Grammar Parser

Hriscu Octavia & Hutupasu Stefana - 933/2

Link to git repo

Implementation:

- A class that manages reading a grammar from a given file and printing found non-terminals, terminals, productions and start symbols; it also implements a function to check whether or not the given grammar is context-free

Operations:

- **parse_line(line)**: Takes one parameter, line, which should be a string. Returns a list of values that are stripped from the line.
- **parse_productions(rules)**: Takes one parameter, rules, which should be a list of strings. Returns a dictionary with the left-hand side of the rule as the key and the right-hand side of the rule as a list of tuples containing the rule value and index.
- **from_file(fileName):** This function reads a text file containing the necessary information to create a Grammar object. It reads the first two lines of the file to create the nonterminals and terminals of the grammar, the third line to create the start symbol, and the rest of the file to create the productions of the grammar. It then returns a Grammar object using these values.
- **get_productions_for_nonterm(self, nonterm)**: This function takes a Grammar object (self) and a nonterminal symbol (nonterm) as arguments and returns a list of all the productions associated with the given nonterminal symbol. - If the given symbol is not a nonterminal symbol, an error will be raised.
- **isCFG(self)**: This function takes a Grammar object (self) as an argument and returns a boolean value. - This function will check the Grammar object for the following properties: (1) all nonterminal symbols have at least one associated production, (2) all keys in the productions dictionary are single-character strings. - If the Grammar object satisfies both of these properties, the function will return True. Otherwise, it will return False.