Γλώσσες Προγραμματισμού Μεταγλωττιστές

Συντακτική Ανάλυση

Πανεπιστήμιο Μακεδονίας
Τμήμα Εφαρμοσμένης Πληροφορικής

Ηλίας Σακελλαρίου

Δομή

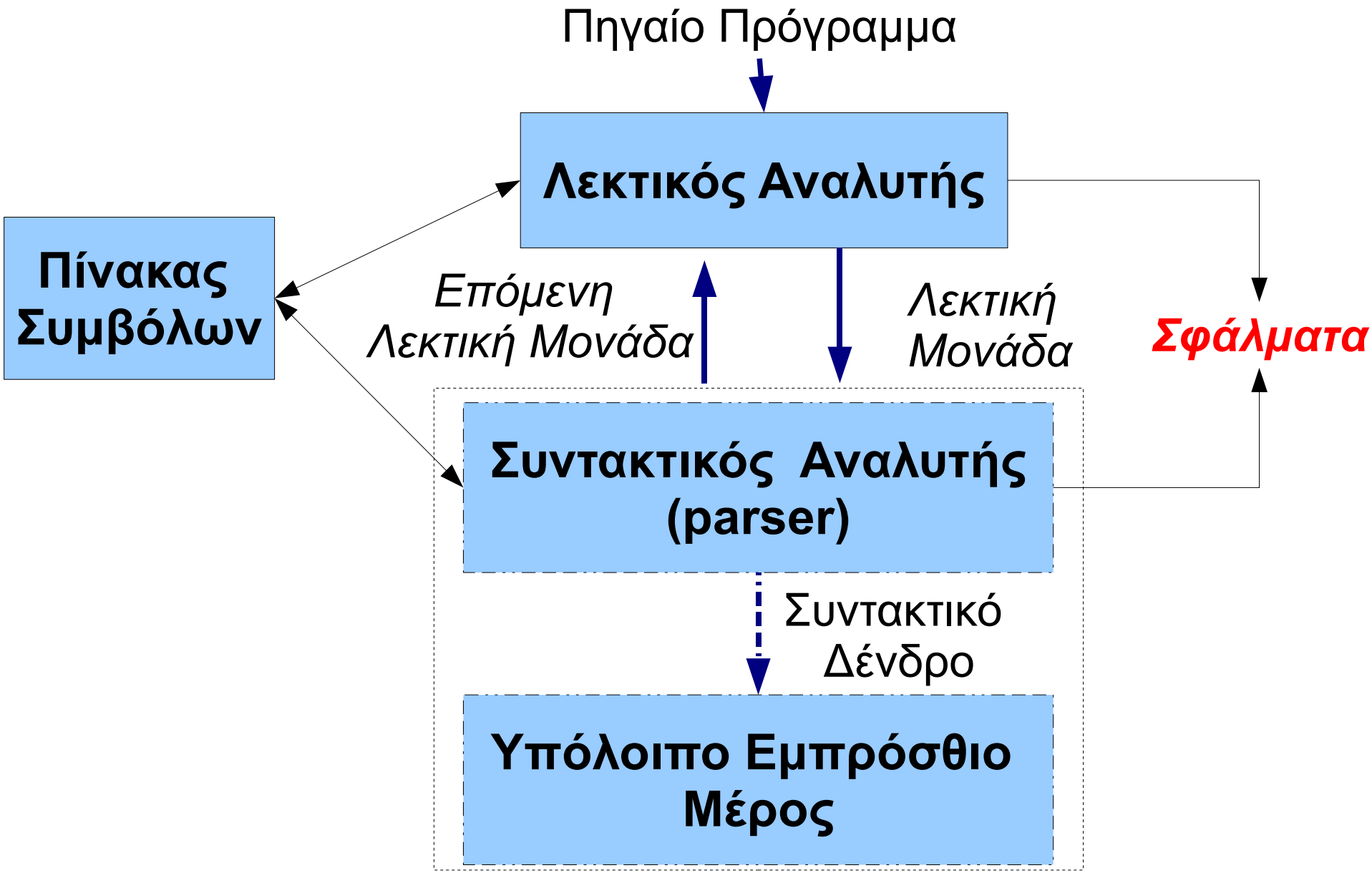
- Συντακτική Ανάλυση
- Γραμματικές χωρίς συμφραζόμενα.
  - Αυτόματα Στοίβας
- Οι συναρτήσεις FIRST και FOLLOW.
- Συντακτικοί Αναλυτές από πάνω προς τα κάτω (top-down)

Φάσεις Μεταγλώττισης

Συντακτική Ανάλυση (Syntax Analysis or Parsing)

- Έλεγχος αν το πρόγραμμα έχει την **ορθή σύνταξη** σύμφωνα με τις προδιαγραφές της γλώσσας.
  - Ορισμός σύνταξης: Γραμματική!
- Είσοδος: ακολουθία λεκτικών μονάδων από λεκτικό αναλυτή.
- Κατασκευή **συντακτικού δένδρου**
  - Άμεση, όταν τα στάδια μεταγλώττισης είναι διακριτά
  - Έμμεση, στην περίπτωση που δεν αποθηκεύεται σε κάποια μορφή το συντακτικό δένδρο.

Αλληλεπιδράσεις ΛΑ και ΣΑ



Γραμματικές χωρίς συμφραζόμενα

- **Γραμματικές τύπου 2 (γραμματικές χωρίς συμφραζόμενα – context free grammars)**
  - Οι κανόνες παραγωγής έχουν μορφή $\alpha \rightarrow \beta$, όπου η συμβολοσειρά α αποτελείται από **ένα μη-τερματικό σύμβολο** και η β είναι συμβολοσειρά.
- Άρα υπάρχουν κανόνες της μορφής:
  - $A \rightarrow \beta$ όπου $A \in N$. πχ.
  - smt $\rightarrow$ if expr then smt
  - smt $\rightarrow$ while (expr) smts

Πρόβλημα Συντακτικής Ανάλυσης

- Πρόβλημα συντακτικής ανάλυσης έγκειται στο να βρεθεί αν μια συμβολοσειρά ανήκει στην γλώσσα που παράγεται από μια γραμματική χωρίς συμφραζόμενα.
- Αλγόριθμοι συντακτικής ανάλυσης:
  - **Καθολικοί (universal)**, οι οποίοι αναγνωρίζουν οποιαδήποτε γραμματική.
  - Από **πάνω προς τα κάτω (top-down parers)**, όπου γίνεται κατασκευή του δένδρου από την ρίζα προς τα φύλλα.
  - Από **κάτω προς τα πάνω (bottom-up parsers)**, όπου η κατασκευή του δένδρου γίνεται από τα φύλλα προς την ρίζα.

Συντακτικοί Αναλυτές (parsers)

- Οι ΣΑ που χρησιμοποιούνται στην πράξη αφορούν ένα **υποσύνολο** των γραμματικών χωρίς συμφραζόμενα.
  - Επιβολή περιορισμών λόγω αποδοτικότητας.
- Ακόμα και με περιορισμούς εν λόγω γραμματικές καλύπτουν τις ανάγκες των περισσοτέρων γλωσσών προγραμματισμού.
- Οι από **πάνω προς τα κάτω ΣΑ (top-down parers),** ευκολότεροι στην κατασκευή και προγραμματίζονται χειρωνακτικά.
- Οι από **κάτω προς τα πάνω (bottom-up parsers),** καλύπτουν μια μεγαλύτερη κλάση γλωσσών και κατασκευάζονται συνήθως από αυτοματοποιημένα εργαλεία.

Γλώσσες Προγραμματισμού
Μεταγλωττιστές

Στοιχεία Γραμματικών
Χωρίς Συμφραζόμενα
(context free grammars)

Παραγωγές στις Γραμματικές χωρίς Συμφραζόμενα (ΓΧΣ)

- **Παραγωγή**: Αντικατάσταση υποσυμβολοσειράς που ταιριάζει με αριστερό μέλος, με το αντίστοιχο δεξιό μέλος του κανόνα παραγωγής.
- Στις ΓΧΣ κάθε φορά αντικαθίσταται οποιοδήποτε μη-τερματικό σύμβολο που περιέχεται στην συμβολοσειρά.
  - **Αριστερότερη παραγωγή**: αντικατάσταση πάντα του αριστερότερου μη-τερματικού συμβόλου
  - **Δεξιότερη παραγωγή**: αντικατάσταση του δεξιότερου μη-τερματικού συμβόλου.

Παράδειγμα

- Γραμματική

T ={+,-,id}
N = {E,T}
S = {E}
P = {
E→E+T
E→E-T
E→T
T→id}

Αριστερότερη παραγωγή:

E=>E+T=>T+T=>id+T=>id+id

Δεξιότερη παραγωγή:

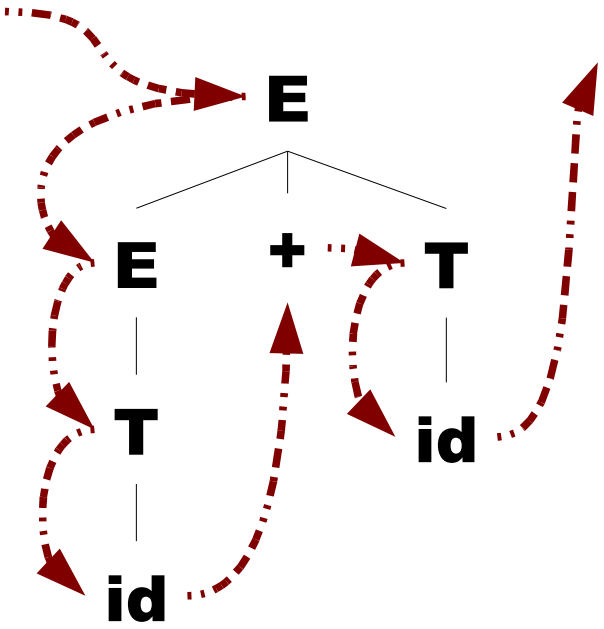
E=>E+T=>E+id=>T+id=>id+id

Συντακτικό Δένδρο (syntax tree)

