

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Inginerie Software și Automatică**

Report

*Laboratory work n.5*

*Variant - 25*

***Formal Languages &Finite Automata***

***Author:*** *Țapu Pavel*

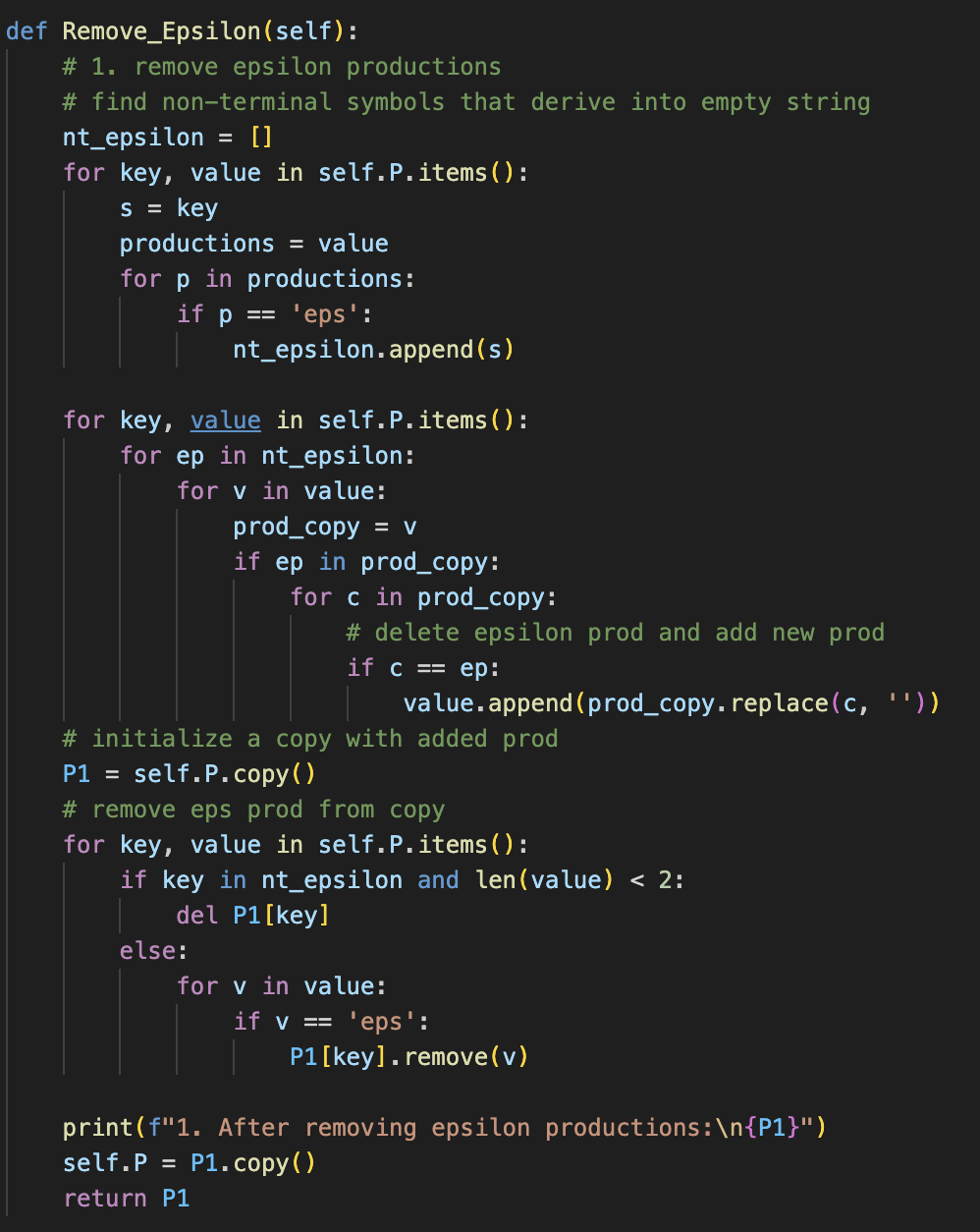
**Chișinău – 2023**

**Objectives**

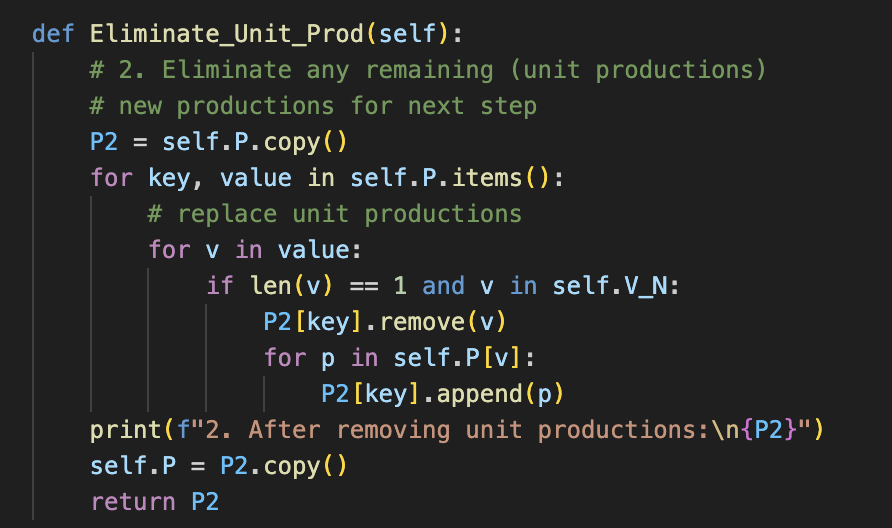
1. Learn about Chomsky Normal Form (CNF) [1].
2. Get familiar with the approaches of normalizing a grammar.
3. Implement a method for normalizing an input grammar by the rules of CNF.
4. The implementation needs to be encapsulated in a method with an appropriate signature (also ideally in an appropriate class/type).
