­

**Term Project Report**

**SLR Parser**

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

Table of Contents

[Abstract | 개요 - 3 -](#_Toc168527699)

[NON-Ambiguous CFG | 모호하지 않은 CFG 제작 - 4 -](#_Toc168527700)

[Change Things | 기존 CFG 대비 변경사항 - 5 -](#_Toc168527701)

[SLR parsing table - 6 -](#_Toc168527702)

[All Implementation - 17 -](#_Toc168527703)

[main.py - 17 -](#_Toc168527704)

[data.py - 20 -](#_Toc168527705)

[html\_to\_txt.py - 21 -](#_Toc168527706)

[Test Cases - 22 -](#_Toc168527707)

[Random Test Case - 23 -](#_Toc168527708)

[Example\_maker.py - 23 -](#_Toc168527709)

[Static Test Case - 27 -](#_Toc168527710)

[Executable Binary File | 실행 가능 파일 사용법 - 31 -](#_Toc168527711)

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

# SLR parsing table

| **LR table** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **State** | **ACTION** | | | | | | | | | | | | | | | | | | | | | | | | | | | **GOTO** | | | | | | | | | | | | | | | | | | | | |
| **id** | **;** | **=** | **literal** | **character** | **(** | **)** | **num** | **{** | **}** | **,** | **if** | **while** | **else** | **return** | **+** | **-** | **\*** | **/** | **==** | **!=** | **true** | **false** | **int** | **float** | **char** | **$** | **CODE** | **CODE\_D** | **VDECL** | **ASSIGN** | **RHS** | **EXPR** | **EXPR\_D** | **EXPR\_DD** | **FDECL** | **ARG** | **MOREARGS** | **BLOCK** | **STMT** | **COND** | **ELSE** | **RETURN** | **ADDSUB** | **MULDIV** | **COMP** | **BOOLSTR** | **VTYPE** |
| 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s5 | s6 | s7 | r3 |  | 1 | 3 |  |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  | 4 |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | acc |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s5 | s6 | s7 | r3 |  | 8 | 3 |  |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  | 4 |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s5 | s6 | s7 | r3 |  | 9 | 3 |  |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  | 4 |
| 4 | s10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | r42 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | r43 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | r44 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  | s13 | s14 |  |  | s12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 |  | s15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  | r20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  |  |  |  |  |  |  |  | 16 |  |  |  |  |  |  |  |  |  |  | 17 |
| 13 | r4 |  |  |  |  |  |  |  |  | r4 |  | r4 | r4 |  | r4 |  |  |  |  |  |  |  |  | r4 | r4 | r4 | r4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 | s28 |  |  | s20 | s21 | s27 |  | s29 |  |  |  |  |  |  |  |  |  |  |  |  |  | s24 | s25 |  |  |  |  |  |  |  |  | 18 | 19 | 23 | 26 |  |  |  |  |  |  |  |  |  |  |  | 22 |  |
| 15 | r5 |  |  |  |  |  |  |  |  | r5 |  | r5 | r5 |  | r5 |  |  |  |  |  |  |  |  | r5 | r5 | r5 | r5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 |  |  |  |  |  |  | s30 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 | s31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 |  | r6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  | r7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  | r8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 21 |  | r9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 22 |  | r10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 23 |  | r12 |  |  |  |  | r12 |  |  |  |  |  |  |  |  | s33 | s34 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 32 |  |  |  |  |
| 24 |  | r40 |  |  |  |  | r40 |  |  |  |  |  |  |  |  |  |  |  |  | r40 | r40 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 25 |  | r41 |  |  |  |  | r41 |  |  |  |  |  |  |  |  |  |  |  |  | r41 | r41 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 26 |  | r14 |  |  |  |  | r14 |  |  |  |  |  |  |  |  | r14 | r14 | s36 | s37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 35 |  |  |  |
| 27 | s28 |  |  |  |  | s27 |  | s29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 38 | 23 | 26 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 28 |  | r16 |  |  |  |  | r16 |  |  |  |  |  |  |  |  | r16 | r16 | r16 | r16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 29 |  | r17 |  |  |  |  | r17 |  |  |  |  |  |  |  |  | r17 | r17 | r17 | r17 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  | s39 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31 |  |  |  |  |  |  | r22 |  |  |  | s41 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 40 |  |  |  |  |  |  |  |  |  |  |
| 32 | s28 |  |  |  |  | s27 |  | s29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 42 | 23 | 26 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 33 | r34 |  |  |  |  | r34 |  | r34 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 34 | r35 |  |  |  |  | r35 |  | r35 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 35 | s28 |  |  |  |  | s27 |  | s29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 43 | 26 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 | r36 |  |  |  |  | r36 |  | r36 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 37 | r37 |  |  |  |  | r37 |  | r37 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 38 |  |  |  |  |  |  | s44 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 39 | s52 |  |  |  |  |  |  |  |  | r24 |  | s49 | s50 |  | r24 |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  | 47 | 48 |  |  |  |  |  |  |  | 45 | 46 |  |  |  |  |  |  |  | 51 |