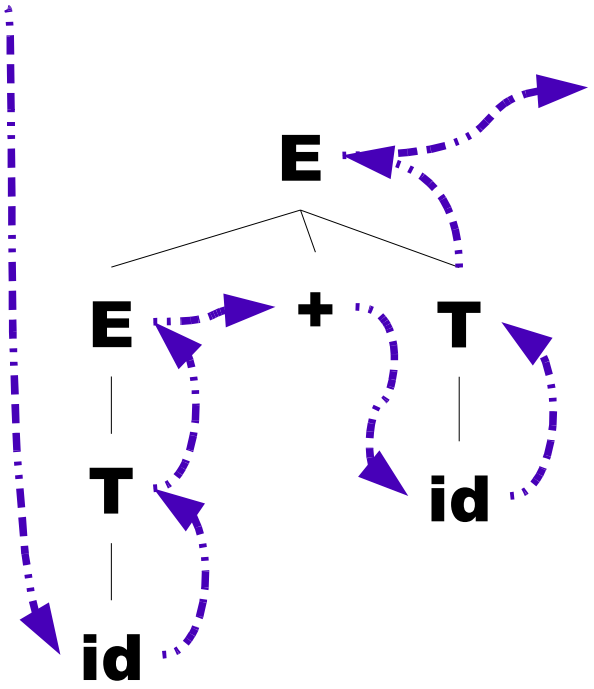
- Τρόπος αναπαράστασης μιας παραγωγής.
- Το αρχικό μη-τερματικό σύμβολο τοποθετείται στη ρίζα του δένδρου.
- Κάθε "ενδιάμεσος" κόμβος του δένδρου αναπαριστά ένα μη-τερματικό σύμβολο.
- Κάθε φύλλο του δένδρου αναπαριστά ένα τερματικό σύμβολο.
- Οι ακμές από ένα "ενδιάμεσο" κόμβο στους απογόνους του αναπαριστούν την αντικατάσταση του μη-τερματικού συμβόλου βάση ενός κανόνα παραγωγής της γραμματικής.

Απλό Παράδειγμα Δένδρου Παραγωγής

T ={+,-,id}
N = {E,T}
S = {E}
P = {
E→E+T
E→E-T
E→T
T→id}



Από πάνω προς τα κάτω

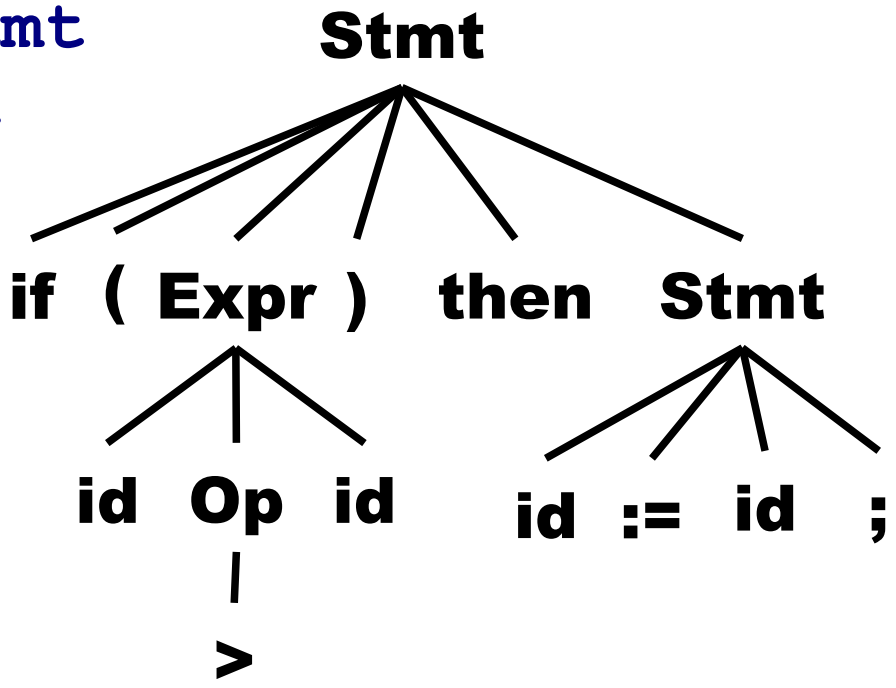


Από κάτω προς τα πάνω

E=>E+T=>T+T=>id+T=>id+id

Παράδειγμα (II.1)

```
P={
Stmt→ if ( Expr ) then Stmt
Stmt→ while ( Expr ) Stmt
Stmt→ id := id ;
Stmt→ id := num ;
Stmt→ begin StmtList end
StmtList→ Stmt
StmtList→ Stmt SmtList
Expr→ id Op id
Expr→ id Op num
Op→ >
Op→ <
}
```

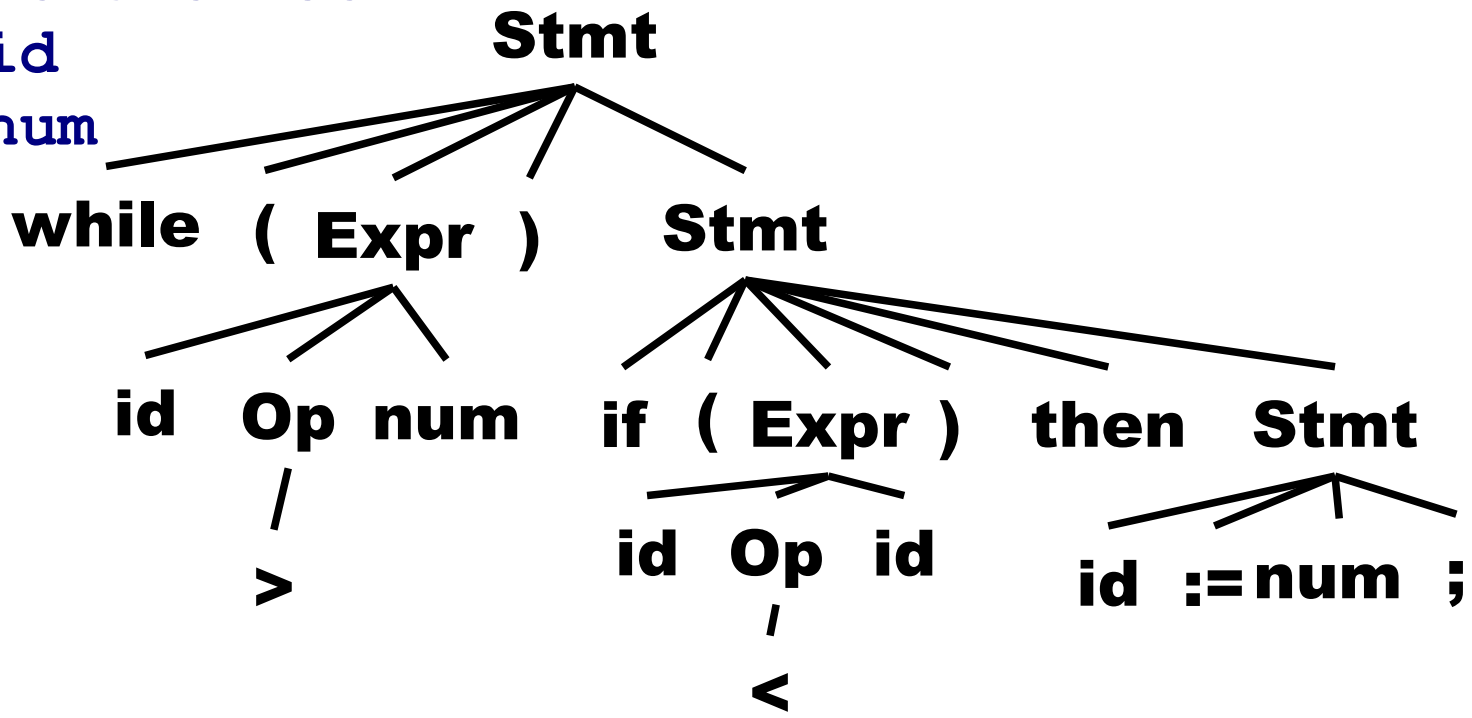



```
if (id > id) then id := id ;
```

Παράδειγμα (II.2)

```
P={
Stmt→ if ( Expr ) then Stmt
Stmt→ while ( Expr ) Stmt
Stmt→ id := id ;
Stmt→ id := num ;
Stmt→ begin StmtList end
StmtList→ Stmt
StmtList→ Stmt SmtList
Expr→ id Op id
Expr→ id Op num
Op→ >
Op→ <
}
```

```
while ( id > num)
if (id < id) then
  id := num;
```



Αναπαράσταση Γραμματικών

- Καθαρή **Backus-Naur** Μορφή: Απλούστερος, συντομότερος και κομψότερος τρόπος αναπαράστασης γραμματικών.
  - Μη τερματικά σύμβολα οριοθετούνται από γωνιακές παρενθέσεις, "**<**" και "**>**"
  - Το σύμβολο → αντικαθίσταται από "**::=**" .
  - Το σύμβολο "**|**" χρησιμοποιείται για να διαχωρίσει πολλαπλά δεξιά μέλη που αντιστοιχούν σε ένα αριστερό μέλος ενός κανόνα.

E→E+T	**<E>::=<E>+<T>|<E>-<T>**
E→E-T	
T→1	**<T>::= 1|2|3 ... |0**
T→2 ...	

Παραλλαγές Backus Naur Form (BNF)

- Υπάρχουν πολλές παραλλαγές και επεκτάσεις της BNF.
- **Extended BNF** (EBNF)
  - Τερματικά σύμβολα σε εισαγωγικά και μη χρήση γωνιακών παρενθέσεων.
  - Συμβολοσειρές σε αγκύλες "**[**" και "**]**"είναι προαιρετικές.
  - Επιτρέπεται χρήση των συμβόλων "*" και "+" με την ίδια σημασία που έχουν και στις κανονικές εκφράσεις.
  - Επιτρέπονται οι παρενθέσεις για ομαδοποίηση συμβόλων.

Παράδειγμα EBNF

Stmt→ if (Expr) then Stmt
Stmt→ if (Expr) Stmt
Stmt→ id := id ;
Stmt→ begin StmtList end
StmtList→ Stmt
StmtList→ Stmt SmtList
Expr→ id Op id
Expr→ id Op num
Op→ >
Op→ <

EBNF Μορφή:

Stmt::="if" "(" Expr ")" ["then"] Stmt | "id" ":=" "id" ";" | "begin" Stmt^{+} "end"
Expr ::= "id" Op "id" | "id" Op "num"
Op ::= ">" | "<"

Αυτόματα Στοίβας
(push-down Automata)

Αυτόματα Στοίβας

- Αφηρημένες Μηχανές οι οποίες λειτουργούν σαν "αναγνωριστές" συμβολοσειρών γραμματικών χωρίς συμφραζόμενα.
- Αποδεικνύεται ότι κάθε ΓΧΣ αντιστοιχεί σε ένα Αυτόματο στοίβας.

