Μεταγλωττιστές 2019

Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Ηλίας Τζάνης

Α.Μ.: Π2016083

Α) Κανόνες γραμματικής

Enter a grammar:

```
Stmt_list -> Stmt Stmt_list|.
Stmt -> id equal Exp| print Exp.
Exp -> Term Term_tail.
Term_tail -> xor Term Term_tail | .
Term -> Factor Factor_tail.
Factor_tail -> or Factor Factor_tail | .
Factor -> Atom Atom_tail.
Atom_tail -> and Atom Atom_tail | .
Atom -> (exp) | id | number.
Xor -> xor .
And -> and .
Or -> or .
```

View Vital Statistics

Β) Αποτελέσματα ελέγχου για LL(1) συμβατότητα

Grammar

```
Stmt_list →  Stmt Stmt_list
           | .
Stmt →       id equal Exp
           | print Exp.
Exp →        Term Term_tail.
Term_tail →  xor Term Term_tail
           | .
Term →       Factor Factor_tail.
Factor_tail →or Factor Factor_tail
           | .
Factor →     Atom Atom_tail.
Atom_tail →  and Atom Atom_tail
           | .
Atom →       (exp)
           | id
           | number.
Xor →        xor .
And →        and .
Or →         or .
```

Some sentences generated by this grammar: {ε, print id, print (exp), id equal id, print number, id equal (exp), id equal number, id equal id and id, print (exp) and id, print id and (exp), id equal id and (exp), print (exp) and (exp), id equal (exp) and id, id equal id and number, id equal number and id, print (exp) and number, id equal (exp) and (exp), id equal number and (exp), id equal (exp) and number, id equal number and number}

- You have unreachable nonterminals in your grammar. They are: Xor And Or
- The nullable nonterminals are: Stmt_list Term_tail Factor_tail Atom_tail.
- The endable nonterminals are: Atom_tail Atom Factor_tail Factor Term_tail Term Exp Stmt_list Stmt.
- No cycles.

Γ) Αποτελέσματα ελέγχου για LL(1) συμβατότητα

nonterminal	first set	follow set	nullable	endable
Stmt_list	id print	∅	yes	yes
Stmt	id print	id print	no	yes
Exp	(exp) id number	id print	no	yes
Term_tail	xor	id print	yes	yes
Term	(exp) id number	xor id print	no	yes
Factor_tail	or	xor id print	yes	yes
Factor	(exp) id number	or xor id print	no	yes
Atom_tail	and	or xor id print	yes	yes
Atom	(exp) id number	and or xor id print	no	yes
Xor	xor	∅	no	no
And	and	∅	no	no
Or	or	∅	no	no

The grammar is LL(1).

- attempt to transform the grammar (to LL(1))
- generate LL(1) parsing table
- generate LR(0)/SLR(1) automaton
- generate LALR(1) automaton
- generate LR(1) automaton

Return home to enter a new grammar.

Δ) Αποτελέσματα εξόδου για έγκυρες μορφές εισόδους

```
a = 1001
b = 101 XOR 111
c = 1001100 AND 0001001
print (10101 AND 10111 OR 11001 XOR 00001)
print a
print 1 OR 0 XOR 1
print c
print b
```

```
ilias@iliastz:~$ cd Desktop/
ilias@iliastz:~/Desktop$ python3 runner.py
11100
1001
0
1000
10
```

Ε) Αποτελέσματα εξόδου για άκυρη μορφή εισόδου

```
b = a XOR 111
c = 1001100 AND 0001001
print (10101 AND 10111 OR 11001 XOR 00001)
print a
print 1 OR 0 XOR 1
print c
print b
```

```
ilias@iliastz:~/Desktop$ python3 runner.py
Traceback (most recent call last):
  File "runner.py", line 133, in <module>
    parser.parse(fp)
  File "runner.py", line 45, in parse
    self.stmt_list()
  File "runner.py", line 49, in stmt_list
    self.stmt()
  File "runner.py", line 60, in stmt
    e=self.expr()
  File "runner.py", line 70, in expr
    func=self.term()
  File "runner.py", line 85, in term
    func=self.factor()
  File "runner.py", line 99, in factor
    func=self.atom()
  File "runner.py", line 123, in atom
    raise ParseRun("perimenw id arxikopoiimeno")
__main__.ParseRun: perimenw id arxikopoiimeno
```