

# Μεταγλωττιστές 2020-21

Συντακτική ανάλυση top-down LL(1)

# Συντακτική ανάλυση top-down LL(1)

- Για μια σημαντική υποκατηγορία context-free γραμματικών με πολυπλοκότητα  $O(n)$ 
  - Σάρωση εισόδου από αριστερά προς τα δεξιά (left-to-right)
  - Αριστερότερες ακολουθίες παραγωγής (leftmost derivations)
  - Συμβουλευόμενοι ένα (το επόμενο) token εισόδου σε κάθε βήμα

# Προϋποθέσεις για αριστερότερες παραγωγές

- Δεν πρέπει οι κανόνες να έχουν **αριστερή αναδρομή**

$\text{Expr} \rightarrow \text{Expr} + \text{Term}$

- Ο συντακτικός αναλυτής θα πέσει σε άπειρη αναδρομή
- Απαλείφεται με αλγοριθμικό τρόπο

$A \rightarrow A \alpha \mid \beta$

( $\alpha, \beta$  ακολουθίες οποιωνδήποτε συμβόλων) γίνεται

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

- **Προσοχή**: πρέπει να γίνει και σε περιπτώσεις έμμεσης αναδρομής (αλυσιδωτά, μετά από περισσότερα από ένα βήματα)

# Προϋποθέσεις για αριστερότερες παραγωγές

- Δεν πρέπει να υπάρχει κοινός παράγοντας στην αρχή π.χ.

$\text{Factor} \rightarrow \text{id} \mid \text{id} \mid \text{ExprList} \mid \text{id} ( \text{ExprList} )$

- Μετασχηματισμός με προσθήκη ενδιάμεσου κανόνα

$\text{Factor} \rightarrow \text{id} \text{ Args}$

$\text{Arg} \rightarrow \varepsilon \mid \text{ExprList} \mid ( \text{ExprList} )$

# Υλοποίηση συντακτικού αναλυτή LL(1)

- Από τη Θεωρία Υπολογισμού έχουμε ένα θεωρητικό εργαλείο
  - **Pushdown Automaton (PDA)**
  - Αυτόματο ενισχυμένο με μια στοίβα (stack)
  - Για κάθε μετάβαση το αυτόματο συμβουλεύεται **το σύμβολο εισόδου** και **την κορυφή της στοίβας**
    - για να αποφασίσει ποια θα είναι η επόμενη κατάσταση
    - και αν/τι θα μπει στη στοίβα
  - Υπάρχει πάντα ένα αυτόματο PDA ισοδύναμο με μια γραμματική χωρίς συμφραζόμενα (CFG)
  - Για τις γραμματικές LL(1) το αυτόματο PDA είναι **αιτιοκρατικό** (ντετερμινιστικό)
    - είναι δυνατή το πολύ μια μετάβαση

# Πώς χρησιμοποιείται το αυτόματο PDA στη συντακτική ανάλυση

- Αρχικά στη στοίβα μπαίνει το **αρχικό** μη τερματικό σύμβολο της γραμματικής
- Το αυτόματο επαναλαμβάνει
  - Αν στην κορυφή της στοίβας βρίσκεται **μη τερματικό σύμβολο A** (λειτουργία **predict**), τότε το A αντικαθίσταται στη στοίβα από το δεξιό μέρος ενός κατάλληλου κανόνα της γραμματικής, σύμφωνα και με το επόμενο σύμβολο στην είσοδο
  - Αν στην κορυφή της στοίβας βρίσκεται **τερματικό σύμβολο a** (λειτουργία **match**), τότε, αν το επόμενο σύμβολο εισόδου είναι ίδιο με το a, το a αφαιρείται από τη στοίβα
  - Σε κάθε άλλη περίπτωση παράγεται σφάλμα

# Απλή (αλλά μη ρεαλιστική) περίπτωση LL(1)

- Έστω η γραμματική

$S \rightarrow a B$

$B \rightarrow b \mid a B b$

- Όλα τα δεξιά μέρη των κανόνων ξεκινούν με τερματικό σύμβολο, διαφορετικό για τις εναλλακτικές παραγωγές του ίδιου μη τερματικού συμβόλου

# Πίνακας συντακτικής ανάλυσης

- Θεωρητικό εργαλείο, μέρος του αυτομάτου PDA
  - αλλά μπορεί να χρησιμοποιηθεί και σε έναν πραγματικό συντακτικό αναλυτή
  - επιλέγει τι θα μπει στη στοίβα στο **predict**

		a	b
S	a B		
B	a B b	b	

τι υπάρχει στην κορυφή της στοίβας;

τι υπάρχει στην είσοδο;

τι θα μπει στη στοίβα;

Π.χ. αν στην κορυφή της στοίβας είναι το **S** και το επόμενο σύμβολο στην είσοδο είναι το **a**, το S θα αντικατασταθεί στη στοίβα από το **a B**



	a	b
S	a B	
B	a B b	b

Είσοδος:

a a b b #

end-of-text (EOT)

