ÖZYEĞİN UNIVERSITY
Automaton Theory and Formal Languages
Fall 2022

CS 410
Project 2

Design Report

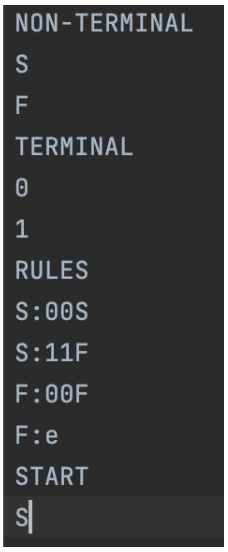
By
Yasemin Orhun
S018151

I. Overview

This report describes the details and the design of a Context-free grammar to Chomsky normal form converter program written with Java.

The program will read an input file, form the CFG and convert it to CNF.

II. Objectives

- The program first reads the input files (G1 or G2 in this case).
- Stores the alphabet, non-terminals, terminals, the start symbol and the rules:
- Example input file format: 

```
NON-TERMINAL
S
F
TERMINAL
0
1
RULES
S:00S
S:11F
F:00F
F:e
START
S
```

- After storing the alphabet information, follows the below steps:
  1. Add a new starting variable and map the rules.
  2. If there exists any epsilon, eliminate and refactor rules.
  3. Delete duplicates.
  4. Handle unit rules:
     a) S0→S, A→S and B→S (unit rules with start symbol)
     b) A→B, B→A (other unit rules)
  5. Delete duplicates.
  6. Add new rulesets (X1, X2, X3….Xn) for remaining rules and convert all for Chomsky Normal Form: A →BC, A →a
- Prints out the results through transformation.

III. Design Choices

Java is used for the implementation of this project. To form the context free grammar with the given alphabet, HashMap is used. First the program stores the alphabet in List formats, later the program maps the lists and forms a HashMap.

Every key value in the alphabet refers to a variable, and the value is the rule:

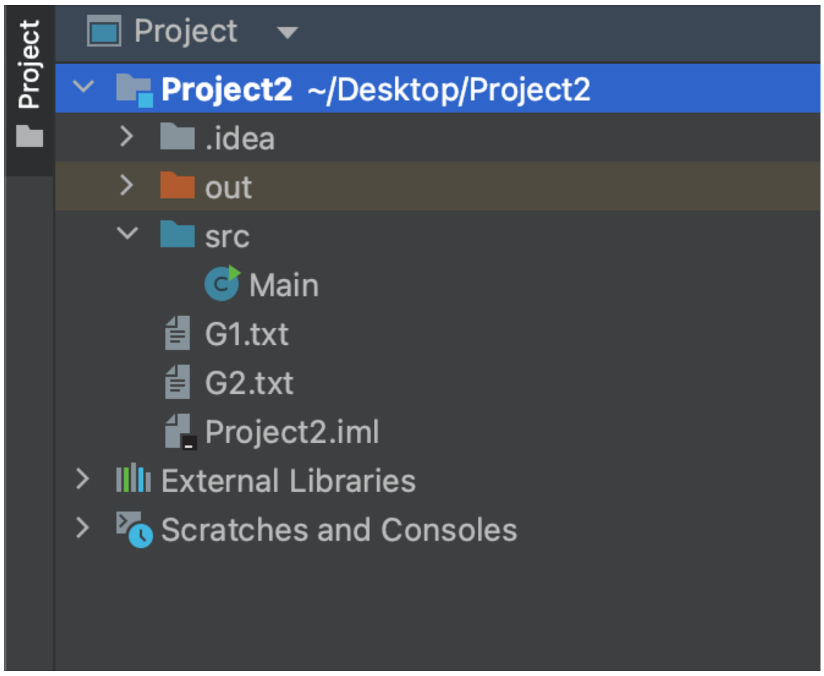
Example:

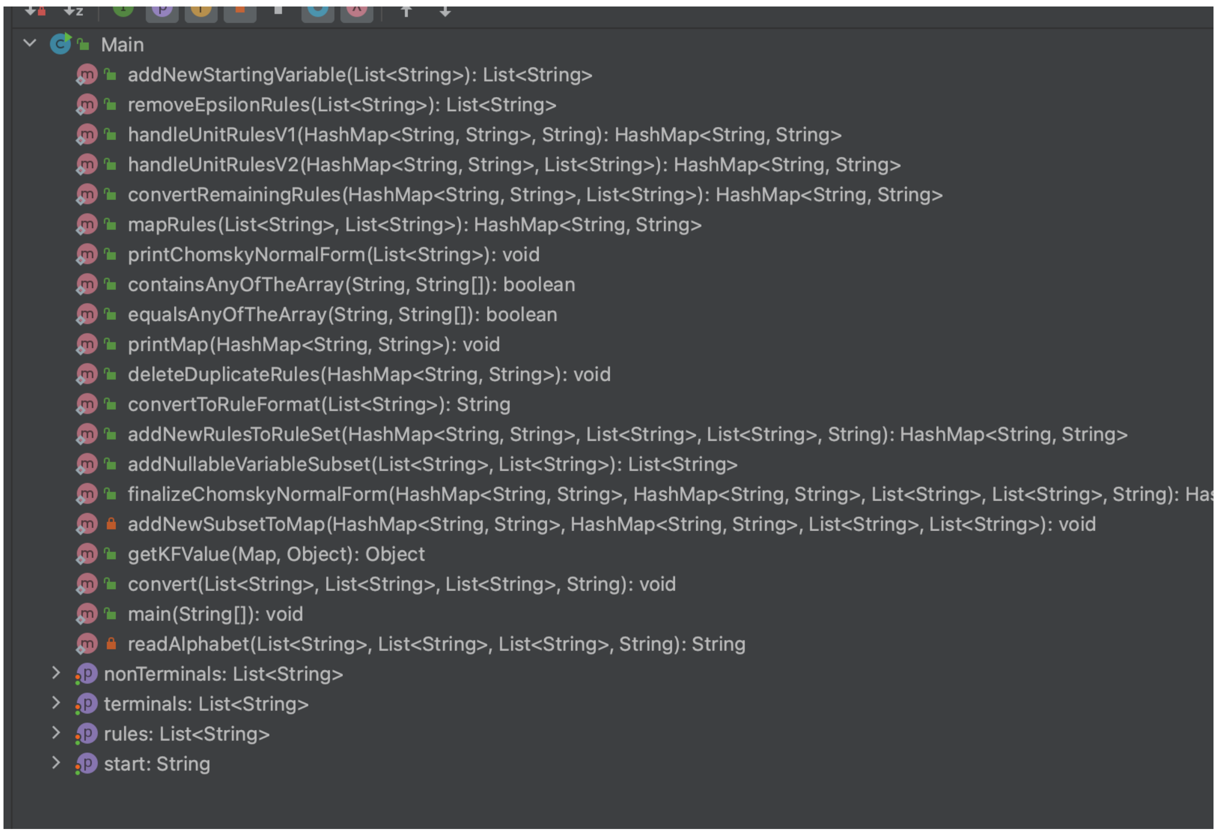
Key = S

Value = AB | b **S→AB | b**

In order to form the final Chomsky normal form reached the rules through keys and avlues of the HashMap.

IV. Design Details





```
Main
    addNewStartingVariable(List<String>): List<String>
    removeEpsilonRules(List<String>): List<String>
    handleUnitRulesV1(HashMap<String, String>, String): HashMap<String, String>
    handleUnitRulesV2(HashMap<String, String>, List<String>): HashMap<String, String>
    convertRemainingRules(HashMap<String, String>, List<String>): HashMap<String, String>
    mapRules(List<String>, List<String>): HashMap<String, String>
    printChomskyNormalForm(List<String>): void
    containsAnyOfTheArray(String, String[]): boolean
    equalsAnyOfTheArray(String, String[]): boolean
    printMap(HashMap<String, String>): void
    deleteDuplicateRules(HashMap<String, String>): void
    convertToRuleFormat(List<String>): String
    addNewRulesToRuleSet(HashMap<String, String>, List<String>, List<String>, String): HashMap<String, String>
    addNullableVariableSubset(List<String>, List<String>): List<String>
    finalizeChomskyNormalForm(HashMap<String, String>, HashMap<String, String>, List<String>, List<String>, String): Has
    addNewSubsetToMap(HashMap<String, String>, HashMap<String, String>, List<String>, List<String>): void
    getKFValue(Map, Object): Object
    convert(List<String>, List<String>, List<String>, String): void
    main(String[]): void
    readAlphabet(List<String>, List<String>, List<String>, String): String
  > nonTerminals: List<String>
  > terminals: List<String>
  > rules: List<String>
  > start: String
```

V. Implementation

In the Main class, the file is read and the convert method is called.

```
convert( List<String> nonTerminals, List<String> terminals, List<String>
rules, String start)
```

is called.

