Documentation for ContextFreeGrammar Class

Github: https://github.com/daniel357/flcd

Overview

The **ContextFreeGrammar** class is designed to represent, manipulate, and analyze context-free grammars (CFGs) in Python. This class provides functionalities to load grammars from files, access different components of the grammar (like terminals and non-terminals), and check the validity of the grammar.

Class Methods

__init__(self)

Initializes a new instance of the **ContextFreeGrammar** class.

- **Attributes**:
    - **non_terminals**: List of non-terminal symbols in the grammar.
    - **terminals**: List of terminal symbols in the grammar.
    - **rules**: Dictionary storing the production rules.
    - **start_symbol**: The starting symbol of the grammar.

terminals_list(self)

Returns the list of terminal symbols in the grammar.

- **Returns**: List of strings representing terminal symbols.

non_terminals_list(self)

Returns the list of non-terminal symbols in the grammar.

- **Returns**: List of strings representing non-terminal symbols.

start_sym(self)

Returns the starting symbol of the grammar.

- **Returns**: String representing the start symbol.

productions_for(self, non_terminal)

Fetches the production rules for a given non-terminal symbol.

- **Parameters**:
    - **non_terminal** (str): The non-terminal symbol to get productions for.
- **Returns**: List of tuples, each representing a production rule.

has_additional_production(self, non_terminal, production_number)

Checks if there is an additional production rule for a given non-terminal symbol.

- **Parameters**:
    - **non_terminal** (str): Non-terminal symbol to check.
    - **production_number** (int): Current production number.
- **Returns**: Boolean. **True** if there is another production, **False** otherwise.

specific_production(self, non_terminal, production_number)

Retrieves a specific production rule for a given non-terminal symbol.

- **Parameters**:
    - **non_terminal** (str): Non-terminal symbol to get the production for.
    - **production_number** (int): The production number to retrieve.
- **Returns**: Tuple representing the specific production rule, or **None** if not found.

load_grammar(self, file_path)

Loads a grammar from the specified file.

- **Parameters**:
    - **file_path** (str): Path to the file containing the grammar.
- **Raises**: **ValueError** if the grammar is not a valid context-free grammar.

display_non_terminals(self)

Returns a string representation of non-terminal symbols in the grammar.

- **Returns**: String of non-terminal symbols.

display_terminals(self)

Returns a string representation of terminal symbols in the grammar.

- **Returns**: String of terminal symbols.

display_start_symbol(self)

Returns a string representation of the start symbol of the grammar.

- **Returns**: String of the start symbol.

display_productions(self)

Returns a string representation of the production rules in the grammar.

- **Returns**: String of production rules.

_parse_line(line)

Internal Method: Parses a line from the grammar file.

- **Parameters**:
    - **line** (str): Line from the grammar file.
- **Returns**: List of symbols extracted from the line.

_interpret_rules(rule_lines)

Internal Method: Interprets and organizes production rules from the file.

- **Parameters**:
    - **rule_lines** (list of str): Lines from the grammar file representing the rules.
- **Returns**: Dictionary of interpreted production rules.

_is_valid_cfg(rules)

Internal Method: Checks if the parsed grammar is a valid context-free grammar.

- **Parameters**:
    - **rules** (list of str): List of rules to be checked.
- **Returns**: Boolean. **True** if valid CFG, **False** otherwise.