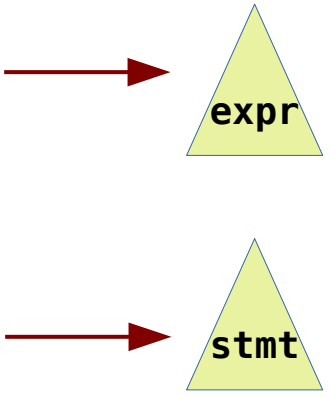
Μεταγλωττιστές 2023-24

Εκτέλεση (διερμηνεία) AST αριθμητικών εκφράσεων

Τι περιέχει το AST (επανάληψη)

- Προκύπτει από τη συντακτική ανάλυση της γραμματικής των αριθμητικών εκφράσεων & εντολών ανάθεσης/εκτύπωσης
  - Μέρος του AST αναπαριστά την αριθμητική έκφραση
  - Το υπόλοιπο AST αναπαριστά την ακολουθία εντολών (ανάθεση/εκτύπωση), με τη σειρά που δίνονται στον πηγαίο κώδικα



AST αριθμητικής έκφρασης (expr)

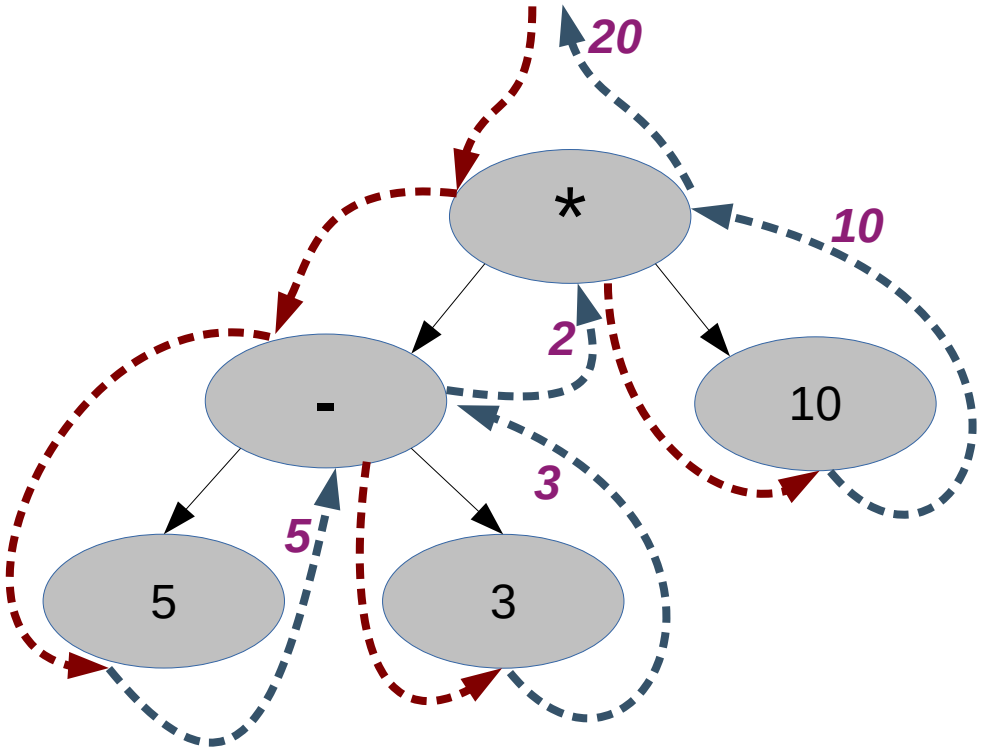
- Τα φύλλα είναι
  - NUMBER: Αριθμητικές σταθερές (τιμές float)
  - DEREF: Αναφορές σε μεταβλητές
    - Κατά τον υπολογισμό, κάθε εμφάνιση μεταβλητής πρέπει να αντικατασταθεί από την τρέχουσα τιμή της
- Οι ενδιάμεσοι κόμβοι είναι
  - OP: Αριθμητικοί δυαδικοί τελεστές που συνδυάζουν τις τιμές του αριστερού και δεξιού υποδένδρου
    - Ανάλογα με το είδος του τελεστή υπολογίζεται μια νέα τιμή

