<레포트>

**SLR parser**

강의명: 컴파일러

분반: 01분반

담당교수: 김효수 교수님

작성자:김진호(20201xx4)

신성섭(20226xx2)

학과: 소프트웨어학부

1. 프로그램 실행 방법

1) pip을 이용하여 Treelib 라이브러리를 설치합니다.

TreeLib 외부 라이브러리를 사용하였기 때문에, 다음 명령어를 통해 라이브러리를 다운 받습니다.

**pip install -U treelib**

* Treelib 라이브러리에 대한 자세한 내용은 두 링크(깃허브와 공식문서)를 통해 더 자세히 확인 가능합니다.

<https://github.com/caesar0301/treelib>

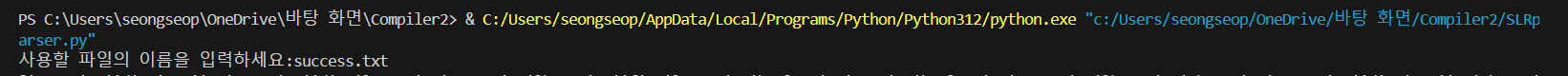
https://treelib.readthedocs.io/en/latest/

2) Command 창에 다음 두 방법 중 하나를 선택하여 프로그램을 실행합니다.

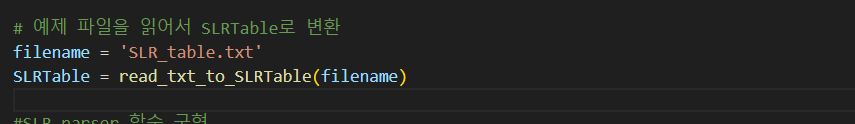
(1) 프로그램 파일과 함께 토큰이 담긴 txt파일 이름을 매개변수로 함께 입력



(2) 프로그램 파일을 우선 실행. 그 후 토큰이 담긴 Txt파일 이름을 입력



* 유의 사항



SLR\_table.txt 파일 이름을 정적으로 선언하여 SLR\_table을 불러오기 때문에 SLR\_table.txt의 파일의 위치가 중요합니다. 이 파일의 위치가 잘못된 경우 프로그램 실행을 실패합니다.



SLR\_table.txt 파일이 있는 곳으로 cd를 이용해 이동한 다음 프로그램을 위와 같이 실행해 주세요. ( 위의 실행의 경우 C:최종 안에 SLRparser.py와 failTest.txt 파일, SLR\_table.txt파일이 모두 들어 있습니다.)

2. CFG

제시되어있던 아래의 CFG에서 ambiguous한 production은 05번과 14번이다. 2개의 production은 특정 input string에대해서 여러 개의 parse tree로 표현될 수 있다.

01: CODE→VDECL CODE | FDECL 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

따라서 ambiguous한 production을 non-ambiguous하게 바꿔준다면 다음과 같다.

1. Non-ambiguous한 05번 production
   1. EXPR -> T addsub EXPR
   2. EXPR -> T
   3. T -> F multdiv T
   4. T -> F
   5. F -> lparen EXPR rparen
   6. F -> id
   7. F -> num
2. Non-ambiguous한 14번 production
   1. COND -> Q comp COND
   2. Q -> boolstr
   3. COND -> Q

3. SLR pasrsing table

텍스트, 평행, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 라인, 평행, 번호이(가) 표시된 사진

자동 생성된 설명

제시된 웹사이트를 통해 non-ambiguous한 CFG를 통해 위의 SLR parsing table을 구할 수 있다. 총 76개의 state로 구성되어있다.

4. syntax analyzer 구현

**1) bottom-up parser 구현을 위한 트리 자료구조**

Treelib 라이브러리를 이용하여 트리를 구현하였습니다.