Ορισμός Αυτομάτων Στοίβας

- Ένα αυτόματο στοίβας ορίζεται ως $M=\{\Sigma,Q,H,\delta,q_0,h_0,F\}$ όπου
  - Σύνολο συμβόλων εισόδου (αλφάβητο) Σ
  - Πεπερασμένο σύνολο καταστάσεων Q
  - Η είναι το αλφάβητο στοίβας
  - Συνάρτηση μετάβασης δ: $Q\times H\times(\Sigma\cup\{\varepsilon\})\rightarrow P(H^*\times Q)$
  - Αρχική κατάσταση q_0
  - Αρχικό σύμβολο στοίβας h_0
  - Σύνολο τελικών καταστάσεων $F \subseteq Q$

Συνάρτηση μετάβασης δ

- Δύο ειδών μεταβάσεις: **μηδενικές** και **μη-μηδενικές (μεταβάσεις ανάγνωσης)** ανάλογα αν "καταναλώνεται" σύμβολο από την συμβολοσειρά εισόδου.
- Πεδίο τιμών του αυτομάτου είναι ένα σύνολο που περιέχουν κινήσεις. Κάθε κίνηση **(γ,q')** έχει δύο αποτελέσματα:
  - μεταφορά του ΑΣ στην κατάσταση q'
  - τοποθέτηση στην στοίβα της συμβολοσειράς γ ($\gamma \in H^*$) στην στοίβα του αυτομάτου από αριστερά προς τα δεξιά.

Λειτουργία ΑΣ

- Αρχικά το ΑΣ βρίσκεται στην q_0 με αρχικό σύμβολο στοίβας το h_0.
- Έστω ότι το ΑΣ είναι σε μια κατάσταση q, με h το πρώτο σύμβολο της στοίβας και a το επόμενο σύμβολο εισόδου:
  - αφαιρείται το h από την στοίβα,
  - αν το σύνολο δ(q,h,a) δεν είναι κενό, επιλέγεται μια κίνηση από αυτό, έστω (γ,q') και το αυτόματο μεταβαίνει στην κατάσταση q', ενώ τοποθετείται στην στοίβα η γ και καταναλώνεται το a,
  - αν το σύνολο δ(q,h,ε) δεν είναι κενό, επιλέγεται μια κίνηση όπως παραπάνω όμως το a δεν καταναλώνεται.

Λειτουργία ΑΣ

- Το ΑΣ αναγνωρίζει μια συμβολοσειρά εισόδου, όταν μετά το τέλος της εκτέλεσης έχει εξαντληθεί η συμβολοσειρά εισόδου και το ΑΣ βρίσκεται σε μια από τις τελικές καταστάσεις.
  - Μη-ντετερμινισμός: αρκεί μια από τις δυνατές εκτελέσεις να οδηγεί σε τελική κατάσταση.
- **Συνολική κατάσταση (configuration)** είναι η τριάδα **(q,γ,α)**, όπου
  - q είναι η τρέχουσα κατάσταση του ΑΣ,
  - γ η συμβολοσειρά που υπάρχει στη στοίβα και
  - α η μη-αναγνωσμένη συμβολοσειρά εισόδου.

Αναπαράσταση αυτομάτων

- **Πίνακας Μετάβασης:** η πρώτη γραμμή είναι το αλφάβητο της στοίβας (με το σύμβολο ┤) και η πρώτη στήλη οι καταστάσεις του αυτομάτου.
- Μια κίνηση αναπαριστάται (γ,q) από
  - pop, αφαίρεση h από στοίβα
  - push(γ), τοποθέτηση συμβόλων στη στοίβα
  - move(q), μετακίνηση στην κατάσταση q
  - push(ε), και move(q) (αν το ΣΑ είναι στην q) παραλείπονται.
  - Το h τοποθετείται εκ νέου στην κορυφή της στοίβας τότε η pop παραλείπεται.
  - read(a) δηλώνει μετάβαση ανάγνωσης, keep μηδενική μετάβαση.

Παράδειγμα

- Αυτόματο το οποίο αναγνωρίζει συμβολοσειρές της μορφής (()()) (ισοζυγισμένες παρενθέσεις).
  - Σ ={ ")","(" }
  - Q = {S,T}
  - H={X,I}
  - δ(S,X,"(")={(XI,S)}
    δ(S,X,ε)={(X,T)}
    δ(S,I,"(")={(I,S}
    δ(S,I,")")={(ε,S}
  - q_0=S
  - h_0=X
  - F={T}

Πίνακας Μετάβασης

- δ(S,X,"(")={(XI,S)}
  δ(S,X,ε)={(X,T)}
  δ(S,I,"(")={(II,S}
  δ(S,I,")")={(ε,S}

	X	I	⊣
S	read("(") => push(I) keep => move(T)	read("(") => push(I) read(")") => push(I)	
T			success

Παράδειγμα Εκτέλεσης

- **Συμβολοσειρά εισόδου**: (()())
- **Μεταβάσεις:**

δ(S,X,"(")={(XI,S)}
δ(S,I,"(")={(II,S}
δ(S,X,ε)={(X,T)}
δ(S,I,")")={(ε,S}

	Κατάσταση	Στοίβα bottom top	Είσοδος
1	S	X	(()())
2	S	XI	()())
3	S	XII	)())
4	S	XI	())
5	S	XII	))
6	S	XI	)
7	S	X	ε
8	T		

Είδη Πεπερασμένων Αυτομάτων

- Ένα ΑΣ ονομάζεται **πραγματικού χρόνου**, όταν δεν περιέχει ε-μετάβασεις και **απλό** όταν έχει μόνο μια κατάσταση.
- Ντετερμινιστίκο αυτόματο:
  - Για κάθε $a \in \Sigma \cup \{\varepsilon\}, q \in Q, h \in H$ το $\delta(q,h,a)$ έχει το πολύ ένα στοιχείο.
  - Αν υπάρχει $\delta(q,h,\varepsilon)$ τότε δεν υπάρχουν μεταβάσεις $\delta(q,h,a)$ για κάθε $a \in \Sigma$.
- Γενικά, δεν υπάρχει ισοδυναμία ανάμεσα σε μη-ντετερμινιστικά και ντετερμινιστικά ΑΣ.

ΑΣ και Μεταγλωττιστές

- Κάθε ΓΧΣ έχει αντίστοιχο ΜΑΣ.
- Ενδιαφέρον (λόγω απόδοσης) έχουν τα ΝΑΣ, άρα ενδιαφέρον για υποσύνολο γλωσσών
  - **Μη-ντετερμινιστικές γλώσσες χωρίς συμφραζόμενα**.
  - Ειδικότερα ενδιαφερόμαστε για γραμματικές LL(1) και LR(1) που αντιστοιχούν σε ΝΑΣ και έχουν έυκολη υλοποίηση.
    - Υποσύνολα των ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα.
- Θα δούμε στη συνέχεια πως "κατασκευάζονται" τέτοια αυτόματα.

Γλώσσες Προγραμματισμού Μεταγλωττιστές

Σχεδιασμός μιας Γραμματικής

Ισοδύναμες Γραμματικές

- Δύο γραμματικές που παράγουν την ίδια γλώσσα ονομάζονται **ισοδύναμες.**
- Συχνά στις γραμματικές γίνονται **μετασχηματισμοί** οι οποίοι δίνουν μια ισοδύναμη γραμματική από μια αρχική περισσότερο πρόσφορη για υλοποίηση.
- Ιδιαίτερα σημαντικό στις **διφορούμενες** γραμματικές.

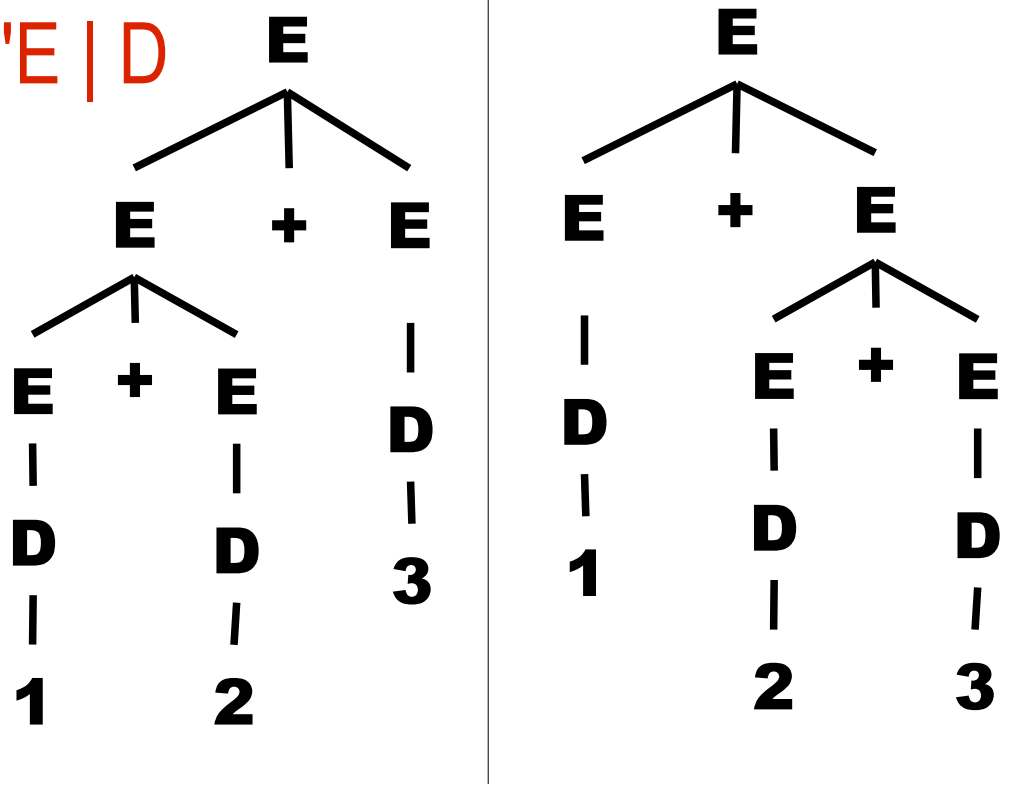
Διφορούμενες Γραμματικές

- Όταν μια συμβολοσειρά μπορεί να αντιστοιχεί σε δύο συντακτικά δένδρα για μια γραμματική τότε η γραμματική ονομάζεται **διφορούμενη**.
- Για παράδειγμα:

E::=E"+"E | E"-"E| E"*"E | E"/"E | D

D::="1"|"2"|"3""...|"9"|"0"

Συμβολοσειρά:
1+2+3



Μετασχηματισμοί

- Διφορούμενες γραμματικές πρέπει να αποφεύγονται κατά την κατασκευή των μεταγλωττιστών.
  - Σημασιολογικές Ασάφειες
- Αρκετές φορές είναι δυνατό να γίνουν κατάλληλοι μετασχηματισμοί οι οποίοι θα οδηγήσουν σε μια **μη-διφορούμενη ισοδύναμη** γραμματική.