5. The implemented functionality needs executed and tested.
6. A BONUS point will be given for the student who will have unit tests that validate the functionality of the project.
7. Also, another BONUS point would be given if the student will make the aforementioned function to accept any grammar, not only the one from the student's variant.

**Implementation Description**

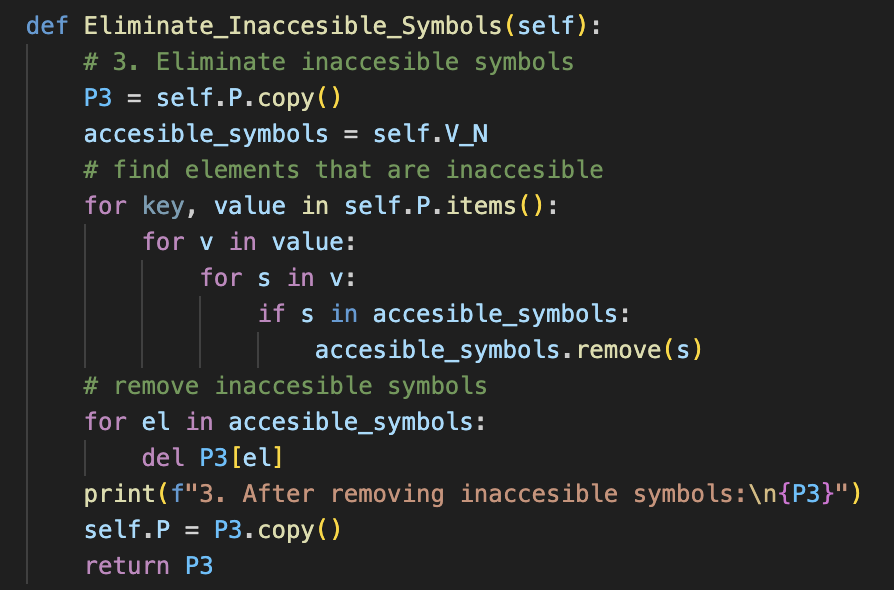
The Grammar class is designed to represent context-free grammars and facilitate their normalization into CNF. Here's a detailed breakdown of the class methods and their functionalities:



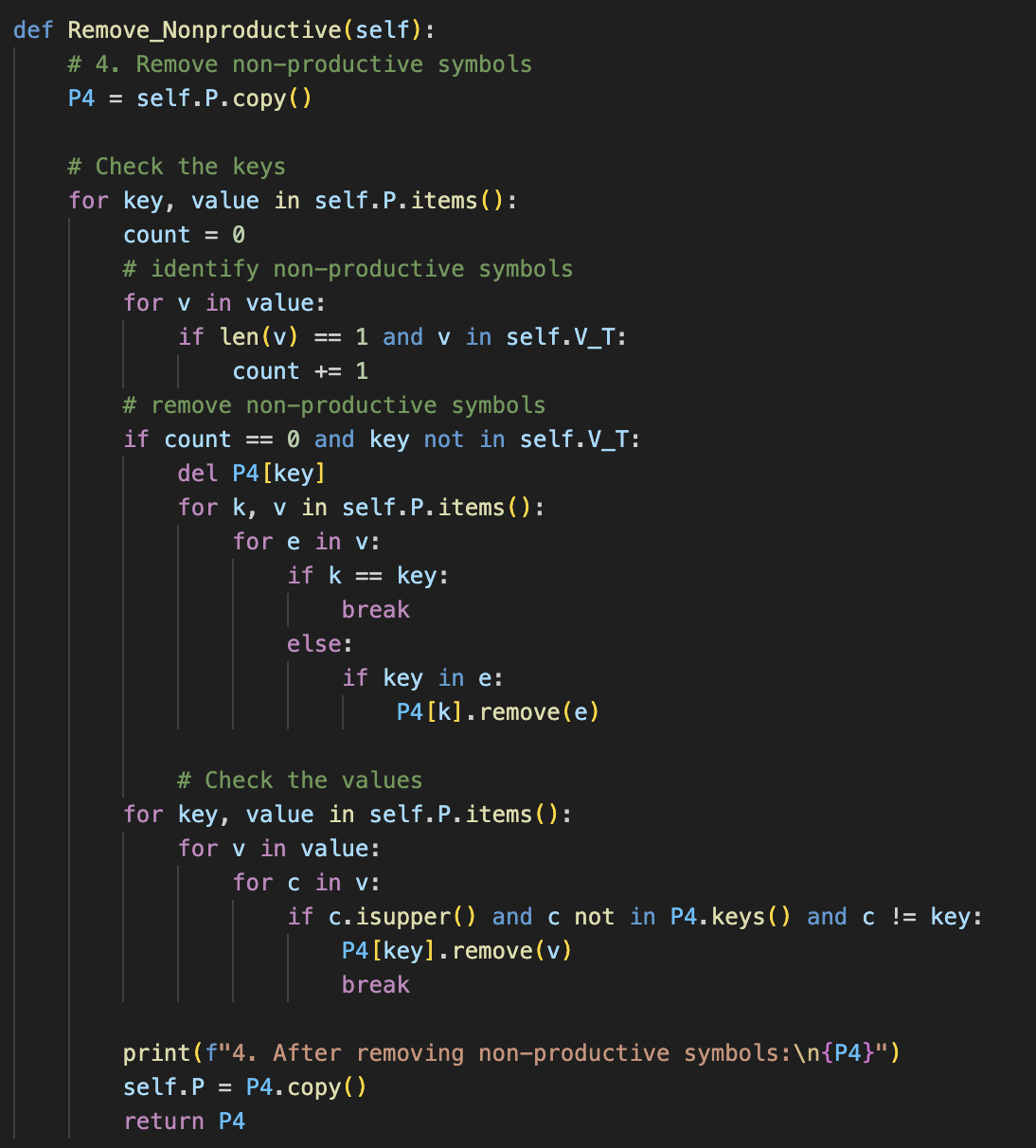
Remove Epsilon Productions (Remove\_Epsilon): This method addresses productions that derive the empty string ('eps'). It iterates through the grammar, identifying non-terminal symbols that produce epsilon and then eliminates those productions from the grammar.



