE역시 ; () { }로 일대일 대응되기에 치환하여 작성하였습니다.
5. 입실론 규칙을 "으로 변경
 - a. 사이트를 이용하기 위해, "를 입실론 규칙 대신 사용하였습니다.

SLR parsing table

[illegible]

[illegible]

[illegible]

LR table																																									
State	ACTION															GOTO																									
	id	;	=	literal	()	un	{	}	,	if	while	return	+-*/	=!	=	rules	fail	ignore	char	\$	CODE	DECL	ASSIGN	EXPR	EXPR	EXPR	FD	ARG	MORE	BLOCK	STMT	COND	ED	SUB	MD	COMP	BO	VT		
2	s				s	s																			3	2	2														
7	2				2	2																			8	3	6														
8	r				r									r	r	r	r																								
	1				1									1	1	1	1																								
	6				6									6	6	6	6																								
2	r				r									r	r	r	r																								
9	1				1									1	1	1	1																								
	7				7									7	7	7	7																								
3					s																																				
0					3																																				
					9																																				
3					r						s																														
1					2						4																		40												
					2						1																														
3	s				s	s																																			
2	2				2	2																																			
8					7	9																																			
3	r				r	r																																			
3	3				3	3																																			
	4				4	4																																			
3	r				r	r																																			
4	3				3	3																																			
	5				5	5																																			
3	s				s	s																																			

[illegible]

LR table																																							
State	ACTION															GOTO																							
	id	=	literal	()	un	{	}	,	if	while	return	+-	-	*	/	=	!	rule	failure	float	char	\$	CODE	DECL	ASSIGN	EXPR	EXPR	EXPR	DECL	ARG	MORE	BLOCK	STMT	COLS	RETURN	VTYP		
44	r				r								r	r	r	r																							
44	1				1								1	1	1	1																							
	5				5								5	5	5	5																							
45												s																											
45												5																											
												5																											
46	s					r	s	s		r																													
46	5					2	4	5		2										s	s	s		4	4													5	
46	2					4	9	0		4										5	6	7		7	8													1	
47	r					r	r	r		r											r	r	r																
47	2					2	2	2		2										2	2	2																	
47	5					5	5	5		5										5	5	5																	
48	s																																						
48	5																																						
48	7																																						
49					s																																		
49					5																																		
49					8																																		
50																																							
50																																							
50																																							
50																																							
50																																							
50											</																												

[illegible]

LR table																																														
	ACTION															GOTO																														
State	d	=	:	()	{	}	,	'	if	while	return	+	-	*	/	=	!	=	!	return	fail	continue	\$	CODE	CODE	DECL	ASSIGN	EXPR	EXPR	EXPR	FD	ARG	MORE	BLOCK	STMT	COLN	RETURN	ADDSUB	MULDIV	COMP	COLSTR	VTYP			
0	3	4																																												
61				r			s																										67													
62																						r	r	r	r																					
63	s	6	8																																											
64				s																																										
65				r													s	s																										7	0	
66				s																																										
67				r																																										
68							r																																							

[illegible]

LR table																																																	
	ACTION															GOTO																																	
State	id	=	literal	()	un	{	}	,	if	while	return	+-	*	/	=	!	=	rule	failure	float	char	\$	CODE	DECL	ASSIGN	EXPR	EXPR	EXPR	FD	ARG	MORE	BLOCK	STMT	COLS	ED	UNDER	AD	DU	COMP	BO	COL	TYPE						
7						79																																											
78						80																																											
79	r32					r32	r33	r33	83	r33										r33	r33	r33																											
80	r28					r28	r28	r28		r28										r28	r28	r28																											
81	r27					r27	r27	r27		r27										r27	r27	r27																											
82						83																																											
83	s52					r24	s54	s54		r24										s55	s56	s57			47	48									84	46										51			
84						85																																											
8	r3					r3	r33	r33		r3										r3	r3	r3																											

LR table																																		
S t a t e	ACTION																	GOTO																
	i	d	;	=	:	l	c	t	h	a	r	()	{	}	,	'	n																
5	1													1		1	1		1															

Doc 디렉터리 내에 .html 파일을 추가로 첨부하였습니다. 참고 바랍니다.