| 40 |  |  |  |  |  |  | r19 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 41 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 53 |
| 42 |  | r11 |  |  |  |  | r11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 43 |  | r13 |  |  |  |  | r13 |  |  |  |  |  |  |  |  | r13 | r13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 44 |  | r15 |  |  |  |  | r15 |  |  |  |  |  |  |  |  | r15 | r15 | r15 | r15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s55 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 54 |  |  |  |  |  |
| 46 | s52 |  |  |  |  |  |  |  |  | r24 |  | s49 | s50 |  | r24 |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  | 47 | 48 |  |  |  |  |  |  |  | 56 | 46 |  |  |  |  |  |  |  | 51 |
| 47 | r25 |  |  |  |  |  |  |  |  | r25 |  | r25 | r25 |  | r25 |  |  |  |  |  |  |  |  | r25 | r25 | r25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 48 |  | s57 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 49 |  |  |  |  |  | s58 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  | s59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 51 | s60 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 52 |  |  | s14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 53 | s61 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 54 |  |  |  |  |  |  |  |  |  | s62 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 55 | s28 |  |  | s20 | s21 | s27 |  | s29 |  |  |  |  |  |  |  |  |  |  |  |  |  | s24 | s25 |  |  |  |  |  |  |  |  | 63 | 19 | 23 | 26 |  |  |  |  |  |  |  |  |  |  |  | 22 |  |
| 56 |  |  |  |  |  |  |  |  |  | r23 |  |  |  |  | r23 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 57 | r26 |  |  |  |  |  |  |  |  | r26 |  | r26 | r26 |  | r26 |  |  |  |  |  |  |  |  | r26 | r26 | r26 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 58 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s24 | s25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 64 |  |  |  |  |  | 65 |  |
| 59 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s24 | s25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 66 |  |  |  |  |  | 65 |  |
| 60 |  | s13 | s14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 61 |  |  |  |  |  |  | r22 |  |  |  | s41 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 67 |  |  |  |  |  |  |  |  |  |  |
| 62 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r18 | r18 | r18 | r18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 63 |  | s68 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 64 |  |  |  |  |  |  | s69 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 65 |  |  |  |  |  |  | r30 |  |  |  |  |  |  |  |  |  |  |  |  | s71 | s72 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 70 |  |  |
| 66 |  |  |  |  |  |  | s73 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 67 |  |  |  |  |  |  | r21 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 68 |  |  |  |  |  |  |  |  |  | r33 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 69 |  |  |  |  |  |  |  |  | s74 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | s24 | s25 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 75 |  |  |  |  |  | 65 |  |
| 71 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r38 | r38 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 72 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | r39 | r39 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 73 |  |  |  |  |  |  |  |  | s76 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 74 | s52 |  |  |  |  |  |  |  |  | r24 |  | s49 | s50 |  | r24 |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  | 47 | 48 |  |  |  |  |  |  |  | 77 | 46 |  |  |  |  |  |  |  | 51 |
| 75 |  |  |  |  |  |  | r29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 76 | s52 |  |  |  |  |  |  |  |  | r24 |  | s49 | s50 |  | r24 |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  | 47 | 48 |  |  |  |  |  |  |  | 78 | 46 |  |  |  |  |  |  |  | 51 |
| 77 |  |  |  |  |  |  |  |  |  | s79 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 78 |  |  |  |  |  |  |  |  |  | s80 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 79 | r32 |  |  |  |  |  |  |  |  | r32 |  | r32 | r32 | s82 | r32 |  |  |  |  |  |  |  |  | r32 | r32 | r32 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 81 |  |  |  |  |  |  |
| 80 | r28 |  |  |  |  |  |  |  |  | r28 |  | r28 | r28 |  | r28 |  |  |  |  |  |  |  |  | r28 | r28 | r28 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 81 | r27 |  |  |  |  |  |  |  |  | r27 |  | r27 | r27 |  | r27 |  |  |  |  |  |  |  |  | r27 | r27 | r27 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 82 |  |  |  |  |  |  |  |  | s83 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 83 | s52 |  |  |  |  |  |  |  |  | r24 |  | s49 | s50 |  | r24 |  |  |  |  |  |  |  |  | s5 | s6 | s7 |  |  |  | 47 | 48 |  |  |  |  |  |  |  | 84 | 46 |  |  |  |  |  |  |  | 51 |
| 84 |  |  |  |  |  |  |  |  |  | s85 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 85 | r31 |  |  |  |  |  |  |  |  | r31 |  | r31 | r31 |  | r31 |  |  |  |  |  |  |  |  | r31 | r31 | r31 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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 = [] # input을 넣으며 실제로 reduce 하는 데 사용하는 스택임  
 self.inputStack.append('$')  
 self.nodequeue = []  
  