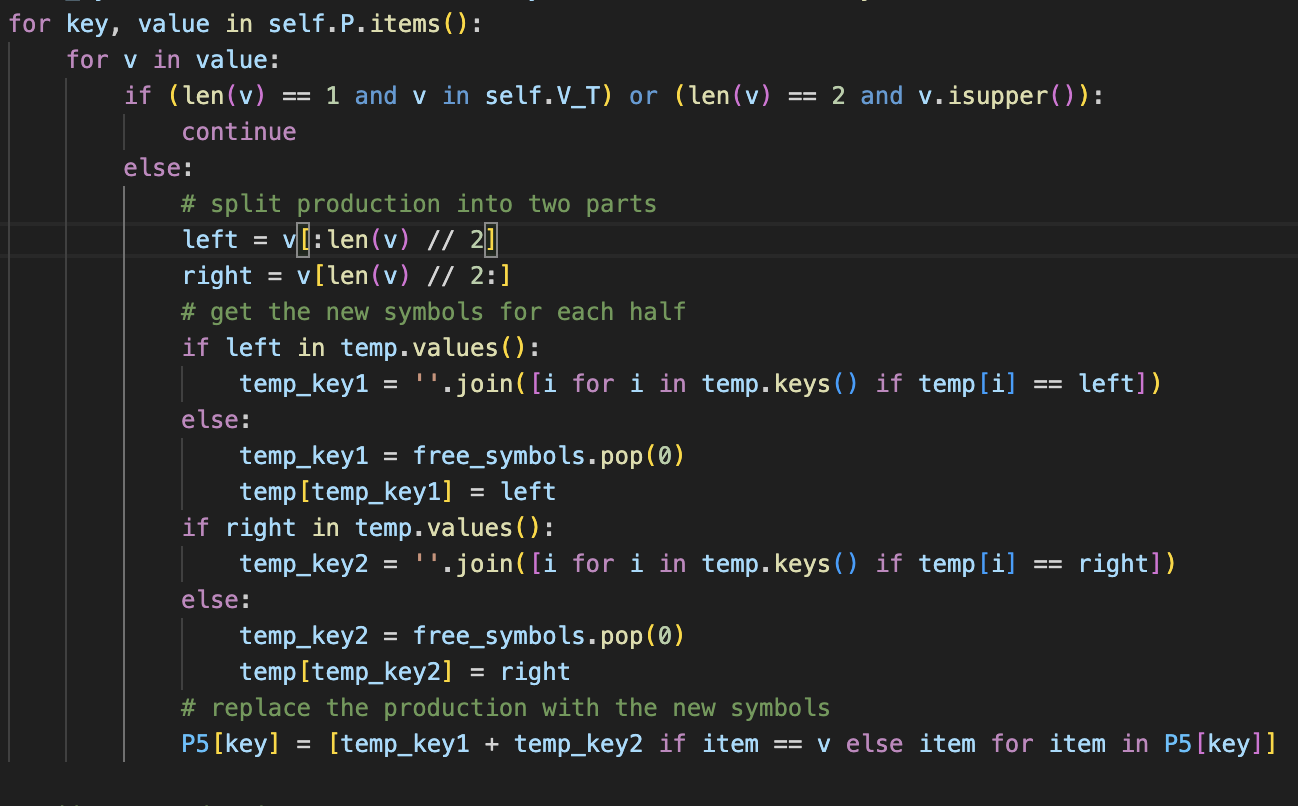
Eliminate Unit Productions (Eliminate\_Unit\_Prod): This method focuses on removing unit productions, i.e., productions where a non-terminal directly produces another non-terminal. It iterates through the grammar, replacing unit productions with equivalent non-unit productions until none remain.



Eliminate Inaccessible Symbols (Eliminate\_Inaccesible\_Symbols): This method targets symbols that are inaccessible from the start symbol. It traverses the grammar, identifying and removing symbols that cannot be reached from the start symbol through any sequence of productions.



Remove Nonproductive Symbols (Remove\_Nonproductive): This method handles symbols that are nonproductive, meaning they cannot derive any terminal string. It identifies and removes such symbols, ensuring that the grammar generates only useful strings.



Chomsky Normal Form (Chomsky\_Normal\_Form): This method transforms the grammar into Chomsky Normal Form (CNF), a canonical form where each production consists of either two non-terminals or a terminal symbol. It achieves this by systematically converting productions into the desired form.

**Unit Tests:**

The normalization process aimed to transform the initial grammar into Chomsky Normal Form (CNF), a standardized form for context-free grammars. However, upon analyzing the results of each normalization step, discrepancies and unexpected outcomes were observed.



**Results** The initial grammar exhibited a diverse set of production rules governing the generation of strings in the language. Each production rule defined expansions for non-terminal symbols, including terminal and non-terminal symbols. This grammar served as the starting point for the normalization process.