Τελεστές και Γραμματικές

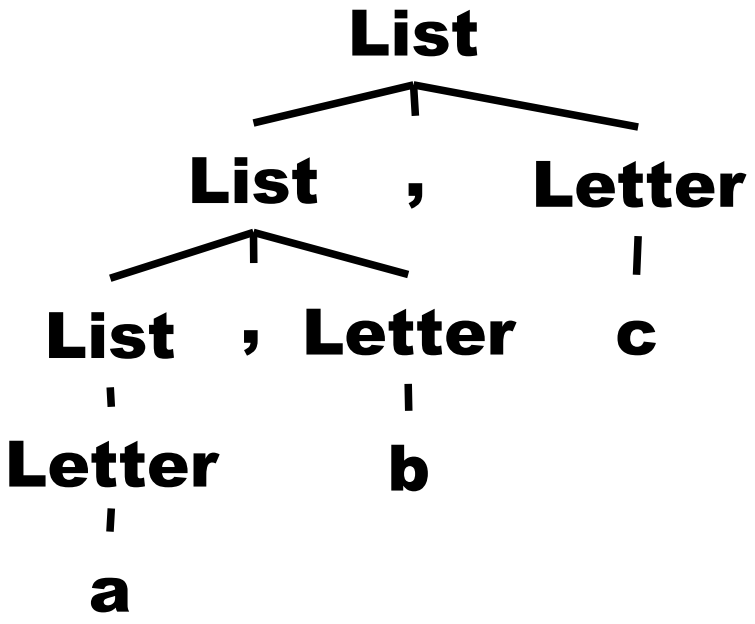
- Έκφραση της **προτεραιότητας** και **προσεταιριστικότητας** των τελεστών σε μια γραμματική.
- **Προτεραιότητα τελεστών**: ο τελεστής με την μεγαλύτερη προτεραιότητα παίρνει τα ορίσματα του πριν από τους άλλους τελεστές.
- **Προσεταιριστικότητα**:
  - **Αριστερά προσεταιριστικός** τελεστής: Αριστερά του τελεστή υπάρχει μια υποέκφραση ίδιας προτεραιότητας.
  - **Δεξιά προσεταιριστικός** τελεστής: Δεξιά του τελεστή υπάρχει μια υποέκφραση ίδιας προτεραιότητας.

Τελεστές και Γραμματικές

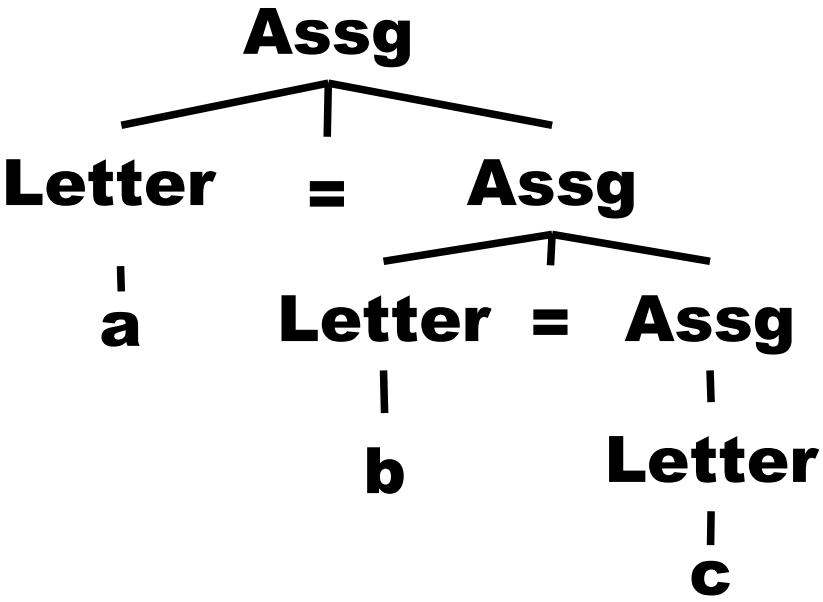
- Πώς μπορεί να εκφραστεί η προτεραιότητα και προσεταιριστικότητα μέσω μιας μη-διφορούμενης γραμματικής;
  - **Προτεραιότητα**: Για κάθε ομάδα τελεστών ίσης προτεραιότητας, δημιουργούμε ένα μη-τερματικό σύμβολο.
  - **Προσεταιριστικότητα**: Για κάθε τελεστή ορίζεται κατάλληλα ο αντίστοιχος κανόνας παραγωγής.

Προσεταιριστικότητα

- Αριστερά προσεταιριστικός τελεστής
  List ::= List "," Letter
  Letter ::= "a"|"b"|...|"z"
  Συμβολοσειρά a,b,c



- Δεξιά προσεταιριστικός τελεστής
  Assg ::= Letter "="Assg | Letter
  Letter ::= "a"|"b"|...|"z"
  Συμβολοσειρά a = b = c



Παράδειγμα (i)

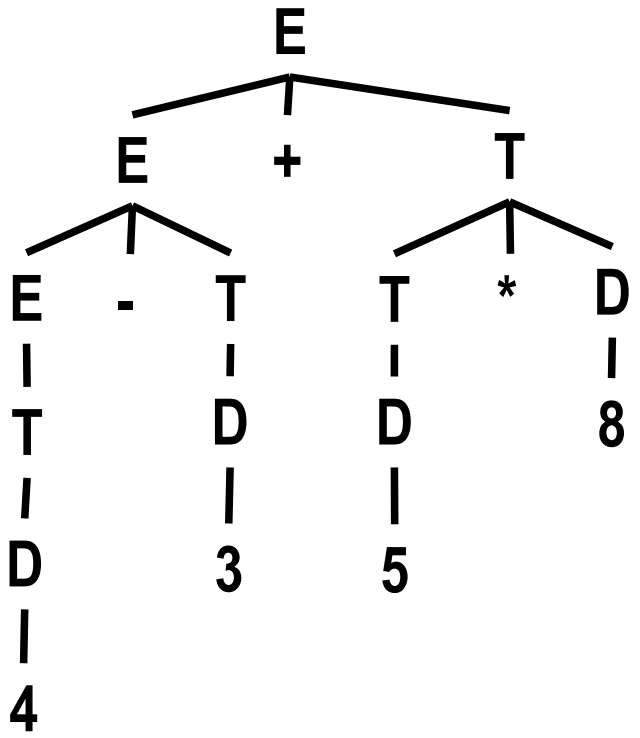
- Γραμματική:
  E::=E"+"E | E"-"E| E"*"E | E"/"E | D
  D::="1"|"2"|"3""...|"9"|"0"
- Δύο κατηγορίες τελεστών ως προς την προτεραιότητα:
  - αριστερά προσεταιριστικοί "+", "-"
    - δημιουργία μη-τερματικού E
  - αριστερά προσεταιριστικοί "*" , "/"
    - δημιουργία μη-τερματικού T.
  - το σύμβολο D παραμένει για να δηλώσει τις βασικές μονάδες.

Παράδειγμα (ii)

- Αρ. προσεταιριστικοί "+", "-" (συμβ. E)
- Αρ. προσεταιριστικοί "*" , "/" (συμβ. T)
- το σύμβολο D

*Συμβολοσειρά 4-3+5*8*

- E::=E"+"T | E"-"T|T
- T::=T"*"D|T"/"D|D
- D::="1"|"2"|...|"9"|"0"

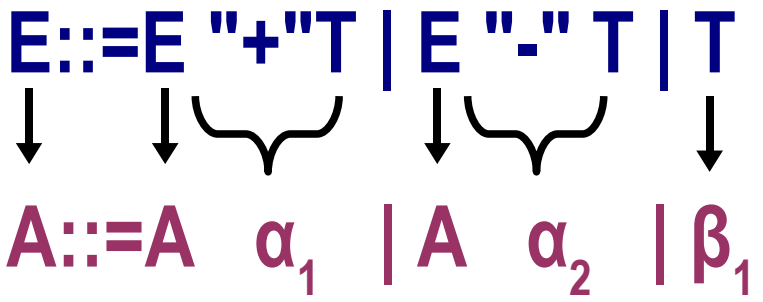


Απαλοιφή Αριστερής Αναδρομής

- **Αριστερά αναδρομικός** κανόνας με:
  - Άμεση αναδρομή A::= Aβ
  - Έμμεση αναδρομή A::=aβ, όπου σε μία ή περισσότερες παραγωγές η συμβολοσειρά α παράγει την Aγ.
  - Γενικότερα $A \rightarrow \alpha$ και $\alpha \Rightarrow^* A\beta$
- Η αριστερή αναδρομή είναι ανεπιθύμητη στην περίπτωση της **από-πάνω προς τα κάτω** ανάλυσης (top-down).
  - Γιατί;

Μέθοδος Απαλοιφής

- Έστω ο κανόνας
  $A::=A\alpha_1|A\alpha_2|...|A\alpha_n|\beta_1|...|\beta_m$
- Ο μετασχηματισμός περιλαμβάνει την εισαγωγή ενός νέου τερματικού Α' και την μεταγραφή του κανόνα στους ακόλουθους:
  $A::=\beta_1A'|...|\beta_mA'$
  $A'::=\alpha_1A'|\alpha_2A'|...|\alpha_nA'|\varepsilon$
- Παράδειγμα:



$A::=\beta_1A'$

γίνεται $E::=TE'$

$A'::=\alpha_1A'|\alpha_2A'|\varepsilon$

γίνεται $E'::="+"TE' | "-"TE'|\varepsilon$

Παράδειγμα Γραμματικής

- Γραμματική με Αριστερή Αναδρομή

  E::=E"+"T | E"-"T|T

  T::=T"*"D|T"/"D|D

  D::="1"|"2"|...|"9"|"0"

