

Μεταγλωττιστές 2020-21

Συντακτική ανάλυση LL(1):
Σύνολα FIRST και FOLLOW

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)

Διατύπωση προβλήματος

- Αν το δεξιό μέρος ενός κανόνα **ξεκινά με τερματικό**, ξέρω πώς θα τον επιλέξω για υλοποίηση

```
def B():  
    if next_token=='B_TOKEN':  
        # B -> b  
        match('B_TOKEN')
```

- Αν **ξεκινά με μη τερματικό**, πώς γίνεται η επιλογή;

```
def Expr():  
    if next_token==?????:  
        # Expr -> Term Term_tail  
        Term()  
        Term_tail()
```

- Κι αν ξέραμε όλα τα πιθανά τερματικά με τα οποία ξεκινούν οι προτάσεις που παράγει το Term;

Σύνολα FIRST

- **FIRST(x)** είναι το σύνολο των τερματικών από τα οποία ξεκινά κάθε πρόταση που παράγεται από το x σε 0 ή περισσότερα βήματα
 - Όπου x οποιαδήποτε ακολουθία τερματικών και μη τερματικών συμβόλων
- Αν x ξεκινά με **τερματικό σύμβολο a** , το $FIRST(x)$ περιέχει το ίδιο το a
- Αν x ξεκινά με **μη τερματικό σύμβολο A** , το $FIRST(x)$ ισούται με το $FIRST(A)$
 - σημ: εάν το A παράγει ϵ , το $FIRST(x)$ μπορεί να περιλαμβάνει και άλλα στοιχεία
 - θα το δούμε αργότερα

Εύρεση συνόλων FIRST

(όταν δεν υπάρχουν κενές παραγωγές)

- Ξεκινάμε με **άδεια** σύνολα FIRST για κάθε μη τερματικό σύμβολο της γραμματικής
- Για κάθε κανόνα της γραμματικής:
 - Αν το δεξιό μέρος του κανόνα ξεκινά με **τερματικό a** , προσθέτουμε το a στο σύνολο FIRST του μη τερματικού που βρίσκεται στο αριστερό μέρος του κανόνα
 - Αν το δεξιό μέρος του κανόνα ξεκινά με **μη τερματικό A** , προσθέτουμε τα σύμβολα του FIRST(A) που ξέρουμε εκείνη τη στιγμή στο σύνολο FIRST του μη τερματικού που βρίσκεται στο αριστερό μέρος του κανόνα
- Επαναλαμβάνουμε τη διαδικασία από την αρχή, έως ότου να μην μπορεί να προστεθεί άλλο σύμβολο σε κάποιο σύνολο FIRST
 - Ο αλγόριθμος τερματίζει εγγυημένα: ο αριθμός των επαναλήψεων που θα γίνουν φράσσεται από τον αριθμό των τερματικών και μη τερματικών συμβόλων, που είναι πεπερασμένος.

Παράδειγμα

Σύνολα FIRST	Κανόνες
	<code>Session -> Fact Session Question (Session) Session</code>
	<code>Fact -> ! string</code>
	<code>Question -> ? string</code>

Παράδειγμα από το: Dick Grune. 2010. *Parsing Techniques: A Practical Guide* (2nd. ed.). Springer

Παράδειγμα

Σύνολα FIRST	Κανόνες
(Session -> Fact Session Question (Session) Session
!	Fact -> ! string
?	Question -> ? string

Παράδειγμα

Σύνολα FIRST	Κανόνες
!	Session -> Fact Session
?	Question
((Session) Session
!	Fact -> ! string
?	Question -> ? string

- Μπορείτε να υπολογίσετε με τον ίδιο τρόπο τα σύνολα FIRST για τη γραμματική των αριθμητικών εκφράσεων;
 - αγνοήστε προς το παρόν του κανόνες με ε

Σύνολα FIRST	Κανόνες
	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 → * /

Σύνολα FIRST	Κανόνες	
	Stmt_list	→ Stmt Stmt_list ε
id print	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
(id number	Factor	→ (Expr) id number
+ -	Addop	→ + -
* /	Multop	→ * /

Σύνολα FIRST	Κανόνες	
id, print	Stmt_list	→ Stmt Stmt_list ε
id print	Stmt	→ id = Expr print Expr
	Expr	→ Term Term_tail
+, -	Term_tail	→ Addop Term Term_tail ε
(, id, number	Term	→ Factor Factor_tail
*, /	Factor_tail	→ Multop Factor Factor_tail ε
(id number	Factor	→ (Expr) id number
+ -	Addop	→ + -
* /	Multop	→ * /

