

ΣΓΠ. Μεθοδολογίες με Παραδείγματα

by Μπαζ.

Γραμματικές Χωρίς Συμφραζόμενα

Γραμματική Χωρίς Συμφραζόμενα (Γ.Χ.Σ)

$$G = (V, \Sigma, R, S)$$

όπου:

- V το αλφάβητο, ένα πεπερασμένο σύνολο συμβόλων
- $\Sigma \subseteq V$ τα τερματικά σύμβολα, ενώ $V - \Sigma$ τα μη-τερματικά σύμβολα
- R ένα σύνολο κανόνων παραγωγής της μορφής
 - $A \rightarrow u$, με
 - $A \in V - \Sigma$
 - $u \in V^*$ μία συμβολοσειρά από σύμβολα
- $S \in V - \Sigma$ ένα μη τερματικό σύμβολο που λέγεται αρχή της G

Παραγωγή

- Αν η συμβολοσειρά $w \in \Sigma^*$ παράγεται από την αρχή S μέσω κανόνων της G , γράφουμε $S \Rightarrow^* w$ και $w \in L(G)$ -η γλώσσα της G
- Σ' αυτή την περίπτωση λέμε ότι η w συμμορφώνεται στη σύνταξη των προτάσεων της γλώσσας της G

Παράδειγμα

$assign-stmt \rightarrow var := expr$	(1)
$expr \rightarrow term$	(2)
$expr + term$	(3)
$expr - term$	(4)
$term \rightarrow prim-expr$	(5)
$term \times prim-expr$	(6)
$term / prim-expr$	(7)
$prim-expr \rightarrow var$	(8)
NUM	(9)
$(expr)$	(10)
$var \rightarrow ID$	(11)
$ID [sbscr-lst]$	(12)
$sbscr-lst \rightarrow expr$	(13)
$sbscr-lst, expr$	(14)

- Αν η συμβολοσειρά $w \in \Sigma^*$ παράγεται από την αρχή S μέσω κανόνων της G , τότε γράφουμε $S \Rightarrow^* w$ και $w \in L(G)$ - η γλώσσα της G .
- Σ' αυτή την περίπτωση λέμε ότι η w συμμορφώνεται στη σύνταξη των προτάσεων της γλώσσας της G .
- Παραγωγή του προγράμματος: $a [4] := 3 \times 5$

$$\begin{aligned} assign-stmt &\xrightarrow{(1)} var := expr \xrightarrow{(12)} ID[sbscr-lst] := expr \xrightarrow{(13)} ID[expr] := expr \\ &\quad \xrightarrow{(2)} ID[term] := expr \xrightarrow{(5)} ID[prim-expr] := expr \xrightarrow{(9)} ID[NUM] := expr \\ &\quad \xrightarrow{(2)} ID[NUM] := term \xrightarrow{(6)} ID[NUM] := term \times prim-expr \\ &\quad \xrightarrow{(5)} ID[NUM] := prim-expr \times prim-expr \\ &\quad \xrightarrow{(9)} ID[NUM] := NUM \times prim-expr \xrightarrow{(9)} ID[NUM] := NUM \times NUM \end{aligned}$$

- **Αριστερή Παραγωγή:** όταν σε κάθε βήμα εφαρμόζουμε κανόνα στο πρώτο μη-τερματικό από τα αριστερά
- **Δεξιά Παραγωγή:** όταν σε κάθε βήμα εφαρμόζουμε κανόνα στο πρώτο μη-τερματικό από τα δεξιά

Παράγωγα ή Συντακτικά Δέντρα

- Η σύνταξη ενός προγράμματος αποτυπώνεται στο **παράγωγο/συντακτικό δέντρο**

Παράγωγο/Συντακτικό Δέντρο

Έστω $G = (V, \Sigma, R, S)$

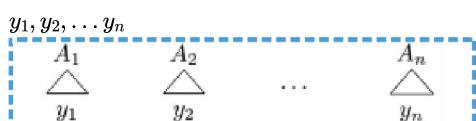
- Για κάθε σύμβολο $a \in \Sigma$, το γράφημα με ένα μόνο κόμβο a είναι πράγωγο δέντρο, με ρίζα και μοναδικό φύλλο τον a



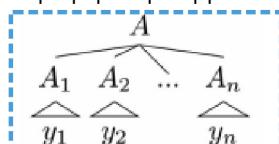
- Αν $A \rightarrow \epsilon$ είναι κανόνας του R με ϵ την κενή συμβολοσειρά, τότε το παρακάτω είναι παράγωγο δέντρο, με ρίζα τον A και μοναδικό φύλλο το ϵ



- Αν τα πρακάτω είναι παράγωγα δέντρα με ρίζες τα A_1, A_2, \dots, A_n και παραγόμενες συμβολοσειρές τις



