LAB 5 – Antonio Suciu, 937/1

https://github.com/AntonioSuciu/FLCD/tree/main/Labs/Lab%205

Statement: Implement a parser algorithm

1. One of the following parsing methods will be chosen (assigned by teaching staff):

   1.a. recursive descendent

   1.b. ll(1)

   1.c. lr(0)

2. The representation of the parsing tree (output) will be (decided by the team):

   2.a. productions string (max grade = 8.5)

   2.b. derivations string (max grade = 9)

   2.c. table (using father and sibling relation) (max grade = 10)

PART 1: Deliverables

1. *Class Grammar* (required operations: read a grammar from file, print set of nonterminals, set of terminals, set of productions, productions for a given nonterminal, CFG check)
2. Input files: *g1.txt* (simple grammar from course/seminar), *g2.txt* (grammar of the minilanguage - syntax rules from Lab 1b)

CLASS Grammar:

N – set of nonterminals

Sigma - set of terminals

P – production rules, map between the nonterminal and its productions

S – starting symbol

CFGcheck(): checks whether the grammar is context-free:

- Verifies if the starting symbol is part of the left-hand side
- If not => not a cfg
- For each left-hand side, if there is more than one symbol => not a cfg
- For each right-hand side, if the symbol is not a nonterminal / part of the alphabet / epsilon => not a cfg

LAB 6

PART 2: **Deliverables**

Functions corresponding to the assigned parsing strategy + appropriate tests, as detailed below:

LL(1) - functions *FIRST*, *FOLLOW*

CLASS Parser:

-

ConcatSizeOne():

generateFirst(): finds the First(X) for each X – nonterminal

generateFollow(): finds the Follow(X) for each X – nonterminal

generateParseTable()

analyseSequence()

LAB 7
PART 3: Deliverables

1. Algorithms corresponding to *parsing table* (if needed) and *parsing strategy*

2. Class *ParserOutput* - DS and operations corresponding to choice 2.a/2.b/2.c (Lab 5) (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 a message containing row (symbol in LL(1), respectively state in LR(0)) and column (symbol) where the conflict appears. For LL(1), values (α,i) might also help.

Class PARSEROUTPUT:

Parser: the parser

Productions: the result of the analysis

hasErrors: a flagship of the errors in the productions

outputfile: the output file

generateTree(): generates the tree

example of grammar:

```
N = { S A B C D }
Sigma = { a + * ( ) }
S = S
P = {
S -> B A
A -> + B A | epsilon
B -> D C
C -> * D C | epsilon
D -> ( S ) | a
}
```

Its parse table:

(*,)) -> (err,-1)

(A,a) -> (err,-1)

(*,*) -> (pop,-1)

(C,$) -> (epsilon,5)

(*,+) -> (err,-1)

(C,() -> (err,-1)

(C,)) -> (epsilon,5)

(C,*) -> (* D C,6)

(C,+) -> (epsilon,5)

(),a) -> (err,-1)

(+,$) -> (err,-1)

(+,() -> (err,-1)

(+,)) -> (err,-1)

(B,a) -> (D C,3)

(+,*) -> (err,-1)

(D,$) -> (err,-1)

(+,+) -> (pop,-1)

(a,$) -> (err,-1)

(D,() -> ((S),8)

(D,)) -> (err,-1)

(D,*) -> (err,-1)

(D,+) -> (err,-1)

(a,() -> (err,-1)

(a,)) -> (err,-1)

(a,*) -> (err,-1)

(a,+) -> (err,-1)

(S,a) -> (B A,4)

(*,a) -> (err,-1)

((,$) -> (err,-1)

(C,a) -> (err,-1)

((,() -> (pop,-1)

($,$) -> (acc,-1)

((,)) -> (err,-1)

((,*) -> (err,-1)

(A,$) -> (epsilon,1)

((,+) -> (err,-1)

($,() -> (err,-1)

($,)) -> (err,-1)

($,*) -> (err,-1)

(A,() -> (err,-1)

($,+) -> (err,-1)

(A,)) -> (epsilon,1)

(A,*) -> (err,-1)

(A,+) -> (+ B A,2)

(+,a) -> (err,-1)

(),$) -> (err,-1)

(D,a) -> (a,7)

(),() -> (err,-1)

(a,a) -> (pop,-1)

(),)) -> (pop,-1)

(),*) -> (err,-1)

(B,$) -> (err,-1)

(),+) -> (err,-1)

(B,() -> (D C,3)

(B,)) -> (err,-1)

(B,*) -> (err,-1)

(B,+) -> (err,-1)

(S,$) -> (err,-1)

(S,() -> (B A,4)

(S,)) -> (err,-1)

(S,*) -> (err,-1)

(S,+) -> (err,-1)

((,a) -> (err,-1)

(*,$) -> (err,-1)

($,a) -> (err,-1)

(*,() -> (err,-1)

The sequence:

```
a * ( a + a )
```

The output of the analyzed sequence:

Index	Value	Parent	Sibling
1	S	0	0
2	B	1	0
3	A	1	2
4	D	2	0
5	C	2	4
6	a	4	0
7	*	5	0
8	D	5	7
9	C	5	8
10	(	8	0
11	S	8	10
12	)	8	11
13	B	11	0
14	A	11	13
15	D	13	0
16	C	13	15
17	a	15	0
18	epsilon	16	0
19	+	14	0
20	B	14	19
21	A	14	20
22	D	20	0
23	C	20	22
24	a	22	0
25	epsilon	23	0
26	epsilon	21	0
27	epsilon	9	0
28	epsilon	3	0

The tree:

[4, 3, 7, 6, 8, 4, 3, 7, 5, 2, 3, 7, 5, 1, 5, 1]