- Ισοδύναμη Γραμματική

  E::=TE'

  E'::="+"TE'|"-"TE'|ε

  T::=DT'

  T'::="*"DT'|"/"DT'|ε

  D::="1"|"2"|...|"9"|"0"

Αντικατάσταση

- Οι μετασχηματισμοί για τις έμμεσες αναδρομές περιλαμβάνουν **αντικαταστάσεις**, έτσι ώστε οι νέοι κανόνες που θα προκύψουν να είναι άμεσα αναδρομικοί, όπου και εφαρμόζεται η προηγούμενη μέθοδος.
- **Αντικατάσταση** είναι ο μετασχηματισμός στον οποίο αντικαθιστούμε ένα μη-τερματικό σύμβολο B στο δεξιό μέλος ενός κανόνα A με όλα τα εναλλακτικά δεξιά μέλη κανόνων που αφορούν το B.
- $A::=\alpha_1 B \alpha_2$ $A::=\alpha_1\beta_1\alpha_2|\alpha_1\beta_2\alpha_2|...|\alpha_1\beta_n\alpha_2$
- $B::=\beta_1|\beta_2|...|\beta_n$ $B::=\beta_1|\beta_2|...|\beta_n$

Παράδειγμα Έμμεσης Αναδρομής

- Έστω η γραμματική

  S::=A "a" | "b"

  A::=A "c" | S "d" | "f"

- Κάνοντας τις αντικαταστάσεις

  S::=A "a" | "b"

  A::=A "c" | A "a" "d"| "b" "d" | "f"

- Απαλείφοντας την άμεση αριστερή αναδρομή

  S::=A "a" | "b"

  A::="b" "d" A' | "f" A'

  A'::="c" A' | "a" "d" A' | ε

Αριστερή Παραγοντοποίηση

- Για την αποδοτική υλοποίηση των συντακτικών αναλυτών δεν πρέπει δύο εναλλακτικοί κανόνες για ένα μη-τερματικό να ξεκινούν με το ίδιο πρόθεμα. Δηλαδή:

  $A::=\alpha\beta_1|\alpha\beta_2|...|\alpha\beta_n|\gamma_1|\gamma_2|...|\gamma_n$

- Η γραμματική με αριστερή παραγοντοποίηση μετατρέπεται στην:

  $A::=\alpha B|\gamma_1|\gamma_2|...|\gamma_n$

  $B::=\beta_1|\beta_2|...|\beta_n$

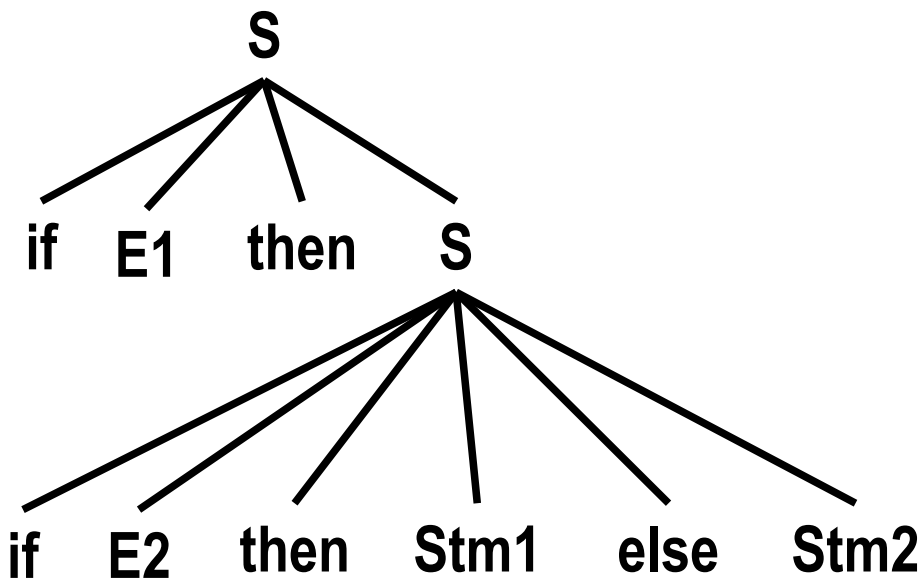
Το πρόβλημα του μετέωρου else (danfling else)

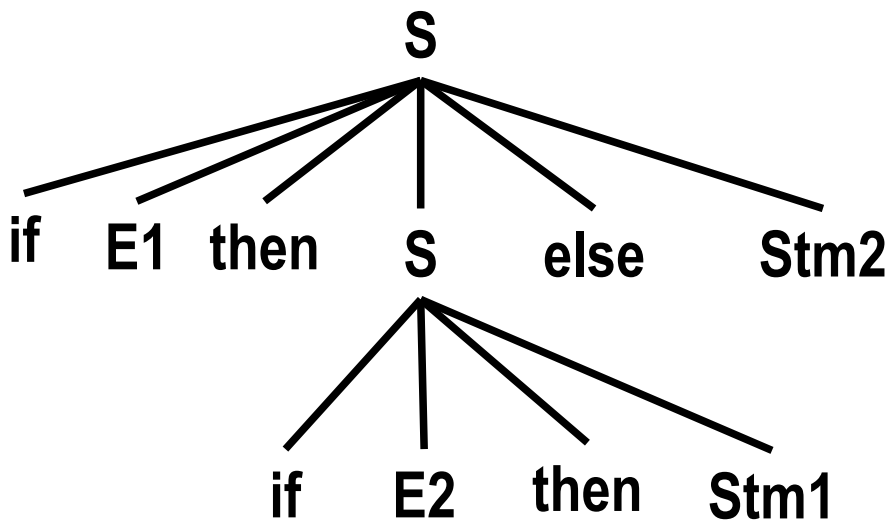
- Έστω μέρος μιας γραμματικής (other είναι οποιοδήποτε άλλο S)

  S ::= "if" E "then" S | "if" E "then" S else S | Other

- Ποιό είναι το συντακτικό δένδρο της

  if E1 then if E2 then Stm1 else Stm2





Μετατροπή σε μη-διφορούμενη γραμματική

- **Βασική ιδέα**: ανάμεσα σε ένα then και σε ένα else δεν πρέπει να υπάρχει ένα "ανοικτό" (unmatched) S. Ως "κλειστό" (matched) S θεωρούμε ένα πλήρες if-then-else ή οποιοδήποτε άλλο S.
  - Το else αντιστοιχεί στο "εσωτερικότερο" if-then.
- Άρα

  S ::= "if" E "then" S | "if" E "then" S else S | Other
- Γίνεται:

  S ::= Mtch | Umtc

  Mtch ::= "if" E "then" Mtch else Mtch | Other

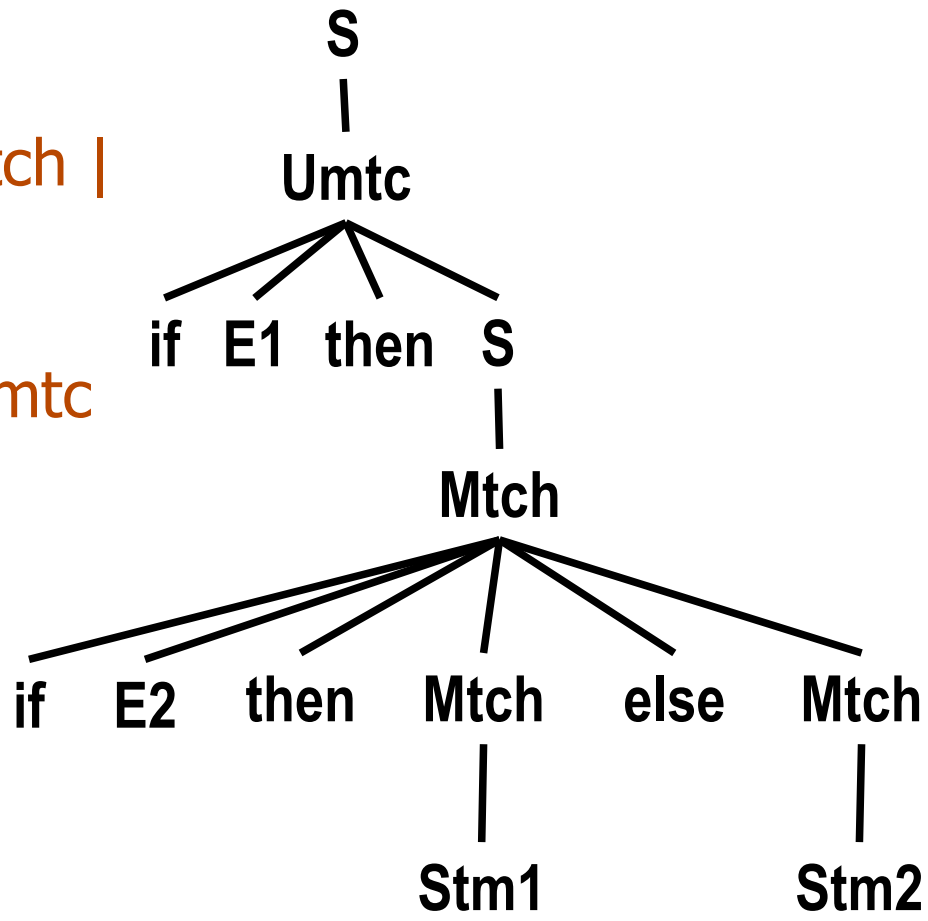
  Umtc ::= "if" E "then" S | "if" E "then" Mtch else Umtc

Το πρόβλημα του μετέωρου else (danfling else)

- Παραγόμενο δένρο

  if E1 then if E2 then Stm1 else Stm2

S::= Mtch | Umtc
Mtch::="if" E "then" Mtch "else" Mtch |
Other
Umtc::="if" E "then" S |
"if" E "then" Mtch "else" Umtc



Συναρτήσεις
FIRST και FOLLOW

Συναρτήσεις FIRST και FOLLOW

- Οι δύο συναρτήσεις ορίζονται για τα σύμβολα μιας γραμματικής και αφορούν την κατασκευή συντακτικών αναλυτών και των δύο τύπων.
- Έστω μια γραμματική G={T,N,P,S}
  - Η συνάρτηση FIRST(α) ορίζεται για κάθε συμβολοσειρά $\alpha \in (T \cup N)^*$ και ισούται με το **σύνολο των τερματικών συμβόλων** με τα οποία μπορεί να **ξεκινά** μια συμβολοσειρά η οποία παράγεται από την α.
  - Η συνάρτηση FOLLOW(A), όπου $A \in N$, είναι το **σύνολο των τερματικών συμβόλων** που μπορεί να **ακολουθούν** την A σε ένα προτασιακό τύπο.

