Μεταγλωττιστές

Προγραμματιστική Εργασία #2

Αλέξανδρος Ζερβόπουλος
ΑΜ: Π2015111

1 Οι κανόνες της γραμματικής

Η γραμματική σχεδιάστηκε ως εξής:

- Stmt_list → Stmt Stmt_list | ϵ
- Stmt → id = Expr | print Expr
- Expr → Term Term_Tail
- Term_Tail → Orop Term Term_Tail | ϵ
- Term → Factor Factor_Tail
- Factor_Tail → Andop Factor Factor_Tail | ϵ
- Factor → Value | Notop Value
- Value → (Expr) | id | bool
- Orop → or
- Andop → and
- Notop → not

Some sentences generated by this grammar: {ε, print id, print bool, print not id, id assign id, print not bool, id assign bool, print id and id, id assign not id, print id and bool, print bool and id, id assign not bool, id assign id and id, print bool and bool, id assign bool and id, id assign id and bool, id assign bool and bool, id assign id and not id, id assign id and not bool, id assign bool and not id}

Figure 1: Παραδείγματα προτάσεων που παράγονται από τη γραμματική

2 Αποτέλεσμα ελέγχου για LL(1) συμβατότητα και FIRST και FOLLOW sets

Η γραμματική είναι πράγματι LL(1). Η εγκυρότητα αυτού επαληθεύτηκε και με το προτεινόμενο εργαλείο. Τα αποτελέσματα του ελέγχου της γραμματικής μπορούν να βρεθούν εδώ.

- All nonterminals are reachable and realizable.
- The nullable nonterminals are: Stmt_list Term_Tail Factor_Tail.
- The endable nonterminals are: Value 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
Term_Tail	or	) id print	yes	yes
Term	(id bool not	) or id print	no	yes
Factor_Tail	and	) or id print	yes	yes
Factor	(id bool not	) and or id print	no	yes
Value	(id bool	) and or id print	no	yes
Expr	(id bool not	) id print	no	yes
Orop	or	(id bool not	no	no
Andop	and	(id bool not	no	no
Notop	not	(id bool	no	no

The grammar is LL(1).

Figure 2: Αποτελέσματα ελέγχου στο προτεινόμενο εργαλείο καθώς και τα FIRST και FOLLOW sets

3 Περιγραφή του parser

Ο parser κατασκευάστηκε με βάση το πρόγραμμα που αναφέρεται στην εκφώνηση, τροποποιώντας τον, όπου είναι απαραίτητο, για τη γραμματική της άσκησης.

Συγκεκριμένα:

- Τροποποιήθηκε το λεξικό του scanner του plex.
- Άλλαξαν οι μέθοδοι που αντιστοιχούν στα μη τερματικά σύμβολα της γραμματικής με βάση τα νέα FIRST και FOLLOW sets και τους νέους κανόνες της γραμματικής.

Συνοπτικά, ο κώδικας λειτουργεί ως εξής:

1. Η κλάση ***MyParser*** αναπαριστά όλη τη λειτουργικότητα parsing για τη συγκεκριμένη γραμματική.
2. Η μέθοδος ***create_scanner*** δημιουργεί έναν scanner πάνω σε ένα file pointer με τη βοήθεια του plex.
3. Η μέθοδος ***parse*** καλεί την *create_scanner* και έπειτα ξεκινά το parsing στο fp με τη μέθοδο ***stmt_list***, η οποία υλοποιεί τον πρώτο κανόνα της γραμματικής.
4. Στη συνέχεια, καλούνται οι υπόλοιπες μέθοδοι που αντιστοιχούν στα άλλα μη τερματικά σύμβολα της γραμματικής, υψώνοντας ***ParseError*** όπου βρεθεί όρος που δεν αναγνωρίζει η γραμματική.
5. Στην παραπάνω διαδικασία συνεισφέρουν και οι μέθοδοι ***next_token***, η οποία επιστρέφει ένα tuple με το επόμενο token και το κείμενο που ταίριαξε, ***match***, που ταιριάζει ένα token με το τωρινό lookahead και καλεί την *next_token* και ***position***, που βοηθάει στον εντοπισμό λαθών.
6. Στην αρχή του προγράμματος, δημιουργείται instance της κλάσης *MyParser* και καλείται η μέθοδος *parse* στο αρχείο με όνομα 'recursive-descent-parsing.txt'. Σε περίπτωση επιτυχής αναγνώρισης, εμφανίζεται το μήνυμα 'Parsing successful!'.

4 Περιγραφή του runner

Ο runner κάνει μερικές αλλαγές και προσθέτει μερικές μεθόδους για τη δυνατότητα διερμηνείας του πηγαίου κώδικα που αναγνωρίστηκε με τον parser.