트리의 각 노드에 값(Terminal/nonterminal Value)를 부여하기 하기 위해 위의 클래스(class nodeValue)를 이용하였습니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

TreeLib 라이브러리 중에서 트리의 새로운 노드를 생성하기 위해 create\_node() 메서드를 이용

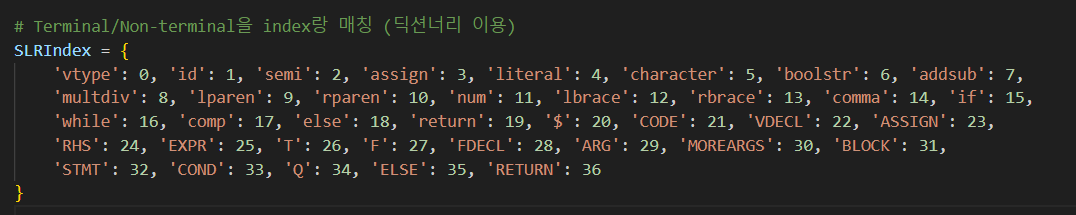
하였고 트리의 노드간 관계를 변경하기 위해 move\_node () 메서드를 이용하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

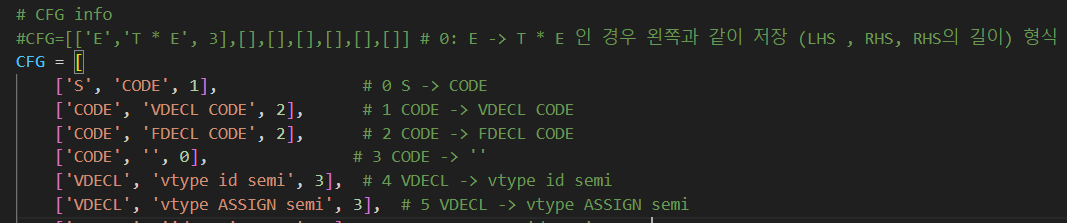
만들어진 Tree를 출력하기 위해 show() 메서드를 이용하였습니다.

**2) SLRIndex 저장**



딕션너리 자료구조를 이용하여, 테이블의 컬럼 값에 해당하는 Terminal과 NonTerminal들한테 순서대로 인덱스를 매겨 값을 부여해 주었습니다.

**3) CFG 저장**



모든 CFG들을 [LHS, RHS, RHS의 길이] 형식으로 저장해 주었습니다.

**4) SLR Table 저장**

txt파일에 SLR parsing table의 값을 저장하여 사용하였다. 빈 값에는 0을 넣고 값이 있는 경우에는 그대로 표기하여 저장하였다.

**5) SLR Parsing 알고리즘**

Pre) 토큰을 메모장에서 읽어 주고 공백을 기준으로 파싱하여 tokens에 token 리스트를 저장한다. Tokens에 ‘$’를 추가한 후 parser(tokens)를 호출한다.

-Parser(tokens) 함수 수행 시작-

(1) 편의성을 위해 dummy-RootNode를 하나 만들어 준다.

(2) tokenPointer, nodeID, 파싱용 stack을 초기화 해준다.

(3) stack에 초기 state 0을 추가한다.

---- 오류가 나거나 Accept 할때 까지 무한 반복한다. ----

(4) 스택의 가장 위에 있는 state 값을 읽어 온다.

(5) tokenPointer가 가리키는 Token 값을 읽어온 후 SLRindex를 이용해 index값을 알아낸다.

(6) (4)와 (5) 과정을 통해 알아 낸 state 값과 index 값을 이용해 SLRTable를 읽는다.

//// 선택문 ////

(A) SLRTable이 빈리스트([]) 인 경우 ( Error에 해당)

SLRTable의 빈 리스트를 읽는 경우는 parsing Error 가 발생한 경우 이므로 error 메시지를 출력하고 error가 발생하기 전까지 형성된 Tree를 출력한다.

(B) SLRTable의 값이 [‘S’, state] 인 경우 (Shift에 해당)

TokenPointer가 가리키는 Token 값을 가지는 새로운 노드를 하나 만든다. 이때 편의상 부모 노드로 우선 dummy-rootNode를 가진다. parser 스택에 노드를 푸쉬한 다음 state 값을 스택에 푸쉬한다. nodeID값과 tokenPointer를 각각 1증가시킨다.