Συνάρτηση FIRST

- Δεδομένης μιας γραμματικής G={T,N,P,S}, και α $\in$ (T$\cup$N)* (δηλαδή κάθε συμβολοσειρά που αποτελείται από τερματικά και μη-τερματικά σύμβολα της G) η συνάρτηση FIRST(α) $\subseteq$ T$\cup${ε}, ορίζεται ως:
  - Αν υπάρχει α $\Rightarrow^*$ aβ, όπου a είναι τερματικό σύμβολο, τότε ισχύει a$\in$ FIRST(α).
  - Αν υπάρχει α $\Rightarrow^*$ ε, τότε ε $\in$ FIRST(α).

Υπολογισμός Συνάρτησης FIRST (i)

- Για κάθε σύμβολο X της γραμματικής G, ισχύουν οι ακόλουθοι κανόνες υπολογισμού:
  - Αν X τερματικό τότε FIRST(X) ={X}
  - Αν $X \in N$ και υπάρχει κανόνας $X \rightarrow \varepsilon$, τότε **FIRST(X)=FIRST(X)∪{ε}**
  - Αν $X \in N$, και για κάθε κανόνα $X \rightarrow A_1, A_2 \ldots A_n$, τότε **FIRST(X) = FIRST(X)∪(FIRST(A_i) -{ε})**, για κάθε i, τέτοιο ώστε $\forall j \mid 0<j<i \ \ \varepsilon \in FIRST(A_j)$.
  - Εάν $\forall j \mid 0<j \leq n \ \ \varepsilon \in FIRST(A_j)$, τότε **FIRST(X)=FIRST(X)∪ {ε}**
- Οι κανόνες εφαρμόζονται μέχρι να μην μπορούν να μεταβληθούν τα σύνολα FIRST.

Υπολογισμός Συνάρτησης FIRST (ii)

- Για κάθε συμβολοσειρά $X_1,X_2,\ldots,X_n$
  - **$FIRST(X_1,\ldots,X_n) = FIRST(X_1) - \{\varepsilon\}$**
  - Για κάθε i | $1<i\leq n$, εάν $\forall$ j | $0<j<i$ $\varepsilon \in FIRST(X_j)$ τότε **$FIRST(X_1,\ldots,X_n)=FIRST(X_1,\ldots,X_n)\cup(FIRST(X_i)-\{\varepsilon\})$**
  - Εάν για κάθε $0<i\leq n$, $\varepsilon \in FIRST(X_i)$, τότε **$FIRST(X_1,\ldots,X_n) = FIRST(X_1,\ldots,X_n)\cup\{\varepsilon\}$**

Παράδειγμα Υπολογισμού FIRST

$E \rightarrow TE'$
$E' \rightarrow +TE'$
$E' \rightarrow \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT'$
$T' \rightarrow \varepsilon$
$F \rightarrow id$
$F \rightarrow (E)$

$FIRST(E)=FIRST(T)-\{\varepsilon\}=\{id,(\}$
$FIRST(E')= \{+,\varepsilon\}$

$FIRST(T)=FIRST(F)-\{\varepsilon\}=\{id,(\}$

$FIRST(T')=\{*,\varepsilon\}$

$FIRST(F) = FIRST(id) \cup FIRST((E))=\{id,(\}$

$FIRST((E)) = \{(\}$
$FIRST(id)=\{id\}$

Συνάρτηση FOLLOW

- Δεδομένης μιας γραμματικής $G=\{T,N,P,S\}$, και $A \in N$, τότε η συνάρτηση $FOLLOW(A) \subseteq T \cup \{EOF\}$, ορίζεται ως:
  - Αν υπάρχει παραγωγή $S \Rightarrow^* \alpha A a \beta$, όπου $\alpha,\beta \in (T \cup N)^*$ και a είναι τερματικό σύμβολο, τότε $a \in FOLLOW(\alpha)$.
  - Αν υπάρχει παραγωγή $S \Rightarrow^* \alpha A$, όπου $\alpha \in (T \cup N)^*$ τότε $EOF \in FOLLOW(\alpha)$.

Υπολογισμός της συνάρτησης FOLLOW

- **FOLLOW(S)={EOF}**
- Για κάθε κανόνα της μορφής $A \rightarrow \alpha B \beta$, όπου A και $B \in N$ και $\alpha, \beta \in (T \cup N)^*$, **FOLLOW(B)=FOLLOW(B)∪(FIRST(b)-{ε})**
- Εάν $\varepsilon \in$ FIRST(β) τοτε **FOLLOW(B)=FOLLOW(B)∪FOLLOW(A)**

- Το παραπάνω εκτελείται μέχρι τα σύνολα FOLLOW να μην μεταβάλλονται.

Παράδειγμα Υπολογισμού FOLLOW

FIRST(E)=FIRST(T)=FIRST(F) ={id,(}
FIRST(E')= {+,ε} FIRST(T')={*,ε}

E→TE'
E'→+TE'
E'→ε
T→FT'
T'→*FT'
T'→ε
F→id
F→(E)

FOLLOW(E)={EOF} (αρχικό σύμβολο)

- **Υπολογισμός FOLLOW(E)**

F→(E)

FOLLOW(E)=FOLLOW(E) ∪ (FIRST( ))-{ε})={EOF,)}

- **Υπολογισμός FOLLOW(E')**

E→TE'

FOLLOW(E')=FOLLOW(E') ∪ (FIRST(ε)-{ε})={}
FOLLOW(E')=FOLLOW(E') ∪ FOLLOW(E)={EOF,)}

E'→+TE'

FOLLOW(E')=FOLLOW(E') ∪ (FIRST(ε)-{ε})={EOF,)}
FOLLOW(E')=FOLLOW(E') ∪ FOLLOW(E')={EOF,)}

Παράδειγμα Υπολογισμού FOLLOW

FIRST(E)=FIRST(T)=FIRST(F) ={id,)}
FIRST(E')= {+,ε} FIRST(T')={*,ε}

E→TE'
E'→+TE'
E'→ε
T→FT'
T'→*FT'
T'→ε
F→id
F→(E)

- **Υπολογισμός FOLLOW(T)**

E→TE'

FOLLOW(T)=FOLLOW(T) ∪ (FIRST(E')-{ε})={+}
FOLLOW(T)=FOLLOW(T) ∪ FOLLOW(E')={EOF,),+}

E'→+TE'

FOLLOW(T)=FOLLOW(T) ∪ (FIRST(ε)-{ε})={}
FOLLOW(T)=FOLLOW(T) ∪ FOLLOW(E')={EOF,),+}

Παράδειγμα Υπολογισμού FOLLOW

FIRST(E)=FIRST(T)=FIRST(F) ={id,)}
FIRST(E')= {+,ε} FIRST(T')={*,ε}

E→TE'
E'→+TE'
E'→ε
T→FT'
T'→*FT'
T'→ε
F→id
F→(E)

- **Υπολογισμός FOLLOW(T')**

T→FT'

FOLLOW(T')=FOLLOW(T') ∪ (FIRST(ε)-{ε})={}
FOLLOW(T')=FOLLOW(T') ∪ FOLLOW(T)={EOF,),+}

T'→*FT'

FOLLOW(T')=FOLLOW(T')∪(FIRST(ε)-{ε})={EOF,),+}
FOLLOW(T')=FOLLOW(T')∪FOLLOW(T')={EOF,),+}

Παράδειγμα Υπολογισμού FOLLOW

FIRST(E)=FIRST(T)=FIRST(F) ={id,)}
FIRST(E')= {+,ε} FIRST(T')={*,ε}

E→TE'
E'→+TE'
E'→ε
T→FT'
T'→*FT'
T'→ε
F→id
F→(E)

- **Υπολογισμός FOLLOW(F)**

T→FT'

FOLLOW(F)=FOLLOW(F) ∪ (FIRST(T')-{ε})={*}
FOLLOW(F)=FOLLOW(F) ∪ FOLLOW(T)={EOF,),+,*}

T'→*FT'

FOLLOW(F)=FOLLOW(F)∪(FIRST(T')-{ε})={EOF,),+,*}
FOLLOW(F)=FOLLOW(F)∪FOLLOW(T')={EOF,),+,*}

Παράδειγμα Υπολογισμού FIRST & FOLLOW

$E \rightarrow TE'$

$E' \rightarrow +T$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *F$

$T' \rightarrow \varepsilon$

$F \rightarrow id$

$F \rightarrow (E)$

- **FIRST(E)=FIRST(T)=FIRST(F) ={id,(}**
- **FIRST(E')= {+,ε}**
- **FIRST(T')={*,ε}**

- **FOLLOW(E)={EOF,)}**
- **FOLLOW(E')={EOF,)}**
- **FOLLOW(T)={EOF,),+}**
- **FOLLOW(T')={EOF,),+}**
- **FOLLOW(F)={EOF,),+,*}**

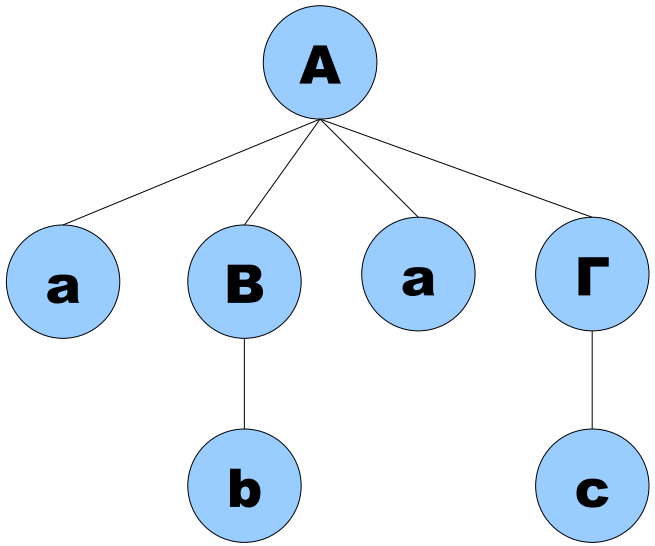
Συντακτική Ανάλυση από Πάνω προς τα Κάτω (top-down)