Σύνολα FIRST	Κανόνες	
id, print	Stmt_list	→ Stmt Stmt_list ε
id print	Stmt	→ id = Expr print Expr
(, id, number	Expr	→ Term Term_tail
+, -	Term_tail	→ Addop Term Term_tail ε
(, id, number	Term	→ Factor Factor_tail
*, /	Factor_tail	→ Multop Factor Factor_tail ε
(id number	Factor	→ (Expr) id number
+ -	Addop	→ + -
* /	Multop	→ * /

Χρήση συνόλων FIRST (γραμματικές χωρίς ε)

- Επιλέγουμε την υλοποίηση ενός κανόνα $A \rightarrow x$ όταν εμφανιστεί token που ανήκει στο **FIRST(x)**

```
def Expr():  
    if next_token=='(' or next_token=='id' or next_token=='number':  
        # Expr -> Term Term_tail  
        # FIRST(Term Term_tail) = FIRST(Term) = { (, id, number }  
        Term()  
        Term_tail()
```

- Ή εναλλακτικά

```
def Expr():  
    if next_token in ('(', 'id', 'number'):  
        # Expr -> Term Term_tail  
        # FIRST(Term Term_tail) = FIRST(Term) = { (, id, number }  
        Term()  
        Term_tail()
```

- Προσοχή: για κάθε εναλλακτικό κανόνα του ίδιου μη τερματικού συμβόλου θα πρέπει τα σύνολα FIRST των δεξιών μερών να μην έχουν κοινά στοιχεία αλλιώς η γραμματική δεν είναι LL(1)!

