Μεταγλωττιστές 2019

2η Προγραμματιστική Εργασία

Γεώργιος Μαντέλλος
Π2016149

Μάιος 2019

1. Οι κανόνες της (LL(1)) γραμματικής

```
Stmt_list       ->      Stmt Stmt_list | .
Stmt            ->      id assign Expr | print Expr.
Expr            ->      Atom Atom_tail.
Atom_tail       ->      xor Atom Atom_tail | .
Atom            ->      Term Term_tail.
Term_tail       ->      or Term Term_tail | .
Term            ->      Factor Factor_tail.
Factor_tail     ->      and Factor Factor_tail | .
Factor          ->      (Expr) | id | binary.
```

2. Το αποτέλεσμα ελέγχου για LL(1) συμβατότητα

Link για τα αναλυτικά αποτελέσματα από το online εργαλείο
Παρακάτω φαίνεται μια ενδεικτικη φωτογραφια του αποτελέσματος

Grammar

```
Stmt_list →   Stmt Stmt_list
            | .
Stmt →        id assign Expr
            | print Expr.
Expr →        Atom Atom_tail.
Atom_tail →   xor Atom Atom_tail
            | .
Atom →        Term Term_tail.
Term_tail →   or Term Term_tail
            | .
Term →        Factor Factor_tail.
Factor_tail → and Factor Factor_tail
            | .
Factor →      (Expr)
            | id
            | binary.
```

Some sentences generated by this grammar: {ε, print id, id assign id and (Expr), print (Expr) and (Expr), id assig

- All nonterminals are reachable and realizable.
- The nullable nonterminals are: Stmt_list Atom_tail Term_tail Factor_tail.
- The endable nonterminals are: Factor_tail Factor Term_tail Term Atom_tail Atom Expr Stmt_list Stmt.
- No cycles.

3. Πίνακες με τα FIRST και FOLLOW sets για όλα τα μη τερματικά σύμβολα

nonterminal	first set	follow set	nullable	endable
Stmt_list	id print	∅	yes	yes
Stmt	id print	id print	no	yes
Expr	(Expr) id binary	id print	no	yes
Atom_tail	xor	id print	yes	yes
Atom	(Expr) id binary	xor id print	no	yes
Term_tail	or	xor id print	yes	yes
Term	(Expr) id binary	or xor id print	no	yes
Factor_tail	and	or xor id print	yes	yes
Factor	(Expr) id binary	and or xor id print	no	yes

The grammar is LL(1).

4. Αποτελέσματα εξόδου για έγκυρες και άκυρες μορφές εισόδου.

Έγκυρες μορφές:

- **Αρχείο έγκυρων μορφών εισόδου**

```
print 00000001 and 00000010
print 00000011 or 00000100 xor 00000101
a = 00000110
b = 00000111
c = 00000100 xor (a or 00000100 and ((b and a) xor b))
print a and b
print (b xor a)
print c
print c xor b and (c or a xor (a and b))
print 1 or 11
```

- **Αποτελέσματα των παραπάνω έγκυρων μορφών**

```
[geocfu@cassandra compilers1819a2]$ python3 runner.py
00000000
00000010
00000110
00000001
00000010
00000010
00000011
```

Μη έγκυρες μορφές:

- **Αρχείο μη έγκυρων μορφών εισόδου**

```
print and 01 xor xor 1
```

- **Αποτελέσματα των παραπάνω μη έγκυρων μορφών**

```
[geocfu@cassandra compilers1819a2]$ python3 runner.py
Traceback (most recent call last):
  File "runner.py", line 135, in <module>
    parser.parse(fp)
  File "runner.py", line 47, in parse
    self.statement_list()
  File "runner.py", line 58, in statement_list
    self.statement()
  File "runner.py", line 74, in statement
    print('{:08b}'.format(self.expression()))
  File "runner.py", line 88, in expression
    raise ParseError('Expected ( or id or binary in expression()')
__main__.ParseError: Expected ( or id or binary in expression()
```

~~~~~~

- **Αρχείο μη έγκυρων μορφών εισόδου**

```
a = print 1
```

- **Αποτελέσματα των παραπάνω μη έγκυρων μορφών**

```
[geocfu@cassandra compilers1819a2]$ python3 runner.py
Traceback (most recent call last):
  File "runner.py", line 135, in <module>
    parser.parse(fp)
  File "runner.py", line 47, in parse
    self.statement_list()
  File "runner.py", line 58, in statement_list
    self.statement()
  File "runner.py", line 70, in statement
    e = self.expression()
```
~~~~~~

```
  File "runner.py", line 88, in expression
    raise ParseError('Expected ( or id or binary in expression()')
__main__.ParseError: Expected ( or id or binary in expression()
```

~~~~~~

- **Αρχείο μη έγκυρων μορφών εισόδου**

```
print c
```

- **Αποτελέσματα των παραπάνω μη έγκυρων μορφών**

```
[geocfu@cassandra compilers1819a2]$ python3 runner.py
Traceback (most recent call last):
  File "runner.py", line 135, in <module>
    parser.parse(fp)
  File "runner.py", line 47, in parse
    self.statement_list()
  File "runner.py", line 58, in statement_list
    self.statement()
  File "runner.py", line 74, in statement
    print('{:08b}'.format(self.expression()))
  File "runner.py", line 80, in expression
    a = self.atom()
  File "runner.py", line 92, in atom
    t = self.term()
  File "runner.py", line 104, in term
    f = self.factor()
  File "runner.py", line 125, in factor
    raise RunError("lathos sto factor")
__main__.RunError: lathos sto factor
```
~~~~~~