Συντακτική Ανάλυση από Πάνω προς τα Κάτω

- Η συντακτική ανάλυση ξεκινά από το αρχικό σύμβολο της γραμματικής (ρίζα του συντακτικού δένδρου) και αντικαθιστά αυτό με τα σύμβολα από ένα κανόνα παραγωγής. Η διαδικασία συνεχίζεται με τους επόμενους κόμβους μέχρι όλα τα φύλλα του δένδρου να είναι τερματικά σύμβολα της γραμματικής.
  - **Με ποια σειρά θα εξεταστούν οι κόμβοι παιδιά ενός κόμβου;**
  - **Ποιος από τους πιθανούς εναλλακτικούς κανόνες θα χρησιμοποιηθεί για την αντικατάσταση του μη-τερματικού συμβόλου;**

Με ποια σειρά θα εξεταστούν οι κόμβοι παιδιά ενός κόμβου;

- Οι πλειοψηφία των συντακτικών αναλυτών επεκτείνουν τους νέους κόμβους από τα αριστερά προς τα δεξιά (**L**eft-to-right) σύμφωνα με την σειρά εμφάνισης τους στο δεξιό μέλος του αντίστοιχου κανόνα παραγωγής.
  - A::="a"B"a"Γ
  - B::="b"
  - Γ::="c"



Ποιος από τους πιθανούς εναλλακτικούς κανόνες θα χρησιμοποιηθεί για την αντικατάσταση του μη-τερματικού συμβόλου;

- Επιλογή κατάλληλου κανόνα είναι ένα δύσκολο ζήτημα (για να λυθεί αποδοτικά).
  - Η περίπτωση της οπισθοδρόμησης δεν είναι αποδεκτή.
- Συνήθως ο ΣΑ διαβάζει ένα ή περισσότερα σύμβολα ώστε να αποφασίσει ποιος είναι ο κατάλληλος κανόνας (look-ahead symbols).
  - **LL(k)**: ΣΑ που διαβάζει την συμβολοσειρά από αριστερά προς τα δεξιά (Left-to-right), κατασκευάζει την αριστερότερη παραγωγή (Leftmost derivation) και απαιτεί k σύμβολα εισόδου.
  - π.χ. LL(1) ΣΑ

LL(1) Γραμματικές

- **LL(1) γραμματική**: Αναγνώσιμη από ένα LL(1) ΣΑ (αναδρομικής κατάβασης ή ΑΣ).
  - Υποσύνολο των γραμματικών χωρίς συμφραζόμενα, αρκετά πλούσιο όμως για να περιγράψει τις περισσότερες γλώσσες.
- Προϋποθέσεις:
  - Για κάθε ζεύγος κανόνων παραγωγής **A→α** και **A→β** πρέπει να ισχύει **FIRST(α)∩FIRST(β) =∅ .**
  - Αν το ε ανήκει στο FIRST(A) (για παράδειγμα υπάρχει κανόνας **A→ε**) πρέπει να ισχύει **FIRST(A)∩FOLLOW(A) = ∅ .**

LL(1) Γραμματικές

- Αποκλείονται οι γραμματικές:
  - Αριστερά αναδρομικές
  - Έχουν δύο εναλλακτικούς κανόνες με δεξιά μέλη που ξεκινούν με το ίδιο σύμβολο.
  - Έχουν δύο εναλλακτικούς κανόνες τα δεξιά μέλη των οποίων παράγουν την κενή συμβολοσειρά.
- Πως επιτυγχάνεται το παραπάνω;
  - Απαλοιφή αριστερής αναδρομής
  - Αντικατάσταση
  - Αριστερή παραγοντοποίηση

ΣΑ Αναδρομικής Κατάβασης

- Recursive Descent
- ΣΑ από πάνω προς τα κάτω, εύκολα κατασκευάσιμος χειρωνακτικά.
- Σε κάθε μη-τερματικό σύμβολο, αντιστοιχίζεται μια ρουτίνα η οποία το αναγνωρίζει.
- **Αποφυγή οπισθοδρόμησης**: Χρήση της συνάρτησης FIRST για την επιλογή της κατάλληλης ρουτίνας για την αναγνώριση, εξετάζοντας την επόμενη λεκτική μονάδα (lookahead).

ΣΑ Αναδρομικής Κατάβασης (ii)

- Έστω $A::=\alpha_1|\alpha_2|...|\alpha_n$ και η επόμενη λεκτική μονάδα είναι η token.
- Ο κώδικας για την αναγνώριση της Α είναι:

if token $\in$ FIRST(α_1) **then** <*κώδικας για την* α_1>

...

ifelse token $\in$ FIRST(α_n) **then** <*κώδικας για την* α_n>

ifelse $\varepsilon \notin$ FIRST(α_1)$\cup$...$\cup$FIRST(α_n) **then**

<*συντακτικό σφάλμα*>

ifelse token $\notin$ FOLLOW(A) **then**

<*συντακτικό σφάλμα*>

ΣΑ Αναδρομικής Κατάβασης (iii)

- Κώδικας για το δεξιό μέλος των κανόνων
  - Η κενή συμβολοσειρά δεν απαιτεί κώδικα
  - Ένα τερματικό t αναγνωρίζεται από τον κώδικα match(a):

    match(t : token)

    if lookahead = t **then** lookahead := nexttoken()

    else syntax_error
  - Ένα μη-τερματικό σύμβολο Α αναγνωρίζεται από την κλήση στη ρουτίνα που αναγνωρίζει το Α.
  - Παράθεση δύο συμβολοσειρών αβ αναγνωρίζεται καλώντας τον τον κώδικα για την α και τον κώδικα για την β.

Παράδειγμα Γραμματικής

$E \rightarrow TE'$

$E' \rightarrow +TE'$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'$

$T' \rightarrow \varepsilon$

$F \rightarrow id$

$F \rightarrow (E)$

- **FIRST(E)=FIRST(T)=FIRST(F) ={id,(}**
- **FIRST(E')= {+,ε}**
- **FIRST(T')={*,ε}**

- **FOLLOW(E)={EOF,)}**
- **FOLLOW(E')={EOF,)}**
- **FOLLOW(T)={EOF,),+}**
- **FOLLOW(T')={EOF,),+}**
- **FOLLOW(F)={EOF,),+,*}**

Παράδειγμα Γραμματικής

$E \rightarrow TE'$

$E' \rightarrow +TE'$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'$

$T' \rightarrow \varepsilon$

$F \rightarrow id$

$F \rightarrow (E)$

- **FIRST(TE') = {id,(}**

```
void E()
  if token = id or token = (
        call T() call E'()
  else syntax_error
```

- **FIRST(+T) = {+} FOLLOW(E')={EOF,)}**

```
void E'()
  if token = + then
        match(+) call T() call E'()
  elseif token not_in { EOF, ) }  then syntax_error
```

Παράδειγμα Γραμματικής

$E \rightarrow TE'$

$E' \rightarrow +TE'$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'$

$T' \rightarrow \varepsilon$

$F \rightarrow id$

$F \rightarrow (E)$

- **FIRST(FT') = {id,(}**

```
void T()
  if token = id or token =(
        call F() call T'()
  else syntax_error
```

- **FIRST(*F) = {*} FOLLOW(T')={EOF,),+}**

```
void T'()
  if token = * then
        match(*) call F() call T'()
  elseif token  not_in { EOF,),+} then syntax_error
```

Παράδειγμα Γραμματικής

$E \rightarrow TE'$

$E' \rightarrow +TE'$

$E' \rightarrow \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'$

$T' \rightarrow \varepsilon$

$F \rightarrow id$

$F \rightarrow (E)$

- **FIRST(id) = {id} FIRST((E)) = {(}**

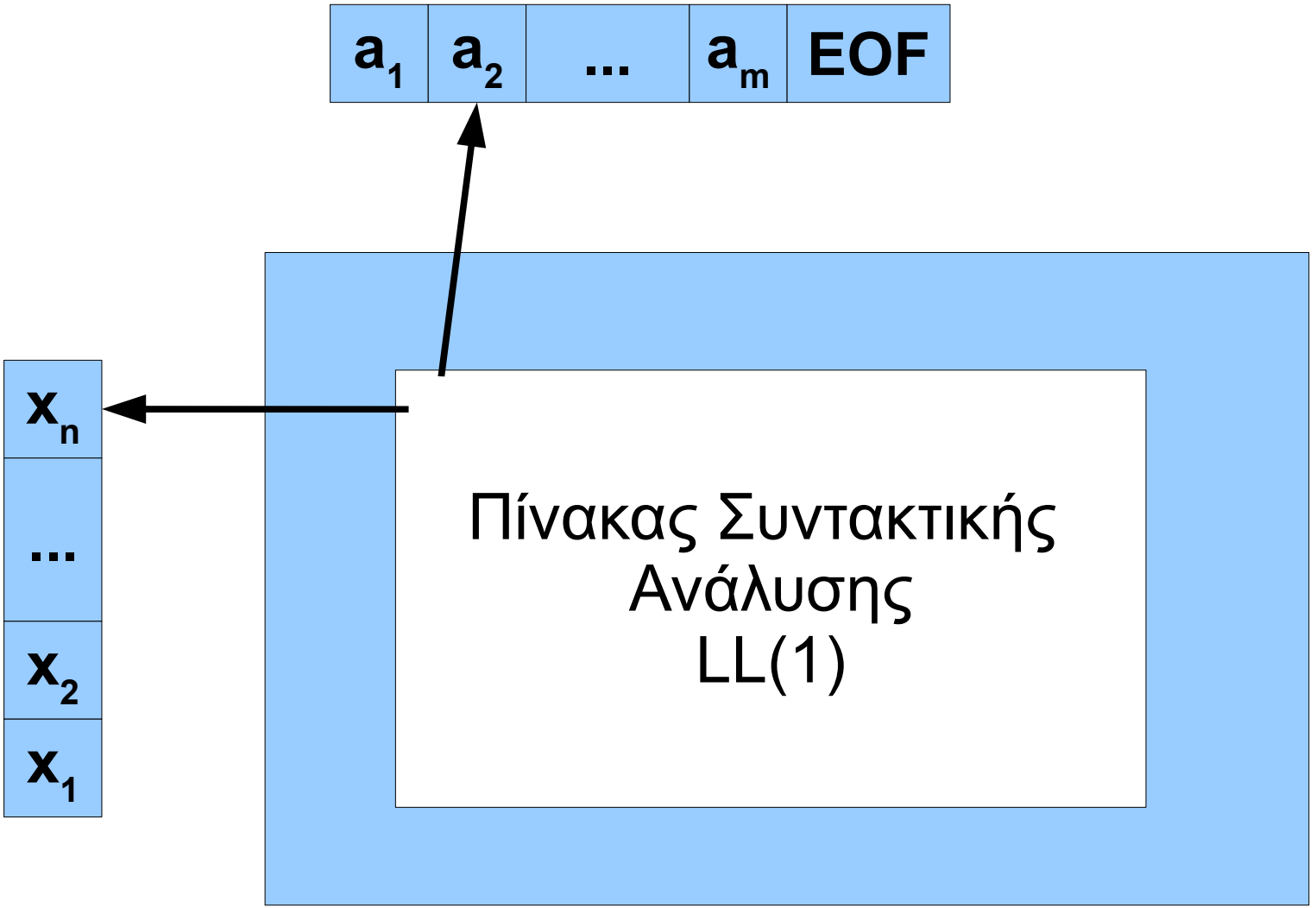
```
void F()
  if token = id then match(id)
  elseif token = ( then
        match( ( ) call E() match( ) )
  else syntax_error

void syntaxAnalysis()
  token = nexttoken()
  call E()
  if token =\= EOF then syntax_error
```

