ΚΑΘΟΔΙΚΗ ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ

Συντακτική ανάλυση αναδρομικής κατάβασης με Python

Ονοματεπώνυμο: Ντάνιελ Γιορντάνωβ
ΑΜ: Π2015105

Κανόνες Γραμματικής

Για την σχεδίαση της γραμματικής βασίστηκα στα παραδείγματα που υποδείχθηκαν στις οδηγίες της εργασίας, καθώς και σε μερικά παραδείγματα του βιβλίου. Η γραμματική καλύπτει όλα τα ζητούμενα της άσκησης και έχει ενσωματωθεί πλήρως στον κώδικα χωρίς τροποποιήσεις.

```
Stmt_list     →   Stmt Stmt_list | ε
Stmt          →   id = Expr | print Expr
Expr          →   Term Term_tail
Term_tail     →   xor Term Term_tail | ε
Term          →   Factor Factor_tail
Factor_tail   →   or Factor Factor_tail | ε
Factor        →   Atom Atom_tail
Atom_tail     →   and Atom Atom_tail | ε
Atom          →   (Expr) | bin | id
```

Συμβατότητα LL(1), Πίνακες FIRST και FOLLOW

Για να επαληθεύσω την συμβατότητα LL(1) της γραμματικής μου, χρησιμοποίησα το web εργαλείο που μας δόθηκε στις οδηγίες της άσκησης. Χρειάστηκε να κάνω μικρές αλλαγές στα σύμβολα για να γίνει δεκτή η είσοδός μου (π.χ. “Assign” αντί για “=”) αλλά σε γενικές γραμμές το εργαλείο ήταν εύκολο και απλό στη χρήση, και η επαλήθευση έγινε επιτυχώς. Τέλος, από το εργαλείο παράχθηκαν και τα ζητούμενα σύνολα FIRST και FOLLOW σε μορφή πίνακα.

Grammar

```
Stmt_list →   Stmt Stmt_list
            | .
Stmt →        id assign Expr
            | print Expr.
Expr →        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 →        (Expr)
            | bin
            | id.
```

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

- All nonterminals are reachable and realizable.
- 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 Expr Stmt_list Stmt.
- No cycles.

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

The grammar is LL(1).

Συνοπτική Περιγραφή του Κώδικα

Ξεκίνησα με το πρόγραμμα αναγνωριστή γλώσσας, `scanner.py`, το οποίο είναι βασισμένο στο ήδη υπάρχον παράδειγμα στο GitHub Gist. Η λειτουργία του είναι απλή και ακολουθεί τα ζητούμενα της άσκησης: Ανοίγει ένα απλό αρχείο κειμένου, ελέγχει αν ακολουθεί τους κανόνες της γραμματικής, και εμφανίζει ανάλογα μηνύματα σε περίπτωση σφαλμάτων. Το επόμενο πρόγραμμα, `runner.py`, αποτελεί επέκταση του πρώτου. Διερμηνεύει τις εντολές και εμφανίζει τα κατάλληλα αποτελέσματα στην οθόνη κατά την εκτέλεση.

Ο κώδικας και των δύο αρχείων είναι κατά την γνώμη μου ευανάγνωστος, και γενικά προσπάθησα να συμπεριλαμβάνω τα κατάλληλα σχόλια για κάθε διακριτό τμήμα του. Επίσης για την συγγραφή του κώδικα και για την περιγραφή της λογικής του, στηρίχθηκα πολύ στα σχήματα/διαγράμματα του βιβλίου στο κεφάλαιο 4.4 “Καθοδική Συντακτική Ανάλυση”, όπου τα σύμβολα της γραμματικής παρουσιάζονται ως επίπεδα σε ένα ανεστραμμένο δέντρο.

Αποτελέσματα Εξόδου

Έγκυρες Μορφές Εισόδου

Για αυτό το παράδειγμα, το αρχείο κειμένου περιέχει τις ακόλουθες εντολές/εκφράσεις:

```
print 10101010
print 10 and 10 xor 01
print 10 and (10 xor 01)
a = 1110 or 1
print a
b = 1001 and 1010 or (11 xor (10010 and 11110))
c = 11111111 xor 10101010
print c and b
a = 10
print a
```

```
PS C:\Users\Daniel\Desktop\Compilers\Assignment2> python.exe .\runner.py
10101010
11
10
1111
10001
10
```

Μη Έγκυρες Μορφές εισόδου

Σε περίπτωση που το κείμενο του αρχείου δεν ακολουθεί τους κανόνες της γραμματικής, το πρόγραμμα εμφανίζει τα ανάλογα μηνύματα λάθους. Ακολουθούν μερικά παραδείγματα:

```
print a
```

```
PS C:\Users\Daniel\Desktop\Compilers\Assignment2> python.exe .\runner.py
Parser Error: in atom: unrecognized variable name at line 1 char 8
```

```
a = 2019
```

```
PS C:\Users\Daniel\Desktop\Compilers\Assignment2> python.exe .\runner.py
Scanner Error: at line 1 char 5
```

```
a = (10010
```

```
PS C:\Users\Daniel\Desktop\Compilers\Assignment2> python.exe .\runner.py
Parser Error: found None instead of ) at line 1 char 11
```

```
print = 1010
```

```
PS C:\Users\Daniel\Desktop\Compilers\Assignment2> python.exe .\runner.py
Parser Error: in expr: "(", "id" or "bin" expected at line 1 char 7
```

Πηγές

1. Aho, Alfred V., et al. *Μεταγλωττιστές: Αρχές, Τεχνικές & Εργαλεία*. Translated by Παναγιώτης Αλεφραγκής et al., ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ, 2014.
2. *Context Free Grammar Tool*, http://smlweb.cpsc.ucalgary.ca/start.html
3. *Recursive descent parser Python3 example*, https://gist.github.com/mixstef/946fce67f49f147991719bfa4d0101fa