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     → * | /
```

- Χρησιμοποιώντας τα σύνολα FIRST υλοποιήστε τους αντίστοιχους κανόνες
 - Παραμένει η εκκρεμότητα με τις κενές παραγωγές (με ε στο δεξιό μέρος των κανόνων)

Το πρόβλημα με τις κενές παραγωγές

- Αν υπάρχει κανόνας $A \rightarrow \epsilon$, με ποιο κριτήριο επιλέγω αυτόν τον κανόνα;
 - Χωρίς να καταναλώσω σύμβολο εισόδου;
- Ο υπολογισμός των συνόλων FIRST τροποποιείται
 - Αν υπάρχει κανόνας $A \rightarrow Bx$ και το B παράγει ϵ , τότε το $FIRST(A)$ δεν περιλαμβάνει μόνο το $FIRST(B)$, εφόσον το B μπορεί να “εξαφανιστεί”
 - Πρέπει να λάβουμε υπόψη και το $FIRST(x)$

Εύρεση συνόλων FIRST

(όταν υπάρχουν κενές παραγωγές)

- Ξεκινάμε με **άδεια** σύνολα FIRST για κάθε μη τερματικό σύμβολο της γραμματικής
- Για κάθε κανόνα της γραμματικής:
 - Αν το δεξιό μέρος του κανόνα ξεκινά με **τερματικό a ή το ϵ** , προσθέτουμε το a **ή το ϵ** στο σύνολο FIRST του μη τερματικού που βρίσκεται στο αριστερό μέρος του κανόνα
 - Αν το δεξιό μέρος του κανόνα ξεκινά με **μη τερματικό A** , προσθέτουμε τα σύμβολα του FIRST(A) που ξέρουμε εκείνη τη στιγμή στο σύνολο FIRST του μη τερματικού που βρίσκεται στο αριστερό μέρος του κανόνα
 - Αν το FIRST(A) περιέχει το ϵ , προσθέτουμε αντί για αυτό, το σύνολο FIRST του συμβόλου μετά το A
 - Επαναλαμβάνουμε αν και το επόμενο σύμβολο παράγει ϵ . Αν όλα τα σύμβολα στο δεξιό μέρος του κανόνα παράγουν ϵ , προσθέτουμε το ϵ .
- Επαναλαμβάνουμε τη διαδικασία από την αρχή, έως ότου να μην μπορεί να προστεθεί άλλο σύμβολο σε κάποιο σύνολο FIRST

Σύνολα FIRST	Κανόνες	
id, print ε	Stmt_list	→ Stmt Stmt_list ε
id print	Stmt	→ id = Expr print Expr
(, id, number	Expr	→ Term Term_tail
+, - ε	Term_tail	→ Addop Term Term_tail ε
(, id, number	Term	→ Factor Factor_tail
*, / ε	Factor_tail	→ Multop Factor Factor_tail ε
(id number	Factor	→ (Expr) id number
+ -	Addop	→ + -
* /	Multop	→ * /

Σύνολα FOLLOW

- Για να αποφασίσω πότε ακολουθώ μια κενή παραγωγή $A \rightarrow \epsilon$
- **FOLLOW(A)** είναι το σύνολο **όλων** των τερματικών συμβόλων που **πιθανόν** ακολουθούν το A, σε **οποιαδήποτε** παραγωγή
 - Δεν μπορούμε να ξέρουμε ακριβώς τι ακολουθεί το A ανά πάσα στιγμή
 - Εξαρτάται από τους προηγούμενους κανόνες που έχουν χρησιμοποιηθεί ως τώρα
 - Το γνωρίζουμε μόνο κατά την εκτέλεση της συντακτικής ανάλυσης
 - Το σύνολο FOLLOW μπορεί να υπολογιστεί εκ των προτέρων και δίνει μια προσεγγιστική λύση, χωρίς να επηρεάζεται η ορθή λειτουργία του συντακτικού αναλυτή
 - Αν υπάρχει συντακτικό σφάλμα, ίσως εκτελεστούν μερικά βήματα παραπάνω πριν αυτό εντοπιστεί

Εύρεση συνόλων FOLLOW

- Τα σύνολα FOLLOW αρχικά είναι κενά.
- Για κάθε δεξί μέρος των κανόνων:
 - Για κάθε μη τερματικό B και τυχαίες ακολουθίες συμβόλων α, β σε κανόνες της μορφής $A \rightarrow \alpha B \beta$, προσθέτουμε το $FIRST(\beta)$ -εκτός από το ϵ - στο $FOLLOW(B)$.
 - το α μπορεί να μην υπάρχει
 - Για κάθε μη τερματικό B και τυχαία ακολουθία συμβόλων α σε κανόνες της μορφής $A \rightarrow \alpha B$, προσθέτουμε το $FOLLOW(A)$ στο $FOLLOW(B)$.
 - Για κάθε μη τερματικό B και τυχαίες ακολουθίες συμβόλων α, β σε κανόνες της μορφής $A \rightarrow \alpha B \beta$, όπου το ϵ ανήκει στο $FIRST(\beta)$, προσθέτουμε το $FOLLOW(A)$ στο $FOLLOW(B)$.
- Επαναλαμβάνουμε ξανά τη διαδικασία για όλους τους κανόνες, έως ότου να μην υπάρχουν νέες προσθήκες στα σύνολα FOLLOW.

Παράδειγμα

Σύνολα FOLLOW	Σύνολα FIRST	Κανόνες
		<code>-> Session #</code>
		<code>Session -> Facts Question (Session) Session</code>
		<code>Facts -> Fact Facts ε</code>
		<code>Fact -> ! string</code>
		<code>Question -> ? string</code>

Παράδειγμα από το: Dick Grune. 2010. *Parsing Techniques: A Practical Guide* (2nd. ed.). Springer

Παράδειγμα

Σύνολα FOLLOW	Σύνολα FIRST	Κανόνες
		<code>-> Session #</code>
	<code>? !</code> <code>(</code>	<code>Session -> Facts Question</code> <code> (Session) Session</code>
	<code>!</code> <code>ε</code>	<code>Facts -> Fact Facts</code> <code> ε</code>
	<code>!</code>	<code>Fact -> ! string</code>
	<code>?</code>	<code>Question -> ? string</code>

Παράδειγμα

Σύνολα FOLLOW	Σύνολα FIRST	Κανόνες
		\rightarrow Session #
#)	? ! (Session \rightarrow Facts Question (Session) Session
?	! ϵ	Facts \rightarrow Fact Facts ϵ
! ?	!	Fact \rightarrow ! string
#)	?	Question \rightarrow ? string

- Μπορείτε να υπολογίσετε με τον ίδιο τρόπο τα σύνολα FOLLOW για τη γραμματική των αριθμητικών εκφράσεων;

Χρήση συνόλων FOLLOW

- Σε κάθε συνάρτηση μη τερματικού συμβόλου με ϵ στο σύνολο FIRST του
- Προσθέτουμε έναν κλάδο if που απλώς επιστρέφει χωρίς να καταναλώνει είσοδο (χωρίς κλήση match())
- Όταν εμφανιστεί κάποιο από τα σύμβολα του συνόλου FOLLOW του μη τερματικού

```
def Term_tail():  
    # FOLLOW(Term_tail) = { id, print, ), # }  
    ...  
  
    elif next_token in ('id', 'print', ')', None):  
        return  
    ...
```