Lab 7

Github Link

https://github.com/rusuraluca/lftc/tree/main/lftc_lab7

Docs

`class Production`: represents a production

`lhs`

- left-hand side of the production

`rhs`

- right-hand side of the production

`class State`: represents a state in a parsing algorithm

`prod`

- production of the state

`index`

- current index in the production's right-hand side (rhs)

`string_after_point`

- the first term after the point

- `shift_dot_right`
  - returns a new `State` with the index moved to the right, after the next space

---

`class Grammar`: representing and managing context-free grammars (CFGs)

`non-terminals`

`terminals`

`starting symbol`

`productions`

- dictionary of productions

`cfg`

- flag indicating whether it's a CFG

`enum Action`

`SHIFT`

`ACCEPT`

`REDUCE`

`class PIFField`: represents an object in a Program Internal Form (PIF)

`key`

- identifier or constant in the PIF

`token_index`

- an integer representing the type of token (e.g., identifier, constant)

`table_index`

- index of the token in the symbol table

`class PIFReader`: used for reading and managing the Program Internal Form (PIF)

`class Node`: represents a node in the Parsing Tree

`child`

- the child of the node

`right_sibling`

- the right sibling of the node

`value`

- the information stored in the node

`depth`

- depth in the tree

`class ParsingTree`:

`head`

- the head of the tree

- `def search_parent`
  - searches the rightmost node with the given value, that has no children, starting from the given node
- `def add_production`
  - adds a new production, the lhs is considered the father (the head of the tree if there is none, otherwise the search_parent(head, lhs)), the rhs is split in terms, that are added as children (the first child) or right_sibling (the rest of the children)
- `def process_parser_output`
  - gets a list of productions and creates the representation
- `def print_to_file`
  - prints the parsing tree to a file

`class ParsingTable`

`table`

- the parsing table (list(dict (action: Action, reduction: production_no, goto: set (state_no))))

- `def add_set()`
  - adds a new set of states to the parsing table
- `def process_canonical_collection`

    - add all the sets in a canonical collection to the parsing table
- `def get_action_for_set()`
    - get the action given a set_no
- `def get_goto_destination()`
    - get the destination given the set number and term
- `def get_reduction()`
    - get the production for the reduction given the set number
- `def get_productions_numbering()`
    - returns a list with the productions (indexes are used as numbers for the reductions)

`class LR0Parser`: represents an implementation of the LR(0) parser

`grammar`

- grammar based on which the parser will operate

`pt`

- parsing table

- `def closure(self, states)`
    - computes the closure of a set of states in the LR(0) automaton
- `def goto(self, states, term)`

    - computes the closure of the set formed by all the states in the original set that have the first string after the point the given term, with the point shifted after it
- `def canonical_collection(self)`
    - computes the canonical collection of LR(0) items for the parser
- `def construct_parsing_table(self)`
    - constructs the parsing table for the LR(0) parser based on the grammar
    - initializes the `self.pt` with the constructed table
- `def parse(self, s)`
    - parses a given string using the LR(0) parser
    - constructs the parsing table if not already done
    - performs the parsing actions (shift, reduce, accept)
    - parses the given array, returns the parsing tree if valid, otherwise throws error