Laboratory 5 – FLCD

POP RUXANDRA PAULA : https://github.com/ruxipaula/flcd

POPA MIHAI ADRIAN: https://github.com/adrian-popa/flcd

GRAMMAR CLASS

Class that contains 4 fields:

- nonTerminals : list of non-terminals
- terminals: list of the terminals
- startingSymbol: starting symbol
- productions: map (string -> list of list of strings)

LRITEM CLASS

Class that contains 3 fields:

- nonTerminal : string
- production: list of strings
- dotIndex: int – index that shows the position of the dot in the production

PARSER CLASS

Functions:

- **closure**: takes a list of LRItems as input and returns the closure of that list
- **goTo**: calls the closure function with the LRItems that match the corresponding state and symbol
- **canonicalCollection**: returns the canonical collection of the given grammar

LR0TABLE CLASS

Class that contains 2 fields:

- table : list of pairs containing a string(action) and a map that maps the symbol to the result of the goTo function(state index)
- list: list of strings containing all symbols from the grammar

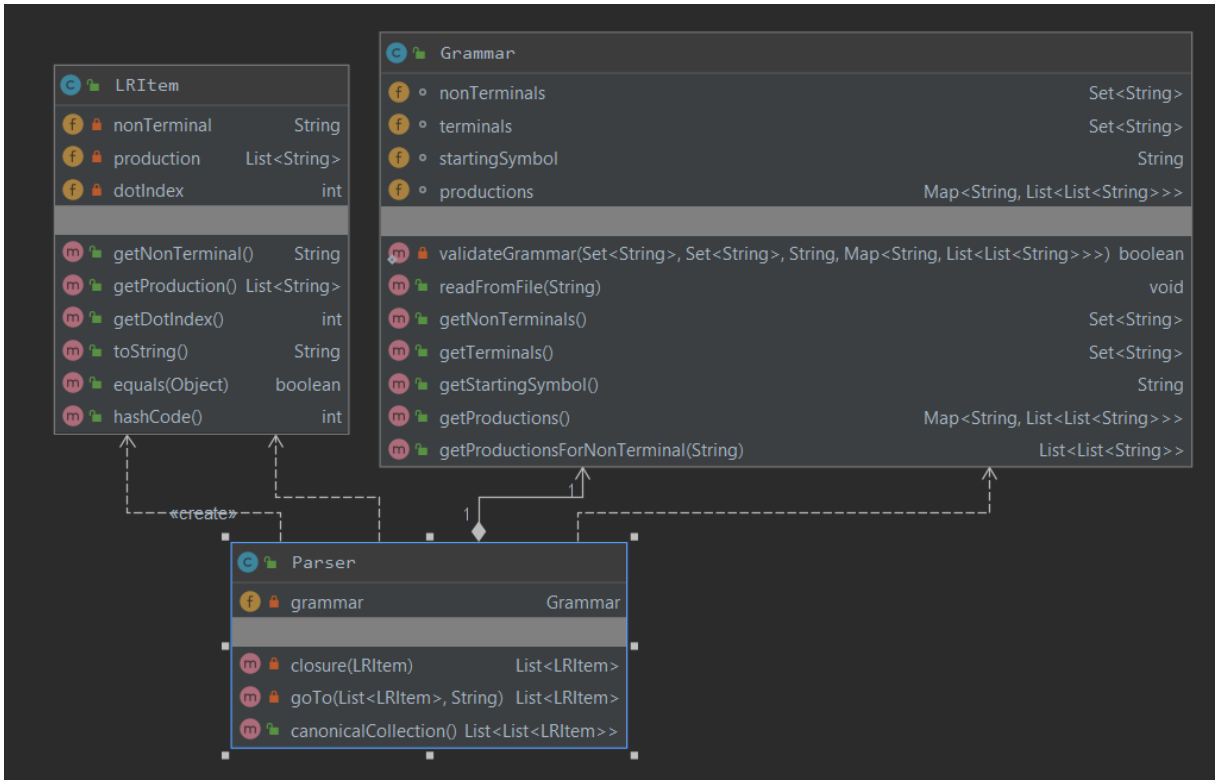
Functions:

- **cannonicalCollectionToLR0Table**: takes the cannonicalCollection and return the lr0table
- **we take each state from the cannonicalCollection and check if only one action(shift, reduce, acc) is appropriate**.
- **then we add the corresponding action to the table, mapping also the goTo symbol to the next state**

PRODUCTIONS

- We used a hashmap to represent the productions, keeping the non-terminals as the keys and the corresponding production as a list of lists that contain every symbol of the production.

CLASS DIAGRAM



EXAMPLES:

1.

- grammar.txt:

```
S A
a b c
S
S -> a A
A -> b A
A -> c
```

- Canonical Collection:

```
Canonical collection:
[[ S' -> .S ] , [ S -> .aA ] ]
[[ S' -> S. ] ]
[[ S -> a.A ] , [ A -> .bA ] , [ A -> .c ] ]
[[ S -> aA. ] ]
[[ A -> b.A ] , [ A -> .bA ] , [ A -> .c ] ]
[[ A -> c. ] ]
[[ A -> bA. ] ]
```

2.

- grammar.txt:

```
S
a b c
S
S -> b S | a S b | c
```

- Canonical Collection:

```
Canonical collection:
[[ S' -> .S ] , [ S -> .bS ] , [ S -> .aSb ] , [ S -> .c ] ]
[[ S' -> S. ] ]
[[ S -> a.Sb ] , [ S -> .bS ] , [ S -> .aSb ] , [ S -> .c ] ]
[[ S -> b.S ] , [ S -> .bS ] , [ S -> .aSb ] , [ S -> .c ] ]
[[ S -> c. ] ]
[[ S -> aS.b ] ]
[[ S -> bS. ] ]
[[ S -> aSb. ] ]
```

3.

LR0Table

```
LR(0) table:
Nr | Action    | goto
   |           | A | S | a | b | c |
0  | SHIFT     |   | 1 | 2 |   |   |
1  | ACC       |   |   |   |   |   |
2  | SHIFT     | 3 |   |   | 4 | 5 |
3  | REDUCE 1  |   |   |   |   |   |
4  | SHIFT     | 6 |   |   | 4 | 5 |
5  | REDUCE 3  |   |   |   |   |   |
6  | REDUCE 2  |   |   |   |   |   |
```

Sequence: abbc

```
Parsing successful..
1, 2, 2, 3,
```