The convert method calls the below methods:

```
addNewStartingVariable(rules);
rules = removeEpsilonRules(rules);
deleteDuplicateRules(map);
map = handleUnitRulesV1(map, start);
deleteDuplicateRules(map);
map = handleUnitRulesV2(map, nonTerminals);
deleteDuplicateRules(map);
HashMap<String, String> newRuleSet = addNewRulesToRuleSet(map,
nonTerminals, terminals, start);
HashMap<String, String> CNF = finalizeChomskyNormalForm(map,newRuleSet,
nonTerminals, terminals, start);
printMap(CNF);
```

Methods:

public List<String> addNewStartingVariable(List<String> rules)

- This method adds a new staring variable to all the rules read from the input file.
  S0 → S

public List<String> removeEpsilonRules(List<String> rules)

- This method checks if a ruleset has epsilon value, if so then removes the epsilon and adds the required additional rules to the other rulesets regarding to that variable.
- Example:

S→ Ab	S→Ab | b
A→B | ~~e~~	A →B

public HashMap<String, String> handleUnitRulesV1(HashMap<String, String> rules, String start)

- "handleUnitRulesV1" method handles the unit rules such as S0 → S, A→S and B→S. Removes unit rules and add their corresponding rulesets.
- Example:

S0→ S	S0→ Ab | AAa
S→Ab | AAa	S→Ab | AAa
A→AA | bA | S	A→AA | bA | Ab | AAa

public HashMap<String, String> handleUnitRulesV2(HashMap<String, String> rules, List<String> nonTerminals)

- "handleUnitRulesV2" method handles the unit rules such as A → B. Removes unit rules and add their corresponding rulesets.

public void deleteDuplicateRules(HashMap<String, String> map)

- Deletes duplicate rules if there exists.

public HashMap<String, String> addNewRulesToRuleSet(HashMap<String, String> map, List<String> nonTerminals, List<String> terminals, String start)

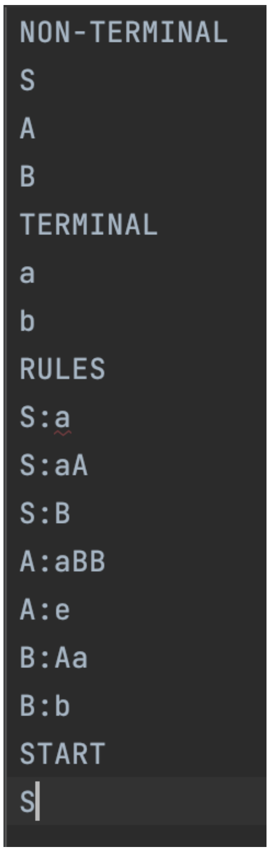
- If there are any rules left that are not in the form of Chomsky normal form **(A→BC or A→a)** creates new rules with adding new non terminals such as **(X1, X2, X3 ....Xn)**
- Example:
  A->aa X1→ a

public HashMap<String, String> finalizeChomskyNormalForm(HashMap<String, String> map, HashMap<String, String> newSubsetMap,List<String> nonTerminals, List<String> terminals, String start)

- With the new rule subset, maps the ruleset and forms the finalized Chomsky normal form with the additional variables.
  A→X1X1