All Implementation

main.py

```
import sys
from data import rules, parse_table

# 출력을 위한 트리 노드
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []

    def add_child(self, child_node):
        self.children.append(child_node)

# 부모에서 자식 방향으로 깊이 우선 탐색하며 출력함.
def print_tree(node, prefix="", is_last=True, is_root=True):
    # 루트 노드 출력
    if is_root:
        print(node.value)
    else:
        print(prefix + "|--" + str(node.value))

    # 새로운 prefix 계산
    new_prefix = prefix + ("    " if is_last else "|    ")

    # 자식 노드들 출력
    for i, child in enumerate(node.children):
        print_tree(child, new_prefix, i == len(node.children) - 1, is_root=False)

# 파서 부분임
class SLRParser:
    def __init__(self, parse_table, rules):
        self.parse_table = parse_table
        self.rules = rules

        self.statestack = [] # 상태를 관리할 스택임
        self.statestack.append(0)

        self.inputStack = [] # input 을 넣으며 실제로 reduce 하는 데 사용하는 스택임
        self.inputStack.append('$')

        self.nodequeue = [] # tree 를 관리할 스택

    # def value_exists(self, current_state, current_token: str):
    #     try:
    #         value = self.parse_table[current_state][current_token]
    #         return True
    #     except KeyError:
    #         return False
    #     except IndexError:
    #         return False

    def parse(self, tokens):
        current_state = 0
        token_index = 0
        nodes = []
        child_nodes = []

        # 내부에서 False 또는 True 로 리턴될거라 루프 돌림
        while True:
            # print('step')
            current_state = self.statestack[-1]
            current_token = tokens[token_index]
```

```

try:
    # 일단 파스 테이블에 현재 상태에서의 현재 토큰일 때의 action 이 있는지를 봄
    action = self.parse_table[current_state][current_token]
except KeyError:
    # 없으면 오류임. 이상한 토큰이 나온 것
    print(f"Reject at state {current_state}, unexpected token: {current_token}")
    self.statestack = []
    self.statestack.append(0)
    self.inputStack = []
    self.inputStack.append('$')
    self.nodequeue = []
    return False
except IndexError:
    # 없으면 오류임. 이상한 토큰이 나온 것
    print(f"Reject at state {current_state}, unexpected token: {current_token}")
    self.statestack = []
    self.statestack.append(0)
    self.inputStack = []
    self.inputStack.append('$')
    self.nodequeue = []
    return False
# print(f'current state: {current_state}')
# print(f'current token: {current_token}')
# print(f'action : {action}')

if action.startswith('s'):
    # action 이 s 일 때는 shift 를 함.
    # inputStack 에 현재 토큰을 push 함
    self.inputStack.append(current_token)
    # 트리에 출력에 활용하기 위해 큐에 붙임
    # 트리 구성을 위해 shift 의 경우 queue 배열에 append 함
    self.nodequeue.append(TreeNode(current_token))
    # 상태 스택에 이제 해야하는 action 을 push 함
    self.statestack.append(int(action[1:]))
    token_index += 1
elif action.startswith('r'):
    # action 이 r 일 때는 reduce 를 함.
    node = TreeNode(self.rules[int(action[1:])[0])
    # reduce 할 노드의 갯수를 받아 queue 길이에서 빼줌
    # reduce 의 경우 뒤에 추가된 노드가 우선적으로 reduce 되지만 input 의 순서 상
    # reduce 되는 노드들의 경우 먼저 추가된 노드가 먼저 pop 해야함
    # 이 때문에 reduce 가 시작되는 위치에서 pop 을 시작
    pop = len(self.nodequeue) - self.rules[int(action[1:])[1]
    # grammar 인 rules 에서 그것의 길이만큼 빼줘야함. 그만큼 reduce 된 것이니.
    for _ in range(self.rules[int(action[1:])[1]):
        self.statestack.pop()
        # print(self.rules[int(action[1:])[1])
        # print(f'debug : {int(action[1:])}')
        # print(f'pop:{self.inputStack.pop()}')
        # 트리에 pop 한 노드 추가
        # reduce 될 노드의 개수만큼 for loop
        node.add_child(self.nodequeue.pop(pop_))
    # reduce 된 노드를 queue 에 append
    self.nodequeue.append(node)
    self.inputStack.append(self.rules[int(action[1:])[0])
    # 현재 상태와 토큰을 업데이트 해줌 (top 에 있는 거로 최신화 해주는 것임)
    current_state = self.statestack[-1]
    current_token = self.inputStack[-1]
    try:
        # 파스 테이블에 현재 상태에서의 현재 토큰일 때의 action 이 있는지를 봄 (위에 처음
        # 시도한거랑 같음. goto 해야해서 한 번 여기서 하는 거)