Τιμές μεταβλητών

- Στη γλώσσα που δουλεύουμε οι μεταβλητές δεν δηλώνονται
  - Υπάρχει ένας μοναδικός τύπος δεδομένων: float
  - Που αποθηκεύονται οι τιμές των μεταβλητών;
    - Κατά τη διάρκεια της διερμηνείας χρησιμοποιούμε ένα λεξικό (dict) ως πίνακα συμβόλων (symbol table) με κλειδιά τα ονόματα των μεταβλητών και τιμές τις τιμές των μεταβλητών
  - Αρχικά το symbol table είναι κενό
    - Γεμίζει με τις εντολές ανάθεσης (ASSIGN) τιμής σε μεταβλητή
  - Όταν συναντήσουμε κόμβο DEREF, αναζητούμε το όνομα της μεταβλητής στο symbol table και επιστρέφουμε την τιμή που βρίσκεται μέσα
    - Σφάλμα Εκτέλεσης (Run Error) αν η μεταβλητή δεν βρεθεί στο symbol table (δεν έχει οριστεί ακόμα)

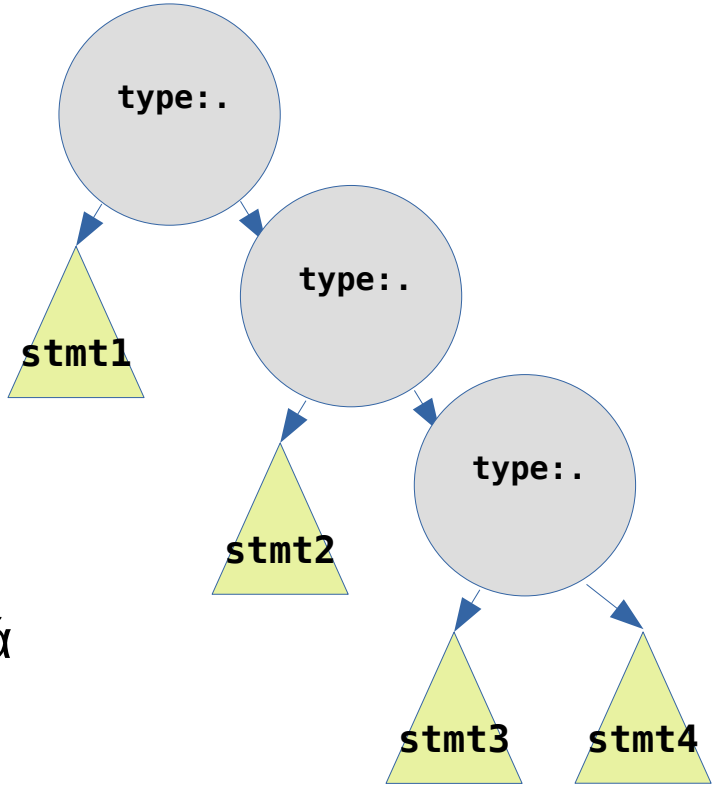
Υπολογισμός αριθμητικής έκφρασης

- Διάσχιση δένδρου DFS
  - Προτεραιότητα κατά βάθος
- Επεξεργασία post-order
  - Αφού επιστρέψουμε από τα παιδιά
  - Υπολογισμός πράξης και προώθηση προς τα πίσω



Υπόλοιπο AST (ακολουθία εντολών)

- Stmt = εντολή
  - ASSIGN ή PRINT
- Και εδώ διασχίζουμε το δένδρο με DFS
  - Αν ο κόμβος είναι εντολή την εκτελούμε
- Αν είναι . (concat) επισκεπτόμαστε τα παιδιά του από αριστερά προς δεξιά

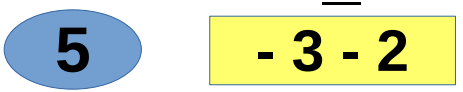


Το πρόβλημα της προσεταιριστικότητας (associativity) των τελεστών

- Η κλασσική υλοποίηση αναδρομικής κατάβασης δεν χειρίζεται σωστά εκφράσεις όπως 5 - 3 - 2 ή 20 / 4 / 2
  - Οι τελεστές - και / είναι αριστερά προσεταιριστικοί (left associative)
  - Οι πιο πάνω εκφράσεις πρέπει να υπολογιστούν ως (5 - 3) - 2 και (20 / 4) / 2
- Αντιθέτως, ο συντακτικός αναλυτής τις χειρίζεται ως 5 - (3 – 2) και 20 / (4 / 2)
  - λόγω της μεθόδου ανάλυσης (recursive descent LL(1))