και αν $A \rightarrow A_1 A_2 \dots A_n$ είναι κανόνας του R , τότε το παρακάτω είναι παράγωγο δέντρο με ρίζα A και παραγόμενη συμβολοσειρά την $y_1 \cdot y_2 \dots y_n$

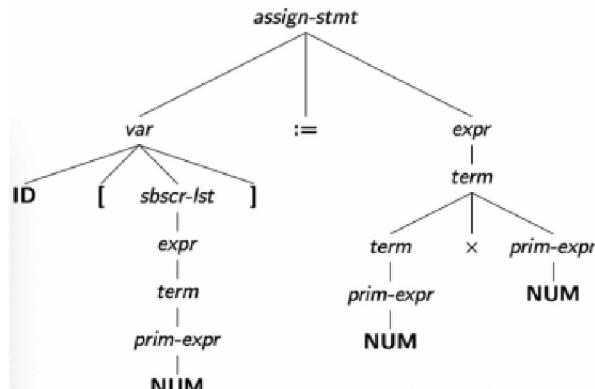


Παράδειγμα

Παράγωγα ή Συντακτικά δέντρα

$assign-stmt \rightarrow var := expr$	(1)
$expr \rightarrow term$	(2)
$expr + term$	(3)
$expr - term$	(4)
$term \rightarrow prim-expr$	(5)
$term \times prim-expr$	(6)
$term / prim-expr$	(7)
$prim-expr \rightarrow var$	(8)
NUM	(9)
$(expr)$	(10)
$var \rightarrow ID$	(11)
$ID [sbscr-lst]$	(12)
$sbscr-lst \rightarrow expr$	(13)
$sbscr-lst, expr$	(14)

- Παραγωγή του προγράμματος: $a [4] := 3 \times 5$
- $assign-stmt \xrightarrow{(1)} var := expr \xrightarrow{(12)} ID[sbscr-lst] := expr \xrightarrow{(13)} ID[expr] := expr$
 $\xrightarrow{(2)} ID[term] := expr \xrightarrow{(5)} ID[prim-expr] := expr \xrightarrow{(9)} ID[NUM] := expr$
 $\xrightarrow{(2)} ID[NUM] := term \xrightarrow{(6)} ID[NUM] := term \times prim-expr$
 $\xrightarrow{(5)} ID[NUM] := prim-expr \times prim-expr$
 $\xrightarrow{(9)} ID[NUM] := NUM \times prim-expr \xrightarrow{(9)} ID[NUM] := NUM \times NUM$

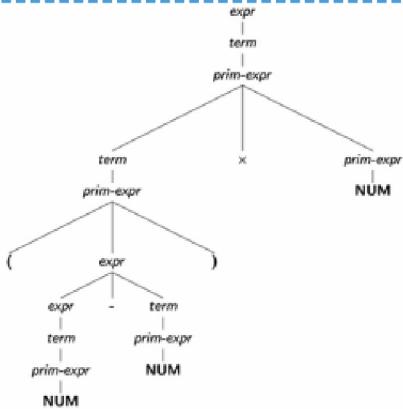


22

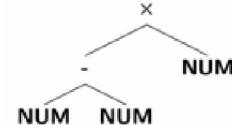
Αιφαρετικά Συντακτικά Δέντρα

- Τα παράγωγα δέντρα διατηρούν όλη την πληροφορία σχετικά με την παραγωγή μίας συμβολοσειράς από μία γραμματική
- Μέρος αυτής της πληροφορίας δεν επηρεάζει τη σημασία της γλ.πρ. και μπορεί να παραληφθεί για την απλούστευση του δέντρου
- Το **Αιφαρετικό συντακτικό δέντρο** δεν περιλαμβάνει όλες τις πληροφορίες

Παράδειγμα



π.χ. μπορούμε να παραλείψουμε τις παρενθέσεις στις εκφράσεις και οι πράξεις να ομαδοποιούνται απευθείας στη δομή του δέντρου



23

Διφορούμενες ή Ασαφείς Γραμματικές

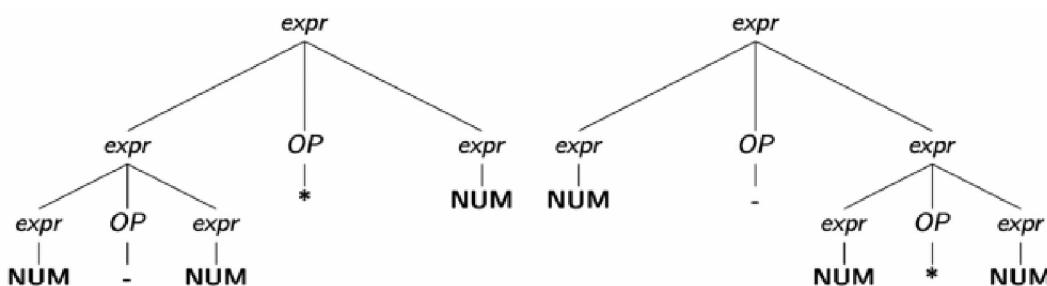
- Μία ΓΧΣ λέγεται **διφορούμενη ή ασαφής** όταν υπάρχει πρόταση της γλώσσας με τουλάχιστον δύο παράγωγα δέντρα (δύο διαφορετικές αριστερές ή δεξιές παραγωγές)
- Η ΓΧΣ που περιγράφει τη σύνταξη μίας γλώσσας δεν πρέπει να είναι διφορούμενη, ή αν είναι, το πρόβλημα πρέπει να αντιμετωπίζεται κατά τη μεταγλώττιση