(C) SLRTable의 값이 [‘R’, num] 인 경우 (Reduce + Goto 에 해당)

CFG[num][2]에 해당하는 값에 비례하여 스택에서 pop을 수행한다. CFG[num][2]\*2 번의 pop을 수행한다. 이유는 각 state와 Node가 동일한 스택을 이용하도록 구현하였기 때문이다. Pop된 모든 Node의 경우 ChildNodes 리스트에 저장한다.

CFG[num][0] 값을 Nonterminal로 저장한다. Stack의 가장 위에 있는 state 값을 읽어온후Nonterminal과 state를 이용해 SRLtable이 Goto 파트를 읽는다. NonTerminal을 Value로가지고 부모가 dummy-root인 노드를 생성한다.

NonTerminal 값을 가지는 노드를 부모로 하고 ChildNode 리스트에 저장된 모든 노드를 자식으로 하여 부모-자식 관계를 형성해 준다.

부모 노드(Nonterminal 값을 가지는 새로운 노드)만 스택에 푸쉬한다. 그 후 Goto 파트에서 읽은 state 값도 Stack에 푸쉬한다. 그후 nodeId를 1증가시킨다.

(D) SLR table 값이 [‘A”, 0] 인 경우

Parsing이 성공적으로 수행된 경우 이다. Parsing Tree를 출력한 다음 프로그램을 종료한다.

/// 선택문 종료 ///

---- 반복 종료 ----

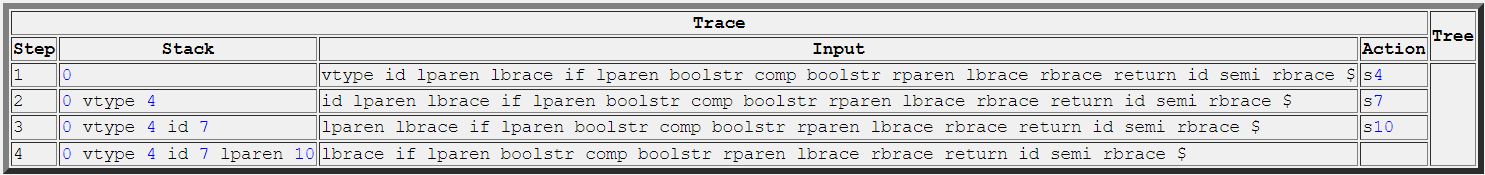
- Parser() 함수 종료 및 프로그램 종료-

**6) error report for reject**

reject되는 input을 SLR parsing 알고리즘에 넣게 되면 SLR parsing table에서 다음에 수행될 shift, reduce 또는 goto action이 없는 경우가 발생하게 된다. 이러한 경우 reject가 발생하기까지 parsing의 과정을 step순서대로 stack과 input의 값을 출력해주도록 하였다. 그리고 Reject가 발생한 지점에서 몇번째 step에서 어떤 특정 state의 action과 goto에 해당하는 값(shift, reduce, goto)이 없는지를 출력하도록 하였다. 또한 파싱이 성공한데까지의 tree를 출력해주었다.

예시) reject되는 경우의 input(제출파일안에 failTest3.txt파일안에 있는 테스트 input값 : vtype id lparen lbrace if lparen boolstr comp boolstr rparen lbrace rbrace return id semi rbrace

1. 제시된 사이트에서의 parsing 과정



1. 구현한 syntax analyzer프로그램의 ouput

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

5. syntax analyzer Test

**1) 성공 케이스**

1. input string(successTest.txt) : vtype id semi vtype id lparen rparen lbrace if lparen boolstr comp boolstr rparen lbrace rbrace return id semi rbrace

2. output : parse tree

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

1. input string(successTest.txt2) : vtype id semi vtype id assign lparen id addsub num rparen multdiv id semi vtype id lparen vtype id comma vtype id rparen lbrace while lparen boolstr comp boolstr rparen lbrace rbrace return boolstr semi rbrace
2. output : parse tree

스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**2) 실패 케이스**

1. input string(failTest.txt) : vtype id semi vtype id lparen rparen lbrace if lparen boolstr comp boolstr lbrace rbrace return id semi rbrace

2. ouput

텍스트, 스크린샷, 메뉴이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명