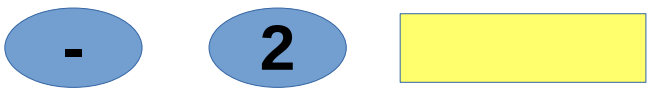
5 - 3 - 2

Expr → Term Term_tail



Term_tail → Addop Term Term_tail | ε

Term_tail → Addop Term Term_tail | ε



Term_tail → Addop Term Term_tail | ε

5 - 3 - 2

Expr → Term Term_tail

5 | -3-2

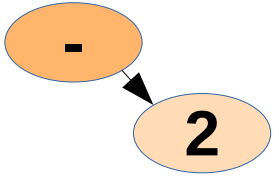
Term_tail → Addop Term Term_tail | ε

\- | 3 | -2

Term_tail → Addop Term Term_tail | ε

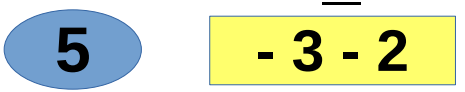
\- | 2 |

Term_tail → Addop Term Term_tail | ε



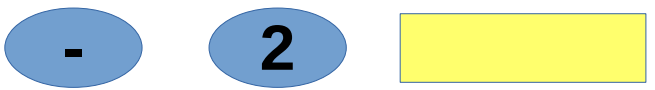
5 - 3 - 2

```
Expr        → Term Term_tail
```

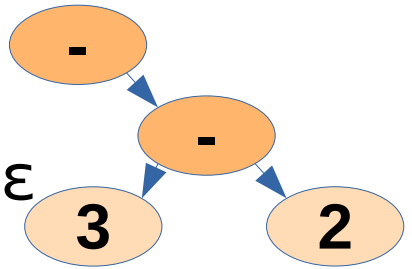


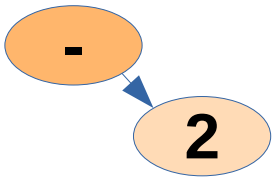

```
Term_tail   → Addop Term Term_tail | ε
```


```
Term_tail   → Addop Term Term_tail | ε
```

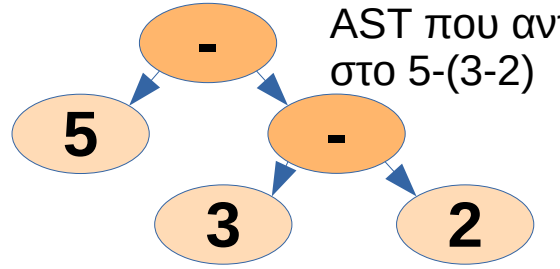



```
Term_tail   → Addop Term Term_tail | ε
```



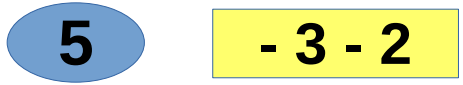


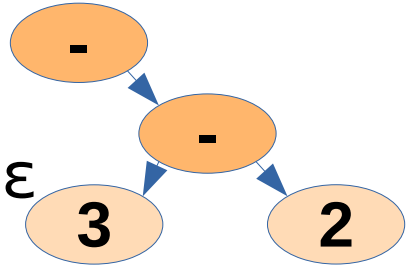
5 - 3 - 2



AST που αντιστοιχεί στο 5-(3-2)

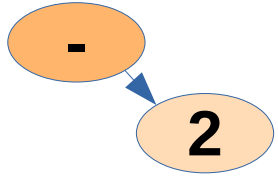
```
Expr        → Term Term_tail
```






```
Term_tail   → Addop Term Term_tail | ε
```


```
Term_tail   → Addop Term Term_tail | ε
```




```
Term_tail   → Addop Term Term_tail | ε
```

Τροποποίηση της μεθόδου ανάλυσης

```
Stmt_list   → Stmt Stmt_list | ε
Stmt        → id = Expr | print Expr
Expr        → Term (Addop Term)*
~~Term_tail → Addop Term Term_tail | ε~~
Term        → Factor (Multop Factor)*
~~Factor_tail → Multop Factor Factor_tail | ε~~
Factor      → (Expr) | id | number
Addop       → + | -
Multop      → * | /
```

- Όσο (while) υπάρχουν τελεστές ίδιου επιπέδου προτεραιότητας, συνεχίζουμε την κατανάλωση εισόδου → επεξεργασία από αριστερά προς τα δεξιά
  - Επεκταμένη σύνταξη γραμματικής, (...)* = «μηδέν ή περισσότερες φορές»

Ο νέος κώδικας για το Expr()

```
def Expr(self):
    if self.next_token in ('(','id','number'):
        # Expr → Term (Addop Term)*
        self.Term()
        while self.next_token in ('+','-'):
            self.Addop()
            self.Term()

    else:
        raise ParseError
```

Αντίστοιχα για το Term(), με *,/

- Ο τροποποιημένος κώδικας μπορεί να εξυπηρετήσει τελεστές με αριστερή μόνο (-, /) ή με αριστερή/δεξιά προσεταιριστικότητα (+,*)
- Αν υπάρχουν τελεστές με δεξιά μόνο προσεταιριστικότητα (π.χ. η ύψωση σε δύναμη) θα πρέπει να χρησιμοποιηθεί ο αρχικός κώδικας