Μη-αναδρομικοί συντακτικοί αναλυτές LL(1)

- Είναι δυνατό να κατασκευαστεί ένα ειδικό αυτόματο στοίβας για μια LL(1) γραμματικοί.
- Συνήθως τέτοιοι ΣΑ κατασκευάζονται με αυτόματο τρόπο.
- Το αυτόματο αποτελείται από:
  - **ενδιάμεση μνήμη**, από όπου διαβάζεται η συμβολοσειρά εισόδου,
  - **στοίβα**, όπου τοποθετούνται τα τερματικά και μη-τερματικά σύμβολα της γραμματικής.
  - **πίνακας συντακτικής ανάλυσης**, με μια γραμμή για κάθε μη-τερματικό σύμβολο και μια στήλη για κάθε τερματικό σύμβολο της γραμματικής και το σύμβολο EOF.

ΣΑ LL(1) Αυτομάτου Στοίβας



Κατασκευή Πίνακα Συντακτικής Ανάλυσης

- Για κάθε κανόνα παραγωγής $A \rightarrow \beta$:
  - Για κάθε τερματικό σύμβολο a | $a \in$ FIRST(β), πρόσθεσε τον κανόνα $A \rightarrow \beta$ στη θέση M[A,a]
  - Αν $\varepsilon \in$ FIRST(β), τότε για κάθε τερματικό σύμβολο $a \in$ FOLLOW(A), πρόσθεσε τον κανόνα $A \rightarrow \beta$ στην θέση M[A,a].
  - Αν $\varepsilon \in$ FIRST(β) και EOF $\in$ FOLLOW(A) τότε πρόσθεσε τον κανόνα $A \rightarrow \beta$ στην θέση M[A,EOF]
  - Όσες θέσεις του πίνακα μένουν κενές συμπληρώνονται με την ένδειξη error (ή απλώς υπονοείται).

Παράδειγμα Κατασκευής Πίνακα Συντακτικής Ανάλυσης

E→TE'
E'→+TE'
E'→ε
T→FT'
T'→*FT'
T'→ε
F→id
F→(E)

- **FIRST(E)=FIRST(T)=FIRST(F) ={id,(}**
- **FIRST(E')= {+,ε}**
- **FIRST(T')={*,ε}**
- **FOLLOW(E)={EOF,)}**
- **FOLLOW(E')={EOF,)}**
- **FOLLOW(T)={EOF,),+}**
- **FOLLOW(T')={EOF,),+}**
- **FOLLOW(F)={EOF,),+,*}**

	id	+	*	(	)	EOF
E	*E → TE'*			*E → TE'*		
E'		*E' →+TE'*			*E' →ε*	*E' →ε*

Παράδειγμα Κατασκευής Πίνακα Συντακτικής Ανάλυσης

- Ο πλήρης πίνακας για την συγκεκριμένη γραμματική είναι:

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \text{id}$			$F \rightarrow (E)$		

Λειτουργία ΣΑ LL(1)

- Συμπίπτει με την λειτουργία του ΑΣ.
- Μέχρι η στοίβα να είναι κενή:
  - Αν στην κορυφή της στοίβας βρίσκεται **το τερματικό σύμβολο a** το οποίο συμπίπτει με το επόμενο σύμβολο της συμβολοσειράς εισόδου, τότε το a αφαιρείται από τη στοίβα και το αυτόματο προχωρά στο επόμενο σύμβολο.
  - Αν στην κορυφή της στοίβας είναι το **μη-τερματικό A**, και το επόμενο σύμβολο της συμβολοσειράς εισόδου τότε είναι το a, και υπάρχει κανόνας **A→β** στο **M[A,a],** τότε το A αφαιρείται από τη στοίβα και τοποθετούνται τα σύμβολα του β στη στοίβα.
  - Αλλιώς σφάλμα.

Παράδειγμα: Συμβολοσειρά id+id*id

	Στοίβα	Είσοδος	Κανόνας/Κίνηση
0	*E*	id + id * id EOF	*E* → *T E'*
1	*E' T*	id + id * id EOF	*T* → *F T'*
2	*E' T' F*	id + id * id EOF	*F* → id
3	*E' T'* id	id + id * id EOF	pop(id)
4	*E' T'*	+ id * id EOF	*T'* → ε
5	*E'*	+ id * id EOF	*E'* → + *T E'*
6	*E' T*+	+id * id EOF	pop(+)
7	*E' T*	id * id EOF	*T* → *F T'*
8	*E' T' F*	id * id EOF	*F* → id
9	*E' T'* id	id * id EOF	pop(id)
10	*E' T'*	* id EOF	*T'* → * *F T'*
11	*E' T' F**	*id EOF	pop(*)
12	*E' T' F*	id EOF	*F* → id
13	*E' T'* id	id EOF	pop(id)
14	*E' T'*	EOF	*T'* →ε
15	*E'*	EOF	*E'* →ε
16	ε	EOF	success

Ανάνηψη από Σφάλματα

- Η βασική ιδέα είναι να γίνει η ανίχνευση ενός σφάλματος και να:
  - γίνει η έξοδος μηνύματος λάθους με σαφήνεια και ακρίβεια
  - προχωρήσει η μεταγλωττιστής στην επεξεργασία της υπόλοιπης συμβολοσειράς εισόδου, ώστε να εντοπιστούν τυχόν άλλα σφάλματα.
  - μην καθυστερεί πολύ η μεταγλώττιση του προγράμματος.

Μέθοδος Πανικού

- Απλούστερη στρατηγική ανάνηψης από σφάλματα (panic mode).
- Ανάγνωση επόμενων λεκτικών μονάδων από την είσοδο, μέχρι να βρεθεί λεκτική μονάδα η οποία να ανήκει στο **σύνολο "συγχρονισμού"**.
  - Κατάλληλες λεκτικές μονάδες και διαχωριστές.
- Επιλογή των κατάλληλων λεκτικών μονάδων στο σύνολο εξαρτάται από την γλώσσα.
- Παραλείπει κάποια σφάλματα
  - Παραδοχή ότι τα δεν εμφανίζονται συχνά περισσότερα του ενός σφάλματα σε μια γραμμή.

Ανάνηψη από σφάλματα σε LL(1) ΣΑ

- Σφάλμα διαπιστώνεται όταν:
- Αν το **τερματικό σύμβολο** στην κορυφή της στοίβας δεν είναι ίδιο με το επόμενο τερματικό σύμβολο εισόδου.
- Αν για το **μη-τερματικό σύμβολο** της στοίβας και το επόμενο τερματικό σύμβολο της εισόδου, δεν υπάρχει κανόνας στον πίνακα συντακτικής ανάλυσης.

Μέθοδος Πανικού σε LL(1) ΣΑ (i)

- Καθορισμός τους συνόλου χρονισμού:
  - Τα σύμβολα στο σύνολο **FOLLOW(A)**. Αφαιρούμε από την είσοδο λεκτικές μονάδες μέχρι να φτάσουμε σε μια η οποία ανήκει στο FOLLOW(A), και αφαιρούμε το A από την στοίβα.
  - Δεν είναι πάντα η κατάλληλη μέθοδος, καθώς αν λείπει ένας διαχωριστικός χαρακτήρας (πχ ";" στη C) οι λεκτικές μονάδες με τις οποίες ξεκινούν οι επόμενες εκφράσεις δεν θα είναι στο FOLLOW(A). Προσθήκη στο σύνολο χρονισμού των "χαμηλότερων" στην ιεραρχία μη-τερματικών, συνόλων FIRST "υψηλότερων" μη-τερματικών.

Μέθοδος Πανικού σε LL(1) ΣΑ (ii)

- Μπορούν να προστεθούν σύμβολα από το FIRST(A) στο και αν βρεθεί κάποιο από αυτά στη συμβολοσειρά εισόδου να συνεχιστεί η αναγνώριση του Α.
- Εάν ένα μη-τερματικό παράγει την κενή συμβολοσειρά, τότε αυτή η παραγωγή προτιμάται.
- Αν ένα τερματικό είναι στην κορυφή της στοίβας τότε απλώς αφαιρείται από αυτή και δίνεται το κατάλληλο μήνυμα λάθους.

Ανάνηψη σε επίπεδο Φάσης (phase-level recovery)

- Σύμφωνα με την στρατηγική, σε κάθε κενή θέση του πίνακα συντακτικής ανάλυσης, αντιστοιχίζεται μια ρουτίνα σφάλματος, η οποία:
  - τυπώνει το κατάλληλο διαγνωστικό μήνυμα
  - Εισάγει/διαγράφει ή μεταβάλλει το σύμβολο εισόδου
  - Αφαιρεί το τρέχον σύμβολο από την στοίβα.
- Κίνδυνος ατερμόνων βρόχων
  - πχ. στην εισαγωγή ενός συμβόλου
- Η εισαγωγή ενός συμβόλου μπορεί να οδηγήσει σε μη επιτρεπτή συμβολοσειρά εισόδου.

Σύνοψη

- Συντακτική Ανάλυση
- Γραμματικές χωρίς συμφραζόμενα.
  - Αυτόματα Στοίβας
- Οι συναρτήσεις FIRST και FOLLOW.
- Συντακτικοί Αναλυτές από πάνω προς τα κάτω (top-down)