```

```

        action = self.parse_table[current_state][current_token]
    except KeyError:
        print(f"Reject at state {current_state}, unexpected token:
{current_token}")
        self.statestack = []
        self.statestack.append(0)
        self.inputStack = []
        self.inputStack.append('$')
        self.nodequeue = []
        return False
    except IndexError:
        print(f"Reject at state {current_state}, unexpected token:
{current_token}")
        self.statestack = []
        self.statestack.append(0)
        self.inputStack = []
        self.inputStack.append('$')
        self.nodequeue = []
        return False

    # goto case 임
    self.statestack.append(int(action))
elif action == 'acc':
    # accept 할 때임.
    # accept 의 경우 현재 node queue 에 있는 node 가 root
    print_tree(node)
    self.statestack = []
    self.statestack.append(0)
    self.inputStack = []
    self.inputStack.append('$')
    self.nodequeue = []
    return True
else:
    print(f"Reject at state {current_state}, unexpected token: {current_token}")
    self.statestack = []
    self.statestack.append(0)
    self.inputStack = []
    self.inputStack.append('$')
    self.nodequeue = []
    return False

# print(f'state stack: {self.statestack}')
# print(f'input stack: {self.inputStack}')
# print()

def main():
    parser = SLRParser(parse_table, rules)
    if len(sys.argv) < 2:
        # 예외처리하기
        print("Usage: team23_kyuho_slr_parser <input_file>")
        return

    input_file = sys.argv[1]

    with open(input_file, 'r') as file:
        lines = file.readlines()

    line_index = 0
    # 한 라인 단위로 파서를 돌림. 즉, 한 라인에 한 케이스(CODE 로 REDUCE 되는. 한 언어의 문장)가 올 수
    # 있음.
    for line in lines:
        line_index += 1
        tokens = line.strip().split()
        tokens.append("$")
        # print(tokens)
        if parser.parse(tokens):
            # 쉘 스크립트에서 이 스트링을 검출하기 위해 앞에 prefix 가 있는 RESULT 를 출력함.
            print("HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT")
        else:
            print(f"line : {line_index}")

```

```
# 쉘 스크립트에서 이 스트링을 검출하기 위해 앞에 prefix 가 있는 RESULT 를 출력함.  
print("HOGEUN_KYUSUNG_SLR_PARSER RESULT: REJECT")  
  
if __name__ == "__main__":  
    main()
```

main.py 에서는, data.py 를 참조하여 그 내부에 있는 grammer 와 table 을 활용하고 있습니다. 아래는 data.py 입니다.

data.py

```
# 문법 파싱해서 규칙 데이터 만들기
def parse_grammar(grammar_file):
    rules = []
    with open(grammar_file, 'r') as file:
        for line in file:
            line = line.strip()
            if '->' in line:
                left, right = line.split('->')
                left = left.strip()
                right = right.strip()
                symbols = right.split()
                if right == '\\\\':
                    rule_length = 0
                else:
                    rule_length = len(symbols)
                rules.append((left, rule_length))
    return rules

# 테이블 파싱해서 딕셔너리 형태로 만들기
def parse_parse_table(parse_table_file):
    parse_table = []
    with open(parse_table_file, 'r') as file:
        for line in file:
            line = line.strip()
            if line.startswith('parse_table = ['):
                continue
            if line.startswith(']'):
                break
            if line.endswith(','):
                line = line[:-1]
            if line.startswith('{') and line.endswith('}'):
                row = eval(line)
                parse_table.append(row)
    return parse_table

grammar_file = './data/grammar.txt'
parse_table_file = './data/parse_table.txt'

rules = parse_grammar(grammar_file)
parse_table = parse_parse_table(parse_table_file)

if __name__ == "__main__":
    print("Rules:")
    for rule in rules:
        print(f"({rule[0]}', {rule[1]})")

    print("\nParse Table:")
    for row in parse_table:
        print(row)
```