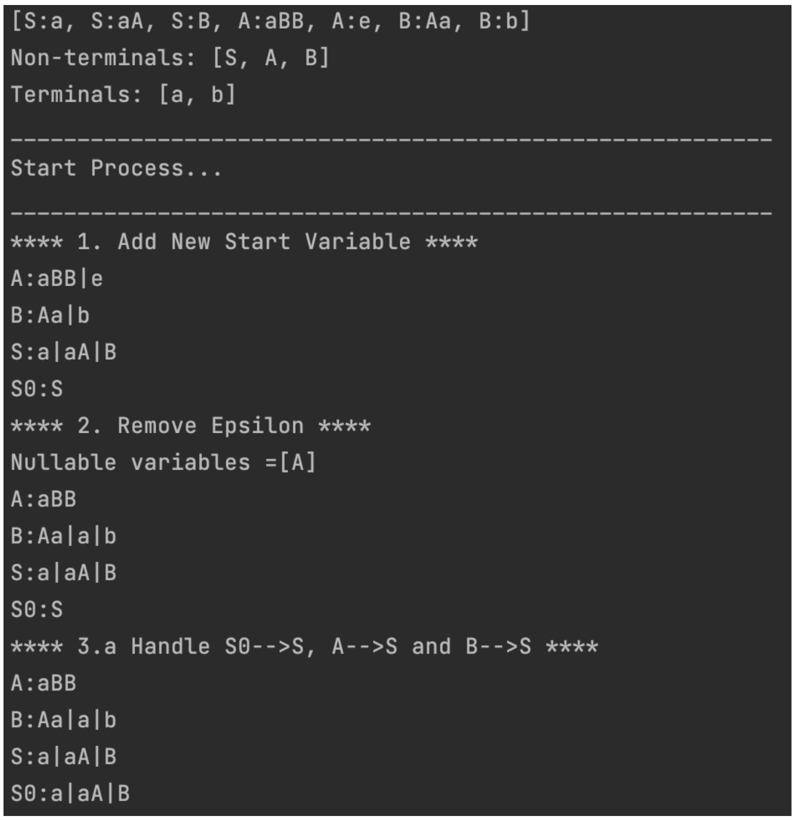
VI. Results

Example result:
Input file:

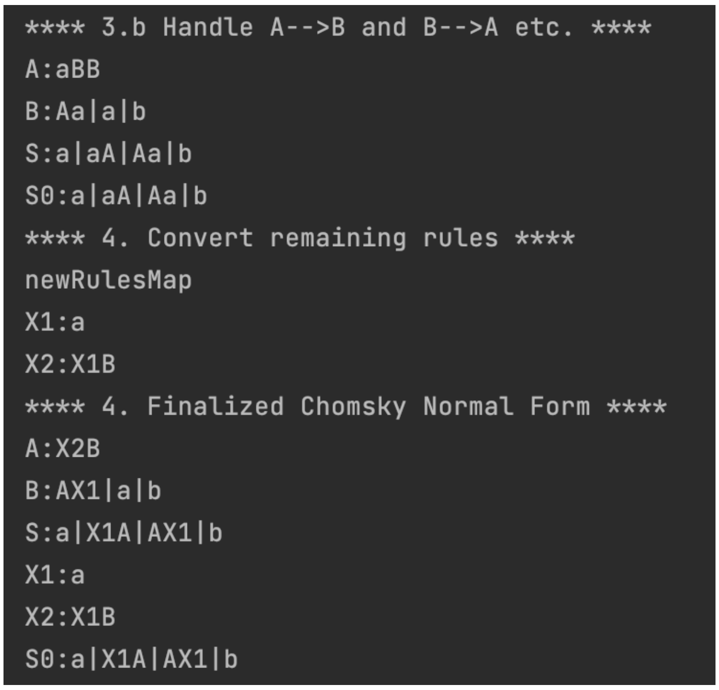


```
NON-TERMINAL
S
A
B
TERMINAL
a
b
RULES
S:a
S:aA
S:B
A:aBB
A:e
B:Aa
B:b
START
S
```

Result:



```
[S:a, S:aA, S:B, A:aBB, A:e, B:Aa, B:b]
Non-terminals: [S, A, B]
Terminals: [a, b]
---------------------------------------------------------
Start Process...
---------------------------------------------------------
**** 1. Add New Start Variable ****
A:aBB|e
B:Aa|b
S:a|aA|B
S0:S
**** 2. Remove Epsilon ****
Nullable variables =[A]
A:aBB
B:Aa|a|b
S:a|aA|B
S0:S
**** 3.a Handle S0-->S, A-->S and B-->S ****
A:aBB
B:Aa|a|b
S:a|aA|B
S0:a|aA|B
```



```
**** 3.b Handle A-->B and B-->A etc. ****
A:aBB
B:Aa|a|b
S:a|aA|Aa|b
S0:a|aA|Aa|b
**** 4. Convert remaining rules ****
newRulesMap
X1:a
X2:X1B
**** 4. Finalized Chomsky Normal Form ****
A:X2B
B:AX1|a|b
S:a|X1A|AX1|b
X1:a
X2:X1B
S0:a|X1A|AX1|b
```