Lab 7 Andrei Candet and Radu Ceaca

Link to git: https://github.com/cinnamonbreakfast/flcd/tree/main/lab5\_final\_%40raduceaca Assignment for a team of 2 students!

**+Run example on last page**

***Statement: Implement a parser algorithm (cont)***

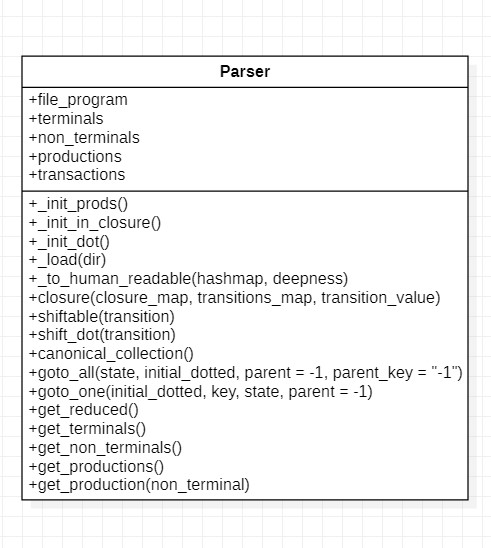
PART 2: Deliverables

1. Algorithm corresponding to parsing tables (if needed) and parsing strategy
2. Class ParserOutput - DS and operations corresponding to choice 2.a/2.b/2.c ([lab 5)](https://moodle.cs.ubbcluj.ro/mod/assign/view.php?id=2841) (required operations:

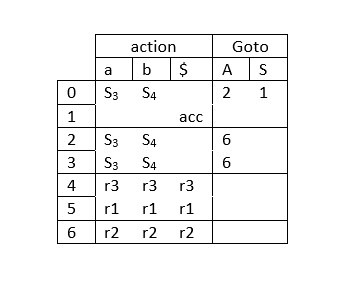
transform parsing tree into representation; print DS to screen and to file)

**Remark**:

- if the table contains conflicts, you will be helped to solve them. It is important to print message containing row (*symbol in LL(1), respectively state in LR(0)*) and column (symbol) where the conflict appears. For LL(1) values (αα,i) might also help



**+ parse\_string(string)**



\_init\_prods():

Initialize the production directory

\_init\_in\_closure()

Initialeze the closure map

\_init\_dot ()

Initialization method for closure

\_load(dir)

Load data from file closure(closure\_map, transitions\_map, transition\_value) Compute the closure\_map

shiftable(transition)

Check if the dot can be shifted

shift\_dot(transition)

Call the shiftable function and shift the dot

canonical\_collection()

Populate the canonical collection

goto\_all(state, initial\_dotted, parent=-1, parent\_key="-1")

Goes through every state

goto\_one(initial\_dotted, key, state, parent=-1)

Goes to a single state get\_reduced()

Returns the reduced map

get\_terminals()

Returns terminals

get\_non\_terminals()

Returns non terminals

get\_produtions()

Returns productions

get\_production(non\_terminal)

Returns the production of a non terminal

**parse\_string(string)**

**pre: string to parse**

**post: parsing table**

**Run Example**

Grammar:

a b c  
S  
S a S b S  
S a S  
S a

Sequence to be parsed: **aac**

**state 0**

**S' : [['.', 'S']]**

**S : [['.', 'a', 'S', 'b', 'S'], ['.', 'a', 'S'], ['.', 'c']]**

**state 1**

**S' : [['S', '.']]**

**state 2**

**S : [['a', '.', 'S', 'b', 'S'], ['.', 'a', 'S', 'b', 'S'], ['.', 'a', 'S'], ['.', 'c']]**

**state 3**

**S : [['a', '.', 'S'], ['.', 'a', 'S', 'b', 'S'], ['.', 'a', 'S'], ['.', 'c']]**

**state 4**

**S : [['c', '.']]**

**state 5**

**S : [['a', 'S', '.', 'b', 'S']]**

**state 6**

**S : [['a', 'S', '.']]**

**state 7**

**S : [['a', 'S', 'b', '.', 'S'], ['.', 'a', 'S', 'b', 'S'], ['.', 'a', 'S'], ['.', 'c']]**

**state 8**

**S : [['a', 'S', 'b', 'S', '.']]**

**S0 : {'S': 1, 'a': 3, 'c': 4}**

**S1 : {'$': 'accept'}**

**S2 : {'a': 3, 'c': 4, 'S': 5}**

**S3 : {'a': 3, 'c': 4, 'S': 6}**

**S4 : {'a': 'r3', 'b': 'r3', 'c': 'r3', '$': 'r3'}**

**S5 : {'b': 7}**

**S6 : {'a': 'r2', 'b': 'r2', 'c': 'r2', '$': 'r2'}**

**S7 : {'a': 3, 'c': 4, 'S': 8}**

**S8 : {'a': 'r1', 'b': 'r1', 'c': 'r1', '$': 'r1'}**

|  |  |  |
| --- | --- | --- |
| **Work Stack** | **Input Stack** | **Output band** |
| **$0** | **aac$** | **Empty** |
| **$0a3** | **ac$** | **Empty** |
| **$0a3a3** | **c$** | **Empty** |
| **$0a3a3c4** | **$** | **Empty** |
| **$0a3a3S6** | **$** | **3** |
| **$0a3S6** | **$** | **2,3** |
| **$0S1** | **$** | **2,2,3** |
| **accepted** | **$** | **2,2,3** |

**Canonical Table**

| **State** | **Action a** | **Action b** | **Action c** | **Action $** | **Go to S** | **Go to A** |
| --- | --- | --- | --- | --- | --- | --- |
| S0 | 2 |  |  |  | 1 |  |
| S1 |  |  |  | accept |  |  |
| S2 |  | 4 | 5 |  |  | 3 |
| S3 | r1 | r1 | r1 | r1 |  |  |
| S4 |  | 4 | 5 |  |  | 6 |
| S5 | r3 | r3 | r3 | r3 |  |  |
| S6 | r2 | r2 | r2 | r2 |  |  |