data.py 에서 사용하는 파일은, 결국 html raw data 에서 한 번 가공이 된 파일입니다. 아래 코드에서는 beautifulsoup 을 이용해 html 을 table 변수에 담아 파일을 만듭니다. 이렇게 한 번

html_to_txt.py 를 거치는 이유는, 혹시나 data.py 에서 활용하는 형태가 달라지는 것이 이 스크립트를 수정해야하는 의존성을 없애기 위함입니다. 일단 눈에 보기 쉬운 꼴로 한 번 table 파일을 만들고, 그것을 data.py 에서 사용하도록 했습니다. 아래는 html_to_txt.py 입니다.

html_to_txt.py

```
from bs4 import BeautifulSoup

with open("../data/SLRtable.html", "r", encoding="utf-8") as file:
    html_content = file.read()

soup = BeautifulSoup(html_content, 'html.parser')
lr_table_view_div = soup.find('div', id='lrTableView')

if lr_table_view_div:
    th_tags = lr_table_view_div.find_all('th')
    td_tags = lr_table_view_div.find_all('td')

    symbol = []
    table = []

    check_symbol = False
    for th in th_tags:
        if check_symbol:
            symbol.append(th.text.strip())
        if th.text.strip() == "GOTO":
            check_symbol = True

    for i, td in enumerate(td_tags):
        command = td.text.strip()
        if i % (len(symbol) + 1) == 0:
            table.append({})
        elif command == "":
            continue
        elif command.isdigit():
            table[-1][symbol[i % (len(symbol) + 1) - 1]] = int(command)
        elif command[0] == 's':
            table[-1][symbol[i % (len(symbol) + 1) - 1]] = 's' +
command[1:]
        elif command[0] == 'r':
            table[-1][symbol[i % (len(symbol) + 1) - 1]] = 'r' +
command[1:]
        elif command == "acc":
            table[-1][symbol[i % (len(symbol) + 1) - 1]] = 'acc'

    with open("../table_data/parse_table.txt", "w", encoding="utf-8") as
file:
        file.write("parse_table = [\n")
        for state in table:
            file.write("    {")
            for symbol, action in state.items():
                file.write(f"'{symbol}': '{action}', ")
            file.write("},\n")
        file.write("]\n")
```

Test Cases

테스트는 총 세 가지 방식으로 진행할 수 있습니다. Random case를 만들고 테스트하는 random_test.sh과 static case들을 테스트하는 static_test.sh 셸 스크립트가 있습니다. 랜덤의 경우 어떻게 만들어지는지 설명하며, static의 경우 어떤 input case들을 대표적으로 테스트하였는지 설명합니다.

```
Total Passed Cases: 39993
Total Accepted Cases: 19993
Total Rejected Cases: 20000
Total Pass Percentage: 100.083%
Total Accept Percentage: 50.0325%
Total Reject Percentage: 50.0501%

Pass Percentage per File:
rand_example/accept_01.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_02.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_03.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_04.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_05.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_06.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_07.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_08.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_09.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_10.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_11.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_12.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_13.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_14.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_15.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_16.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_17.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_18.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/accept_19.test: Passed: [#####] 100%, Accepted: [#####] 100%, Rejected: [##] 0%
rand_example/accept_20.test: Passed: [#####] 100.1%, Accepted: [#####] 100.1%, Rejected: [##] 0%
rand_example/reject_01.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_02.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_03.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_04.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_05.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_06.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_07.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_08.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_09.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_10.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_11.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_12.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_13.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_14.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_15.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_16.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_17.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_18.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_19.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
rand_example/reject_20.test: Passed: [#####] 100.1%, Accepted: [##] 0%, Rejected: [#####] 100.1%
+ src git:(main) *
```

Random Test Case

랜덤 테스트는 ./random_test.sh 이라는 shell script로 작동합니다. 해당 스크립트가 볼릴 때, 아래의 example_maker.py라는 파이썬 스크립트가 실행되는데, 이는 랜덤하게 CODE에서 하단으로 depth를 따져가며 다양한 example을 만듭니다. 이 example_maker는 accept와 reject 케이스 둘 다 만들 수 있는데, 원리는 다음과 같습니다.

