Documentatie LFDC

Opris Vlad Marius 935, Socaciu Mihai 938

**Github link:** <https://github.com/Socca98/LFTC-2020>

Lab6 – Parser LR(0)

**Requirements**

***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

Parsing strategy: LR(0)

Parser.py

LR(0) parsing algorithm bottom-up to construct the syntax tree.

**Properties**:

grammar – the Grammar

workingStack – stack

inputStack – stack

output – stack

Methods:

**closure**(productions):

Takes a state containing productions.

In: productions – List of productions for closure

Out: closure

**go\_to**(state, symbol):

Transition from a state to another using a terminal or non-terminal.

in: state – String

symbol – String

out: a closure of a list

**canonical\_collection**():

Construct a set of states.

C - canonical collection

ex: [('S1', ['.', 'program']), ]

out: Collection of states

**generate\_table**(state, symbol):

Generates the parsing table used to check the input tokens.

out: parsing table (list of dictionaries)

**parse**(input\_string):

Using the parsing table we check the input stack if is syntactically correct.

in: inputStack, obtained from PIF (list of strings)

out: output, list of integers representing reduce states

representation of the parsing tree = derivations string

ParserOutput.py

Fields:

-parser (the LR(0) parser)

Methods:

-**derivations\_string**(output\_parser):

Takes a list containing production rules needed to apply to obtain the input string. List of derivations. (right most derivation)

in: output\_parser, list of integers representing reduce states  
out: list of production rules read from grammar

-**print\_to\_console**(output\_parser):

Print the derivations string to console

in: output\_parser, list of integers representing reduce states

-**write\_to\_file**(output\_parser):

Print the derivations string to a file (derivations\_string.out)

in: output\_parser, list of integers representing reduce states

**The UML diagram**