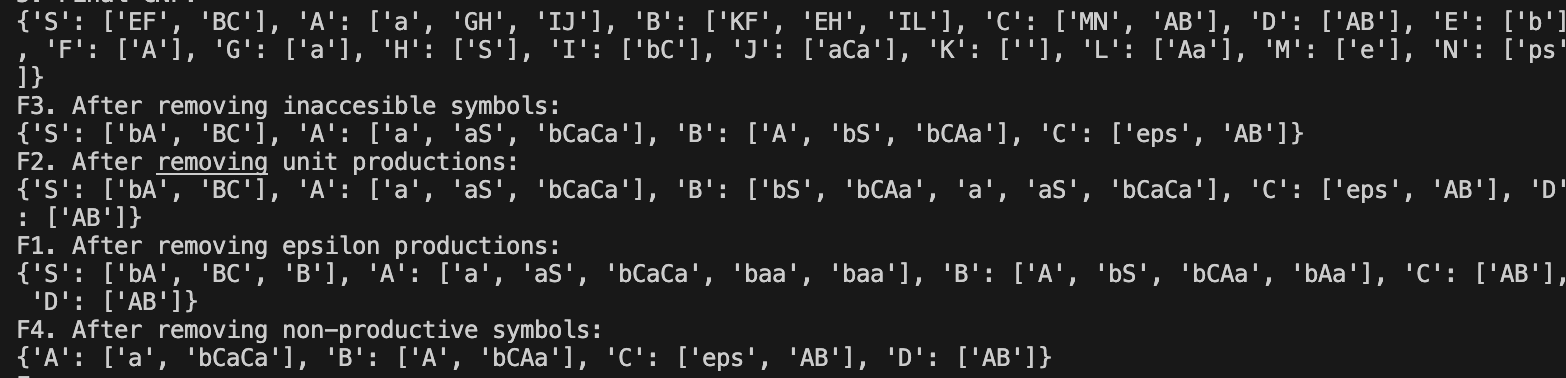
The removal of epsilon productions aimed to eliminate productions that derive the empty string. While the implementation successfully removed epsilon productions, the resulting grammar showed unexpected adjustments in certain production rules, indicating potential issues in the handling of epsilon removal.

Eliminating unit productions targeted the removal of productions where a non-terminal directly produces another non-terminal. Although the implementation successfully replaced unit productions with equivalent non-unit productions, discrepancies were observed in the resulting grammar compared to the expected outcome, indicating potential issues in the unit production elimination process.

The removal of inaccessible symbols aimed to eliminate symbols that were unreachable from the start symbol. While the implementation identified and removed inaccessible symbols, discrepancies were observed in the resulting grammar, suggesting potential issues in the identification or elimination process.

Removing nonproductive symbols aimed to eliminate symbols that cannot derive any terminal string, ensuring that the grammar generates only useful strings. The implementation successfully removed nonproductive symbols, but discrepancies were observed in the resulting grammar compared to the expected outcome, indicating potential issues in the nonproductive symbols removal process.

The final step involved transforming the grammar into Chomsky Normal Form (CNF), where each production consists of either two non-terminals or a terminal symbol. Although the implementation attempted to convert the grammar into CNF, discrepancies were observed in the final CNF representation compared to the expected outcome, indicating potential issues in the conversion process.

****

**Conclusion**

In the pursuit of transforming the provided grammar into Chomsky Normal Form (CNF) through a Python implementation of the context-free grammar normalization process, a series of challenges and limitations emerged. The removal of epsilon productions proved intricate, with unexpected adjustments in certain production rules indicating potential issues in the handling of epsilon productions. Despite efforts to eliminate unit productions, discrepancies in the resulting grammar hinted at limitations in the elimination process. While inaccessible symbols were successfully identified and removed, discrepancies in the resulting grammar raised concerns about the accuracy of the elimination process. Although nonproductive symbols were removed, discrepancies in the resulting grammar suggested potential issues in the removal process. Despite attempts to convert the grammar into CNF, discrepancies in the final CNF representation indicated potential challenges in the conversion process. To address these challenges, future directions include refining algorithms, conducting extensive testing, enhancing documentation, seeking community feedback, and collaborating for improvements. Through these efforts, the Python implementation can overcome its limitations and achieve its intended goal of effectively normalizing context-free grammars into CNF.