Accept case : 항상 해당 CFG에서 인식 가능한 케이스만을 말단으로 가지고 있습니다. 따라서 depth가 될 때까지 다양한 grammar생성 규칙에 의해 확장되다가, 마지막에는 무조건 terminal로 정리되도록 짰기 때문에, 이론 상 Accept 언어만 나올 수 있습니다.

Reject case : 항상 해당 CFG에서 REJECT하는 케이스만을 무조건 포함하도록 마지막에 검사하여 포함시킵니다. 따라서 무조건 REJECT되는 케이스만 나옵니다.

Example_maker.py

```
import os
import random

grammar = {
    'CODE': ['CODE_D'],
    'CODE_D': ['VDECL CODE_D', 'FDECL CODE_D', ''],
    'VDECL': ['VTYPE id ;', 'VTYPE ASSIGN ;'],
    'ASSIGN': ['id = RHS'],
    'RHS': ['EXPR', 'literal', 'character', 'BOOLSTR'],
    'EXPR': ['EXPR_D ADDSUB EXPR', 'EXPR_D'],
    'EXPR_D': ['EXPR_DD MULTDIV EXPR_D', 'EXPR_DD'],
    'EXPR_DD': ['( EXPR )', 'id', 'num'],
    'FDECL': ['VTYPE id ( ARG ) { BLOCK RETURN }'],
    'ARG': ['VTYPE id MOREARGS', ''],
    'MOREARGS': ['VTYPE id MOREARGS', ''],
    'BLOCK': ['STMT BLOCK', ''],
    'STMT': ['VDECL', 'ASSIGN ;', 'if ( COND ) { BLOCK } ELSE', 'while ( COND )
{ BLOCK }'],
    'COND': ['BOOLSTR COMP COND', 'BOOLSTR'],
    'ELSE': ['else { BLOCK }', ''],
    'RETURN': ['return RHS ;'],
    'VTYPE': ['int', 'float', 'char'],
    'ADDSUB': ['+', '-'],
    'MULTDIV': ['*', '/'],
    'COMP': ['==', '!='],
    'BOOLSTR': ['true', 'false']
}

# accpet 에 대한 최대 깊이 도달 시 사용하는 우선순위 배열. terminal 을 보장하기 위해서...
example_grammar_accept = {
    'CODE': [''],
    'CODE_D': [''],
    'VDECL': ['int id ;', 'char id = literal ;', 'char id = character ;', 'int id =
true ;', 'int id = false ;'],
    'ASSIGN': ['id = literal', 'id = character', 'id = true', 'id = false'],
    'RHS': ['literal', 'character', 'true', 'false'],
    'EXPR': ['num'],
    'EXPR_D': ['num'],
    'EXPR_DD': ['num'],
    'FDECL': ['char id ( ) { return character ; }', 'int id ( ) { return
true ; }'],
    'ARG': ['int id'],
    'MOREARGS': ['float id'],
    'BLOCK': ['int id ;'],
    'STMT': ['int id ;'],
```



```

    'COND': ['true', 'false'],
    'ELSE': ['else { }'],
    'RETURN': ['return literal ;', 'return character ;', 'return true ;', 'return
false ;'],
    'VTYPE': ['int', 'float', 'char'],
    'ADDSUB': ['+', '-'],
    'MULTDIV': ['*', '/'],
    'COMP': ['==', '!='],
    'BOOLSTR': ['true', 'false']
}

# 터미널 기호. validation 을 위해
terminals = [';', 'id', 'literal', 'character', 'num', 'true', 'false', 'int',
'float', 'char', '+', '-', '*', '/', '==', '!=', '(', ')', '{', '}', 'else',
'return']

# reject 에 대한 최대 깊이 도달 시 사용하는 우선순위 배열
example_grammar_reject = {
    'CODE': ['!~'],
    'CODE D': ['}~'],
    'VDECL': ['char id literal ;'],
    'ASSIGN': ['id == true'],
    'RHS': ['rhsererror'],
    'EXPR': ['exprerror'],
    'EXPR_D': ['exprerror'],
    'EXPR_DD': ['exprerror'],
    'FDECL': ['char ) return character ; }'],
    'ARG': ['int id id'],
    'MOREARGS': ['float id ;'],
    'BLOCK': ['int id num ;'],
    'STMT': ['int num num = id ;'],
    'COND': ['?'],
    'ELSE': ['else ()'],
    'RETURN': ['false return ;'],
    'VTYPE': ['isanghantype'],
    'ADDSUB': ['+~-'],
    'MULTDIV': ['*.../'],
    'COMP': ['==='],
    'BOOLSTR': ['idonno']
}

# 랜덤 언어 생성 함수 accept cases 만 생성함.
def generate_random_language(symbol, depth, max_depth):
    if depth >= max_depth:
        expansion = random.choice(example_grammar_accept[symbol])
    else:
        expansion = random.choice(grammar[symbol])

    result = []
    for token in expansion.split():
        if token in grammar:
            result.append(generate_random_language(token, depth + 1, max_depth))
        else:
            result.append(token)

    return ' '.join(result)

# 혹시 몰라 만든 검증 함수
def validate_terminal_end(language):
    tokens = language.split()
    return all(token in terminals for token in tokens[-1:])

# 리젝트 케이스를 생성하는 함수