Συνοπτικά, ακολουθείται η εξής λογική:

- Κάθε φορά που καλείται η *match*, προστίθεται στη λίστα ***run_values*** η τιμή val του κειμένου, που ταίριαξε με το token.
- Όποτε τελειώνει το κάλεσμα της *stmt* από την *stmt_list* καλείται η μέθοδος ***evaluate_stmt***, η οποία εκτελεί τις εντολές (ανάθεση και print) που μόλις αναγνωρίστηκαν από τη *stmt* και αποθηκεύτηκαν στη *run_values*. Όταν εκτελεστούν οι εντολές, αδειάζει η λίστα *run_values* για την καταγραφή της επόμενης εντολής.
- Η κύρια διαδικασία εκτελείται από τη μέθοδο ***recursive_eval*** που καλείται από την *evaluate_stmt*, η οποία εξηγείται αναλυτικότερα παρακάτω. Επιγραμματικά, ελαττώνει αναδρομικά, εκτελώντας πράξεις, το μήκος της λίστας των όρων προς υπολογισμό. Αυτό επιτυγχάνεται απλοποιώντας συνεχώς σύνθετες εκφράσεις, μέχρι να είναι εύκολα υπολογίσιμες.
- Η μέθοδος ***get_value*** δέχεται ως όρισμα μία μεταβλητή name και επιστρέφει την τιμή που χρησιμοποιείται για τον υπολογισμό των εκφράσεων. Π.χ. αν σε κάποιο σημείο του πηγαίου κώδικα έχει δωθεί η τιμή 0 στη μεταβλητή με όνομα var1, η get_value('var1') θα επιστρέψει False. Αν η μεταβλητή name δεν έχει οριστεί προηγουμένως, θα υψώσει ***RuntimeError***.

Η μέθοδος *recursive_eval* είναι υπεύθυνη για τον υπολογισμό των λογικών εκφράσεων που διαβάζονται απ' τον πηγαίο κώδικα. Η μέθοδος αυτή δέχεται ως όρισμα μία λίστα sequence και στο εξής θα γίνεται αναφορά στο συγκεκριμένο όρισμα απλώς ως λίστα ή sequence. Αρχικά, περνιέται στη μέθοδο ως όρισμα το κομμάτι της *run_values* που απαιτεί υπολογισμό. Ακολουθεί τα εξής βήματα αναδρομικά:

1. Αν το μέγεθος της λίστας είναι 1, επίστρεψε get_value(sequence[0]).
2. Αν υπάρχουν παρενθέσεις στη λίστα, υπολόγισε την τιμή της έκφρασης εντός των παρενθέσεων, αντικατέστησε τις παρενθέσεις και την έκφραση μέσα σε αυτές με την τιμή που υπολογίστηκε και επίστρεψε την τιμή recursive_eval() για τη νέα λίστα.

3. Για κάθε θέση i της λίστας όπου ισχύει $sequence[i] =$'not', αντικατέστησε τις τιμές στις θέσεις i και $i+1$ με την τιμή not get_value($sequence[i+1]$) και επίστρεψε την τιμή της recursive_eval() για τη νέα λίστα.

4. Αν υπάρχει το 'and' στη λίστα στη θέση i, αντικατέστησε το μαζί με τους γειτονικούς του όρους με το αποτέλεσμα της πράξης get_value($sequence[i-1]$) and get_value($sequence[i+1]$), και επίστρεψε το αποτέλεσμα της recursive_eval() για τη νέα λίστα.

5. Επανάλαβε το βήμα 4 για τον όρο 'or' αντί για 'and'.

Για να είναι ξεκάθαρη η έννοια της αντικατάστασης όπως περιγράφεται παραπάνω, δίνεται το εξής παράδειγμα: Αν στο βήμα 4, το όρισμα sequence είναι [True, 'and', False, 'or', 'a'], η νέα λίστα μετά το βήμα της αντικατάστασης θα είναι [False, 'or', 'a'].

Δεν είμαι βέβαιος για την αποδοτικότητα του παραπάνω αλγορίθμου ή το βαθμό στον οποίο εκμεταλλεύεται το μέρος του parsing, μιας και ο μόνος τρόπος με τον οποίο επωφελείται από το parsing είναι η συντακτική ορθότητα των εντολών. Ίσως θα ήταν δυνατό η εκτέλεση να γίνεται ταυτόχρονα με το parsing. Παρ' όλ' αυτά, η έλλειψη χρόνου (και πιθανότατα η απουσία από τα πρόσφατα εργαστήρια) οδήγησαν σε αυτόν τον τρόπο επίλυσης αυτού του ερωτήματος.

Αποτελέσματα εξόδου

```
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> cat recursive-descent-parsing.txt
a = True
print a
b = a or 0
print b
c = not (a or b) and f
print c
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python parser.py
Parsing successful!
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python runner.py
True
True
False
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2>
```

```
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> cat recursive-descent-parsing.txt
a = 0 or 1
print not a or not T and TruE
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python parser.py
Parsing successful!
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python runner.py
False
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2>
```

```
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> cat recursive-descent-parsing.txt
a = not b
b = not (t or) and 1
print b
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python parser.py
Parser Error: in term: ( or IDENTIFIER or BOOL or not expected at line 2 char 14. Found: )
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python runner.py
Run Error: Variable "b" referenced before assignment. at line 2 char 1
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2>
```

```
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> cat recursive-descent-parsing.txt
print T and FaLsE or (True or (0) and 1) )
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python parser.py
Parser Error: in stmt_list: IDENTIFIER or print expected at line 1 char 42. Found: )
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2> python runner.py
True
Parser Error: in stmt_list: IDENTIFIER or print expected at line 1 char 42. Found: )
PS C:\Users\Alex\Desktop\Python\Compilers_Assignment_2>
```