DOKUZ EYLUL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CME3203 Theory of Computation

Converting CFGs to Chomsky Normal Form

BY

2020510137 Emre Özkaya
2018510117 Furkan Özdemir

The code imports several Java classes and declares several global variables. The alphabet variable is a List object containing all the uppercase letters from 'A' to 'Z', and the hashMapTemp variable is a HashMap object that will be used later in the code.

The main method is where the main logic of the program is implemented. It starts by reading in a file containing a context-free grammar and storing the grammar in a HashMap object called hashMap. The key of each entry in the hashMap is a nonterminal symbol, and the value is a List object containing the right-hand side of the production for that nonterminal symbol.

After the hashMap has been initialized, the code calls the printHash method to print the grammar to the console. It then calls the removeEpsilon, removeUnitProduction, and removeTerminals methods to transform the grammar. These methods modify the hashMap object in place, so the original grammar is modified as a result of these operations.

After the transformation methods have been called, the code calls the breakVariables method to break variables that are longer than two into shorter variables. This method modifies a separate HashMap object called hashMapTemp, which is used to store the resulting grammar.

Finally, the code calls the printHash method to print the transformed grammar to the console.

Pseudocode:

- Read in a file containing a context-free grammar and store it in a HashMap object called hashMap.
- Print the grammar using the printHash method.
- Remove epsilon productions from the grammar using the removeEpsilon method.
- Remove unit productions from the grammar using the removeUnitProduction method.
- Remove terminals from the grammar using the removeTerminals method.
- Break variables that are longer than two into shorter variables and store the resulting grammar in a HashMap object called hashMapTemp using the breakVariables method.
- Print the transformed grammar using the printHash method.

Sample screenshot:

```
CFG Form
S-A1A
B-A|10
A-0B0|€

------Eliminate €----
S-A1A|1A|A1|1
B-A|10
A-0B0|00

------Eliminate Unit Product----
S-A1A|1A|A1|1
B-10|0B0|00
A-0B0|00

------Eliminate Terminals----
S-AVA|VA|AV|1
B-VI|IBI|II
A-IBI|II
I-0
V-1

------Break Variables That Are Longer Than Two------
                ----CNF------
S-AZ|VA|AV|1
B-VI|IU|II
A-IU|II
Z-VA
I-0
V-1
U-BI
```

Sample One

- **void printHash(HashMap<String, List<String>> hashMap)**

This function takes a HashMap object containing a context-free grammar and prints it to the console. The function iterates through the entries in the HashMap and prints each nonterminal symbol and its corresponding right-hand side production.

- **void removeEpsilon(HashMap<String, List<String>> hashMap)**

This function removes epsilon productions (productions of the form "A -> ε", where ε is the empty string) from the context-free grmmar stored in the hashMap object. The function iterates through the entries in the HashMap and removes any occurrences of the empty string from the right-hand side productions.

- **void removeUnitProduction(HashMap<String, List<String>> hashMap)**

This function removes unit productions (productions of the form "A -> B", where A and B are nonterminal symbols) from the context-free grammar stored in the hashMap object. The function iterates through the entries in the HashMap and, for each entry, checks if the right-hand side production consists of a single nonterminal symbol. If it does, the function looks up the production for that nonterminal symbol in the hashMap and adds it to the current entry's right-hand side production.

- **void removeTerminals(HashMap<String, List<String>> hashMap)**

This function removes terminals (productions of the form "A -> a", where A is a nonterminal symbol and a is a terminal symbol) from the context-free grammar stored in the hashMap object. The function iterates through the entries in the HashMap and, for each entry, checks if the right-hand side production consists of a single terminal symbol. If it does, the function removes that production from the hashMap.

- **void breakVariables(HashMap<String, List<String>> hashMap)**

This function breaks variables that are longer than two (productions of the form "A -> XYZ", where X, Y, and Z are nonterminal symbols) into shorter variables. The function iterates through the entries in the HashMap and, for each entry, checks if the right-hand side production consists of more than two nonterminal symbols. If it does, the function breaks the production into shorter productions and adds them to the hashMapTemp object. For example, the production "A -> XYZ" would be broken into the productions "A -> X Y" and "Y -> Z".

Note that the breakVariables function is the only function that modifies the hashMapTemp object instead of the hashMap object. This is because the breakVariables function is intended to be called after the other transformation functions have been applied, and it operates on the transformed grammar rather than the original grammar.