```

```

def generate_reject_language(symbol, depth, max_depth,
used_example_grammar_reject):
    if depth >= max_depth:
        expansion = random.choice(example_grammar_reject[symbol])
        used_example_grammar_reject[0] = True
    else:
        expansion = random.choice(grammar[symbol])

    result = []
    for token in expansion.split():
        if token in grammar:
            result.append(generate_reject_language(token, depth + 1, max_depth,
used_example_grammar_reject))
        else:
            if random.random() < 0.98:
                result.append(token)
            else:
                # 랜덤한 잘못된 토큰을 삽입
                result.append(random.choice(['wrong', 'error', 'invalid',
'unknown']))

    return ' '.join(result)

# 리젝트 케이스에 example_grammar_reject 의 내용을 추가하는 함수
def ensure_reject_case_uses_example_grammar(case):
    parts = case.split()
    for key, expansions in example_grammar_reject.items():
        for expansion in expansions:
            if expansion in case:
                return case # 이미 사용된 경우

    # 사용되지 않은 경우 추가
    key = random.choice(list(example_grammar_reject.keys()))
    expansion = random.choice(example_grammar_reject[key])
    parts.append(expansion)
    return ' '.join(parts)

os.makedirs('rand_example', exist_ok=True)

max_depth = 20
num_cases_per_depth = 1000
for depth in range(1, max_depth + 1):
    if depth < 10:
        accept_filename = f'rand_example/accept_0{depth}.test'
        reject_filename = f'rand_example/reject_0{depth}.test'
    else:
        accept_filename = f'rand_example/accept_{depth}.test'
        reject_filename = f'rand_example/reject_{depth}.test'

    accept_cases = []
    reject_cases = []

    for _ in range(num_cases_per_depth):
        accept_language = generate_random_language('CODE', 0, depth)
        if validate_terminal_end(accept_language):
            accept_cases.append(accept_language)

        used_example_grammar_reject = [False]
        reject_language = generate_reject_language('CODE', 0, depth,
used_example_grammar_reject)
        if not used_example_grammar_reject[0]:
            reject_language =
ensure_reject_case_uses_example_grammar(reject_language)

```

```

if reject_language: # 빈 문자열이 아닌 경우에만 추가
    reject_cases.append(reject_language)

with open(accept_filename, 'w') as accept_file:
    accept_file.write('\n'.join(accept_cases))

with open(reject_filename, 'w') as reject_file:
    reject_file.write('\n'.join(reject_cases))

print("File generation complete.")

```

자동 생성된 Accept Case의 일부 (depth 10 설정 시 나올 수 있는 예시들)

...

```

int id ( char id , int id ) { return false ; } float id ( float id ) { id = character ; if ( true ) { if ( true == true ) { } else { int
id ; } } return character ; } char id ( char id ) { return literal ; } char id ;

```

```

char id ; float id ( float id ) { return false ; } float id ( int id ) { return literal ; }

```

```

int id = false ;

```

...

자동 생성된 Reject Case의 일부 (depth 10 설정 시 나올 수 있는 예시들)

...

```

char id ; float id ( ) { return false ; } float id ( char id ) { id = literal ; return error ; } char id ; char id ( int id ) { if
( idonno ) { } else { int id num ; } char id literal ; return character ; }

```

```

char id ( char id , float id , float id , invalid id ) { return true ; } float id = true ; float id ( float id , char id , float id ) { return
literal ; } char id literal ;

```

...