κορυφή  
στοίβας

S  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

**“Predict”**

Είσοδος:

a a b b #

S  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

**“Predict”**

Είσοδος:

a a b b #

a  
B  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

a a b b #

a  
B  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

a b b #

B  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

**“Predict”**

Είσοδος:

a b b #

B  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

**“Predict”**

Είσοδος:

a b b #

a  
B  
b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

**“Match”**

Είσοδος:

a b b #

a  
B  
b  
#

Στοίβα



	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

b b #

B  
b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

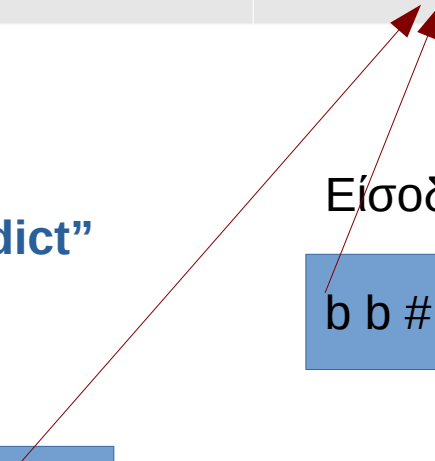
**“Predict”**

Είσοδος:

b b #

B  
b  
#

Στοίβα



	a	b
S	a B	
B	a B b	b

**“Predict”**

Είσοδος:

b b #

b  
b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

b b #

b  
b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

b #

b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

Είσοδος:

b #

b  
#

Στοίβα

	a	b
S	a B	
B	a B b	b

“Match”

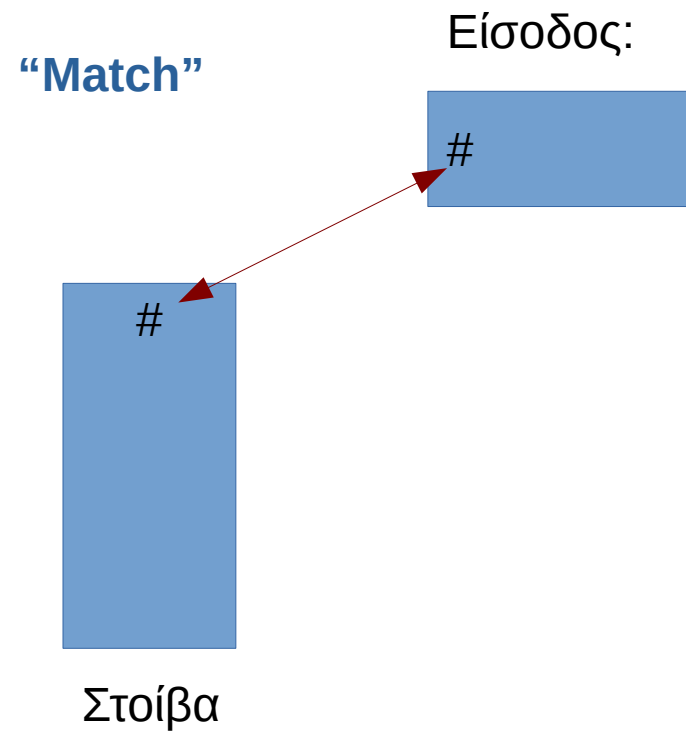
Είσοδος:

#

#

Στοίβα

	a	b
S	a B	
B	a B b	b





	a	b
S	a B	
B	a B b	b

Τερματισμός

Είσοδος:



Στοίβα

# Κατασκευή συντακτικού αναλυτή LL(1)

- Θα μπορούσαμε να φτιάξουμε ένα πρόγραμμα που να μιμείται τη λειτουργία ενός PDA
- Είναι όμως πιο βολικό να υλοποιήσουμε με συστηματικό τρόπο τον συντακτικό αναλυτή ως σύνολο **συναρτήσεων** που καλούν η μία την άλλη (ενδεχομένως και αναδρομικά)
  - «Stack»: υλοποιείται από τη **στοίβα κλήσης** των συναρτήσεων
  - Μπορούμε να κάνουμε τη σημασιολογική ανάλυση μέσα στις συναρτήσεις αυτές
- Συντακτικός αναλυτής **αναδρομικής κατάβασης** (recursive descent parser)

# Μέθοδος αναδρομικής κατάβασης

- Για κάθε **μη τερματικό σύμβολο** φτιάχνουμε **μία συνάρτηση**
  - Καλείται πρώτη η συνάρτηση του αρχικού μη τερματικού συμβόλου της γραμματικής
- Κάθε συνάρτηση υλοποιεί τους **κανόνες** που έχουν στο αριστερό μέρος το αντίστοιχο μη τερματικό σύμβολο
  - Ένας κλάδος if για κάθε κανόνα
    - **Πώς διαλέγω**; στην απλή γραμματική του παραδείγματος, αρκεί να ελέγχω το επόμενο σύμβολο εισόδου – δεν αρκεί φυσικά για πιο σύνθετες γραμματικές!
- Υλοποίηση κανόνων
  - Υλοποιώ το **δεξιό μέρος κάθε κανόνα**
    - Όπου υπάρχει **τερματικό**, καλώ συνάρτηση **match()**
      - Η match(), αν η είσοδος ταιριάζει με το αναμενόμενο τερματικό, προχωρά στο επόμενο σύμβολο εισόδου
    - Όπου υπάρχει **μη τερματικό**, καλώ την **αντίστοιχη συνάρτηση**

# Παράδειγμα

```
def B():  
  
    if next_token=='B_TOKEN':  
        # B -> b  
        match('B_TOKEN')  
  
    elif next_token=='A_TOKEN':  
        # B -> a B b  
        match('A_TOKEN')  
        B()  
        match('B_TOKEN')  
  
    else:  
        raise ParseError("...error msg...")
```

# Γραμματική LL(1) αριθμητικών εκφράσεων

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

- Πιο σύνθετη από το προηγούμενο παράδειγμα
  - Υπάρχουν κανόνες που το δεξιό μέρος ξεκινά με μη τερματικό
  - Υπάρχουν κενές παραγωγές (με ε στο δεξιό μέρος των κανόνων)
- Ποιους κανόνες μπορείτε να υλοποιήσετε με όσα ξέρετε μέχρι τώρα;