LFTC - Lab 5

https://github.com/samzirbo/Formal-Languages-and-Compiler-Design

Class **Grammar**:

The `Grammar` class represents a context-free grammar (CFG) and provides methods to work with grammatical rules and productions.

Fields:

- `nonterminals`: list to store nonterminal symbols in the grammar
- `terminals`: list to store terminal symbols in the grammar
- `productions`: dictionary where keys are nonterminals, and values are lists of production rules associated with each nonterminal
- `start`: starting symbol for the grammar
- `productionNo`: dictionary to store a unique number for each production rule. The keys are tuples containing nonterminal and the concatenated string of its production rule symbols

Methods:

- `__init__(self, path: str)`: The constructor method that initializes the grammar using a JSON file specified by the `path`. The JSON file is expected to contain keys 'NonTerminals', 'Terminals', 'Productions', and 'Start'.
  - Reads nonterminals, terminals, productions, and start symbol from the JSON file.
  - Prints nonterminals, terminals, start symbol, and calls `getProductions()` to print the productions.
  - Calls `setProductionNo()` to assign unique numbers to each production rule.
- `getProduction(self, nonterminal: str) -> list`: Retrieves the list of production rules associated with a given nonterminal.
- `isCFG(self) -> bool`: Checks if the grammar adheres to the definition of a Context-Free Grammar (CFG). Verifies that the starting symbol is in the set of

productions and that all left-hand and right-hand sides of productions meet CFG criteria.

- `getProductions(self) -> None`: Prints all productions in a readable format
- `setProductionNo(self) -> None`: Assigns a unique number to each production rule and stores it in the `productionNo` dictionary.