이렇게 여러 케이스를 만들어내는 랜덤 테스트가 있지만, static하게 저희가 직접 Accept, Reject 될 케이스를 다루고 그것을 테스트 하는 것도 안정적인 테스트를 마련하는 데 좋다고 판단하여, static한 케이스를 따로 제공하고 있습니다.

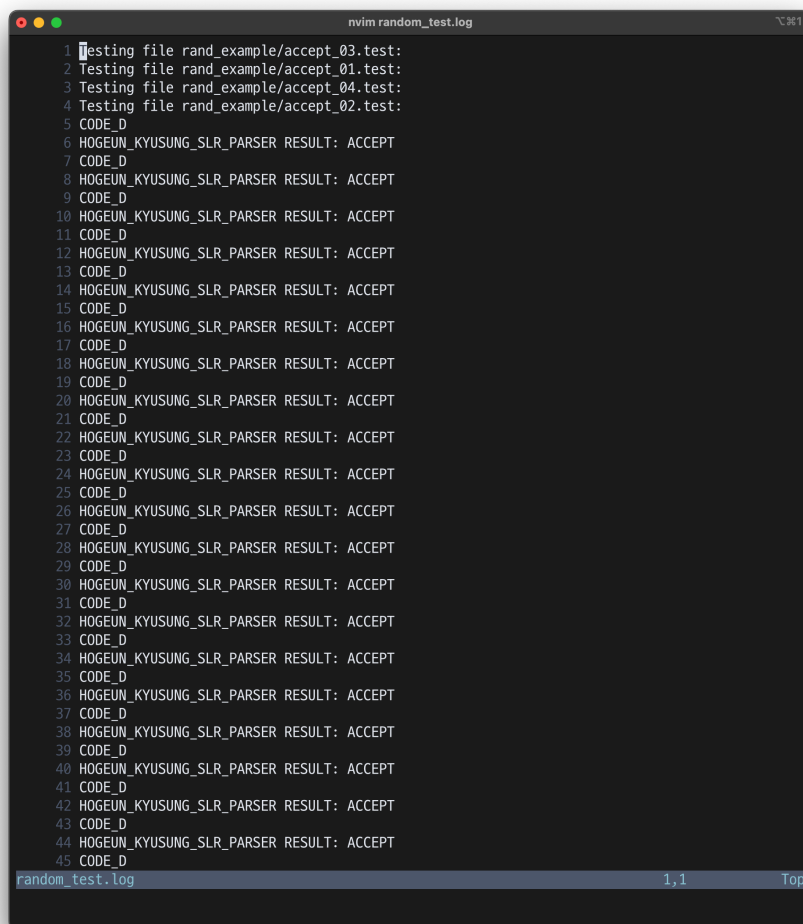
Static Test Case

Src directory내에서, ./static_test.sh 이라는 shell script로 실행될 때, main.py에 ./static_example 에 있는 두 파일 (static_accept_case.txt, static_reject_case.txt) 을 테스트하여 결과를 내보냅니다.

```
→ src git:(main) x ./static_test.sh
Testing file static_example/static_accept_case.txt:
File: static_example/static_accept_case.txt - Passed: 28 /      28 (100.00%)
Testing file static_example/static_reject_case.txt:
File: static_example/static_reject_case.txt - Passed: 50 /     50 (100.00%)
** PERFECT **
```

또한 이러한 모든 테스트의 결과는 .log파일에 담겨 디버깅이 가능합니다.

```
→ src git:(main) x ls
__pycache__      example_maker.py  main.py           random_test.sh    static_test.sh
data             html_to_txt.py   rand_example      static_example     test.log
data.py          input.txt        random_test.log   static_test.log
```



```
nvim random_test.log
1 Testing file rand_example/accept_03.test:
2 Testing file rand_example/accept_01.test:
3 Testing file rand_example/accept_04.test:
4 Testing file rand_example/accept_02.test:
5 CODE_D
6 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
7 CODE_D
8 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
9 CODE_D
10 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
11 CODE_D
12 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
13 CODE_D
14 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
15 CODE_D
16 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
17 CODE_D
18 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
19 CODE_D
20 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
21 CODE_D
22 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
23 CODE_D
24 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
25 CODE_D
26 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
27 CODE_D
28 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
29 CODE_D
30 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
31 CODE_D
32 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
33 CODE_D
34 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
35 CODE_D
36 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
37 CODE_D
38 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
39 CODE_D
40 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
41 CODE_D
42 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
43 CODE_D
44 HOGEUN_KYUSUNG_SLR_PARSER RESULT: ACCEPT
45 CODE_D
random_test.log 1,1 Top
```

고정된 테스트 케이스의 전체를 참조합니다.