 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 True로 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 action이 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'):  
 # action이 s일 때는 shift를 함.  
 # inputStack에 현재 토큰을 push함  
 self.inputStack.append(current\_token)  
 # 트리에 출력에 활용하기 위해 큐에 붙임  
 self.nodequeue.append(TreeNode(current\_token))  
 # 상태 스택에 이제 해야 하는 action을 push함  
 self.statestack.append(int(action[1:]))  
 token\_index += 1  
 elif action.startswith('r'):  
 # action이 r일 때는 reduce를 함.  
 node = TreeNode(self.rules[int(action[1:])][0])  
 pop\_ = len(self.nodequeue) - self.rules[int(action[1:])][1]  
 # grammar인 rules에서 그것의 길이만큼 빼줘야 함. 그만큼 reduce된 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()}')  
 node.add\_child(self.nodequeue.pop(pop\_))  
 self.nodequeue.append(node)  
 self.inputStack.append(self.rules[int(action[1:])][0])  
 # 현재 상태와 토큰을 업데이트 해줌 (top에 있는 거로 최신화 해주는 것임)  
 current\_state = self.statestack[-1]  
 current\_token = self.inputStack[-1]  
 try:  
 # 파스 테이블에 현재 상태에서의 현재 토큰일 때의 action이 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 case임  
 self.statestack.append(int(action))  
 elif action == 'acc':  
 # accept할 때임.  
 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(CODE로 REDUCE되는. 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 prefix가 있는 RESULT 를 출력함.  
 print("HOGEUN\_KYUSUNG\_SLR\_PARSER RESULT: ACCEPT")  
 else:  
 print(f"line : {line\_index}")  
 # 쉘 스크립트에서 이 스트링을 검출하기 위해 앞에 prefix가 있는 RESULT 를 출력함.  
 print("HOGEUN\_KYUSUNG\_SLR\_PARSER RESULT: REJECT")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

main.py에서는, data.py를 참조하여 그 내부에 있는 grammer와 table을 활용하고 있습니다. 아래는 data.py입니다.

## 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.py에서 사용하는 파일은, 결국 html raw data에서 한 번 가공이 된 파일입니다. 아래 코드에서는 beautifulsoup을 이용해 html을 table변수에 담아 파일을 만듭니다. 이렇게 한 번 html\_to\_txt.py를 거치는 이유는, 혹시나 data.py에서 활용하는 형태가 달라지는 것이 이 스크립트를 수정해야하는 의존성을 없애기 위함입니다. 일단 눈에 보기 쉬운 꼴로 한 번 table 파일을 만들고, 그것을 data.py에서 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 쉘 스크립트가 있습니다. 랜덤의 경우 어떻게 만들어지는지 설명하며, static의 경우 어떤 input case들을 대표적으로 테스트하였는지 설명합니다.

A screen shot of a computer

Description automatically generated

## Random Test Case

랜덤 테스트는 ./random\_test.sh 이라는 shell script로 작동합니다. 해당 스크립트가 불릴 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']  
}  
  
# accpet에 대한 최대 깊이 도달 시 사용하는 우선순위 배열. terminal을 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. validation을 위해  
terminals = [';', 'id', 'literal', 'character', 'num', 'true', 'false', 'int', 'float', 'char', '+', '-', '\*', '/', '==', '!=', '(', ')', '{', '}', 'else', 'return']  
  
# reject에 대한 최대 깊이 도달 시 사용하는 우선순위 배열  
example\_grammar\_reject = {  
 'CODE': ['!~'],  
 'CODE\_D': ['}~'],  
 'VDECL': ['char id literal ;'],  
 'ASSIGN': ['id === true'],  
 'RHS': ['rhserror'],  
 '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 cases만 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\_reject의 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.

A black screen with white text

Description automatically generated

또한 이러한 모든 테스트의 결과는 .log파일에 담겨 디버깅이 가능합니다.

A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

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