GitHub link: https://github.com/Gabarsolon/FLCD-Parser

Grammar

read_grammar_from_file(*self*, file_path): Reads grammar from file. The non-terminals are on the first line, the terminals on the second, the start symbol on the third and the rest of the lines are the productions.

productions_for_a_given_non_terminal(self, non_terminal): Returns the productions of a given non-terminal if it’s valid.

cfg_check(*self*): Checks if all the productions come from a non-terminal

closure(*self*, analysis_element): Function that gets as a parameter a set of analysis elements and it returns another set of analysis elements containing the current set concatenated with all productions that are after the dot for a non terminal

goto(*self*, analysisElements, symbol): Function that finds all the analysis elements from a state which have . right before a given symbol, moves that . after the symbol and calls the closure function on it

canonicalCollection(*self*): Function that builds the canonical collection based on a new starting symbol

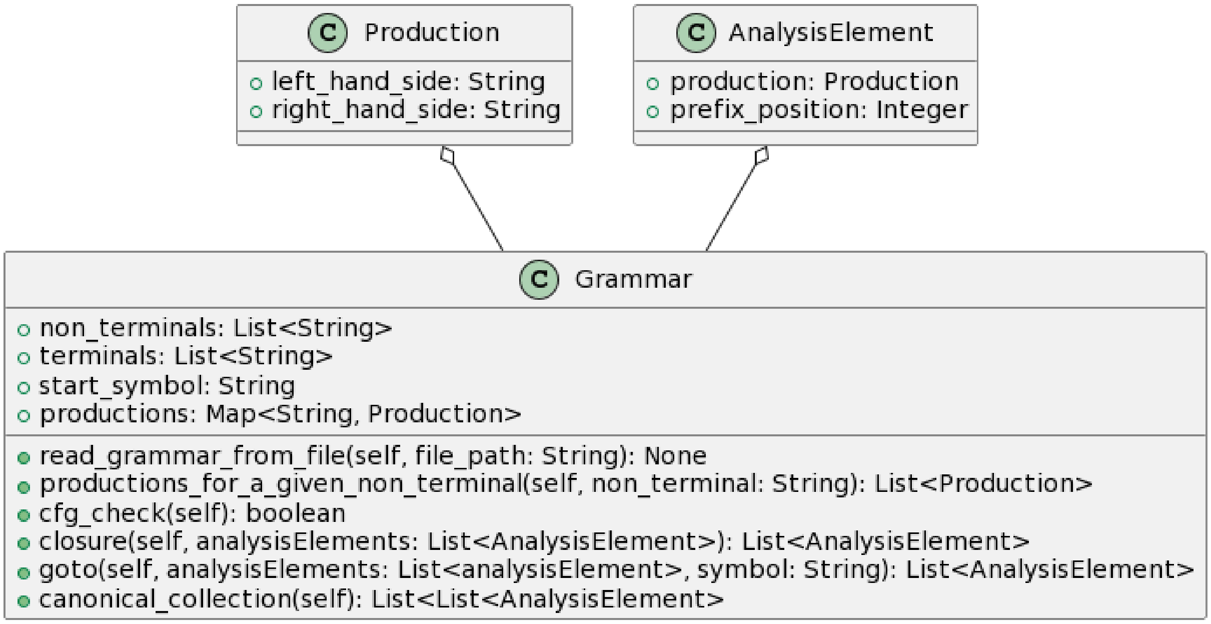
Production

```
def __init__(self, left_hand_side, right_hand_side):
    self.left_hand_side = left_hand_side
    self.right_hand_side = right_hand_side
```

Analysis element

```
def __init__(self, production, prefix_position):

    self.production = production
    self.prefix_position = prefix_position
```


C Production
left_hand_side: String
right_hand_side: String
C AnalysisElement
production: Production
prefix_position: Integer
C Grammar
non_terminals: List<String>
terminals: List<String>
start_symbol: String
productions: Map<String, Production>
read_grammar_from_file(self, file_path: String): None
productions_for_a_given_non_terminal(self, non_terminal: String): List<Production>
cfg_check(self): boolean
closure(self, analysisElements: List<AnalysisElement>): List<AnalysisElement>
goto(self, analysisElements: List<analysisElement>, symbol: String): List<AnalysisElement>
canonical_collection(self): List<List<AnalysisElement>