Static Accept Cases

```
int id ;

float id ;

char id ;

int id = num ;

float id = num ;

char id = character ;

int id = num ;

float id = num ;

char id = character ;

int id = true ;

int id = false ;

int id ( int id , float id ) { int id ; return num ; }

char id ( ) { return character ; }

int id ( ) { if ( true == true ) { int id ; } else { float id ; } return id ; }

int id ( ) { while ( false != true ) { char id ; } return num ; }

int id = ( num + num ) ;

float id = ( num * num ) ;

int id ( ) { int id ; return num ; }

float id ( float id ) { float id ; return num ; }

int id ; float id ; char id ;

int id = num ; float id = num ; char id = character ;

int id ( int id ) { if ( true == false ) { } else { } return num ; }

char id ( ) { while ( true == true ) { if ( false == false ) { } } return character ; }

int id ; int id ( ) { int id ; return num ; } float id ;

int id ( int id , char id ) { int id ; return num ; } float id ( ) { return num ; }

int id ( ) { if ( true == false ) { while ( true != false ) { char id ; } } else { int id ; } return id ; }

int id = ( num + num ) * ( num - num ) ;

float id = ( num / num ) + ( num * num ) ;
```

Static Reject Cases

int id

float id

char id

int id = ;

float id = ;

char id = ;

int id = unknown ;

float id = 'a' ;

char id = 5 ;

int id = 3.14 ;

if (true) { int id }

while (false) { int id }

int id () { int id return num ; }

char id ({ return 'a' ; }

int id = (num +) ;

float id = (num *) ;

if (true ==) { int id ; } else { float id ; }

while (!= true) { char id ; }

int id (int id ,) { int id ; return num ; }

float id (float id , int) { float id ; return num ; }

int id (int id , float id) { int id return num ; }

char id () { return 'c' }

if (true ==) { int id ; } else { float id ; }

while (!= true) { char id ; }

int id = (num +) ;

float id = (num *) ;

int id (int id ,) { int id ; return num ; }

float id (float id , int) { float id ; return num ; }

```

int id ( ) { int id return num ; }

char id ( ) { return 'a' ; } else { float id ; }

float id ; int id ( int id ) { float id ; }

if ( true == true ) { int id ;

while ( true != false ) { char id ;

int id ; int id ( ) { int id return num ; } float id ;

float id ( float id , int id { float id ; return num ; }

int id ( int id , float id ) { int id ; return ; }

char id = 'a'

int id = num float id = num ;

if ( true == true ) int id ; else { float id ; }

while ( true != false ) char id ;

int id ( ) { int id = ( num + ; return num ; }

float id ( ) { float id = ( num * ; return num ; }

char id ( ) { if ( true == ) { return 'a' ; } }

int id ( ) { while ( != false ) { int id ; } return num ; }

float id ( float id , int id , ) { float id ; return num ; }

int id ( int id , float ) { int id ; return num ; }

char id ( ) { return 'a' ; else { return 'b' ; } }

if ( true ) { int id ; else { float id ; } }

while ( false ) { int id ; if ( true ) { char id ; }

int id ( ) { if ( true == false ) { return num ; } else return num ; }

```

Executable Binary File | 실행 가능 파일 사용법

Python 스크립트를 pyinstaller를 통해 바이너리 파일로 만들었습니다. Apple M1 MacBook Air에서 빌드했기에, 동일 환경에서 실행 가능합니다. 다른 환경의 경우 해당 파일이 실행되지 않는다면, python3.12.2 버전에서 `python3 main.py <input_file>` 형식으로 실행할 수 있습니다. 제공된 Input.txt 파일은 예시임으로, 수정하여 input_file로서 활용하여도 됩니다.

EOD