https://github.com/alexreitler/FLCD_Lab5/tree/parser

The `RecursiveDescendant` class implements a recursive descent parser, a top-down parsing technique commonly used for parsing context-free grammars. It facilitates parsing a sequence of tokens based on a given grammar and produces a parser output.

Constructor

- `RecursiveDescendant(grammar: Grammar)`: Initializes the parser with a provided grammar. The initial configuration is set up with an epsilon symbol on both the working stack and input stack.

Public Methods

1. `run(w: List<String>): List<String>`:
   - **Parameters:** `w` - List of input tokens.
   - **Returns:** List of strings representing the working stack after parsing.
   - **Description:** Runs the recursive descent parsing algorithm on the input sequence `w`. It processes the input based on the grammar and updates the parser configuration accordingly.
2. `expand()`:
   - **Description:** Expands a nonterminal on the input stack by replacing it with the corresponding production rule from the grammar.
3. `advance()`:
   - **Description:** Advances the parser position and pushes the current token from the input stack to the working stack.
4. `momentaryInsuccess()`:
   - **Description:** Sets the parser state to indicate momentary insuccess, triggering backtracking.
5. `back()`:
   - **Description:** Backtracks by moving the parser position back and restoring the previous state from the working stack to the input stack.
6. `anotherTry()`:
   - **Description:** Handles error recovery by considering alternative production rules for the current nonterminal and adjusting the parser state and stacks accordingly.
7. `success()`:
   - **Description:** Marks successful completion of parsing by setting the parser state to "f" (finished).

The `Configuration` class represents the state of the parser during its execution. It includes information such as the parser state, position in the input sequence, and the contents of the working and input stacks.

Constructor

- `Configuration(state: String, position: Int, workingStack: Stack<String>, inputStack: Stack<String>)`: Initializes a parser configuration with the provided state, position, working stack, and input stack.

Properties

- `state: String`: Represents the current state of the parser ("q" for processing, "b" for backtracking, "e" for end, and "f" for finished).
- `position: Int`: Represents the current position in the input sequence.
- `workingStack: Stack<String>`: Represents the working stack of the parser.
- `inputStack: Stack<String>`: Represents the input stack of the parser.

Methods

- `toString(): String`: Overrides the `toString` method to provide a string representation of the configuration for debugging purposes.