Ασάφεια Προτεραιότητας Τελεστών

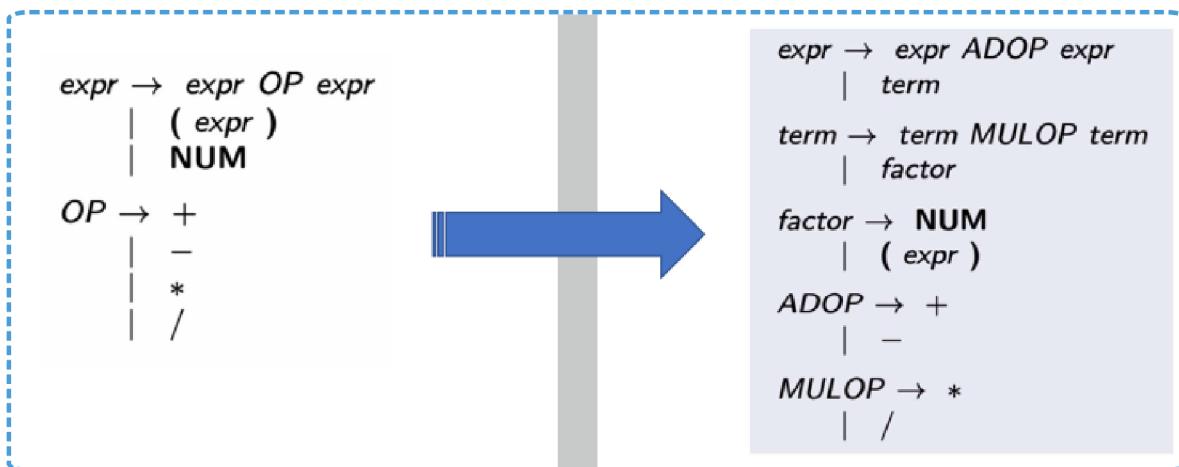
Παράδειγμα

$expr \rightarrow expr \text{ } OP \text{ } expr$	(1)
$(\text{ } expr \text{ })$	(2)
NUM	(3)
$OP \rightarrow +$	
$-$	(4)
$*$	(5)
$/$	(6)
	(7)

Δέντρα της ΓΧΣ για τις δύο παραγωγές της $27 - 5 * 8$ (η σωστή σημασία/αποτέλεσμα δίνεται στο δεξί δέντρο):



- Για να διορθώσουμε το πρόβλημα ασάφειας προτεραιότητας τελεστών:
 - Ωμαδοποιούμε τους τελεστές με την ίδια προτεραιότητα σε κανόνα με κοινό μη-τερματικό σύμβολο

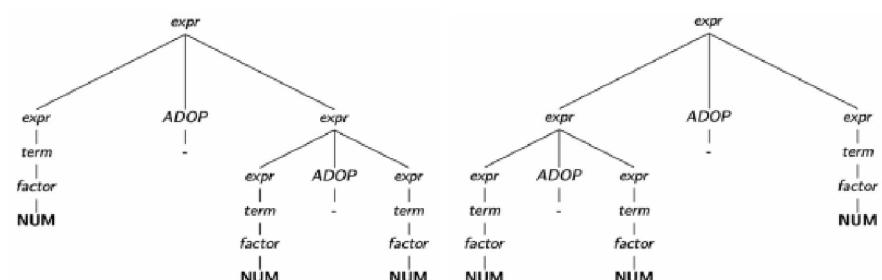
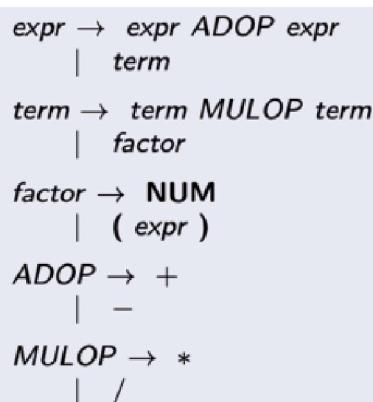


Ασάφεια Προσεταιριστικότητας Τελεστών

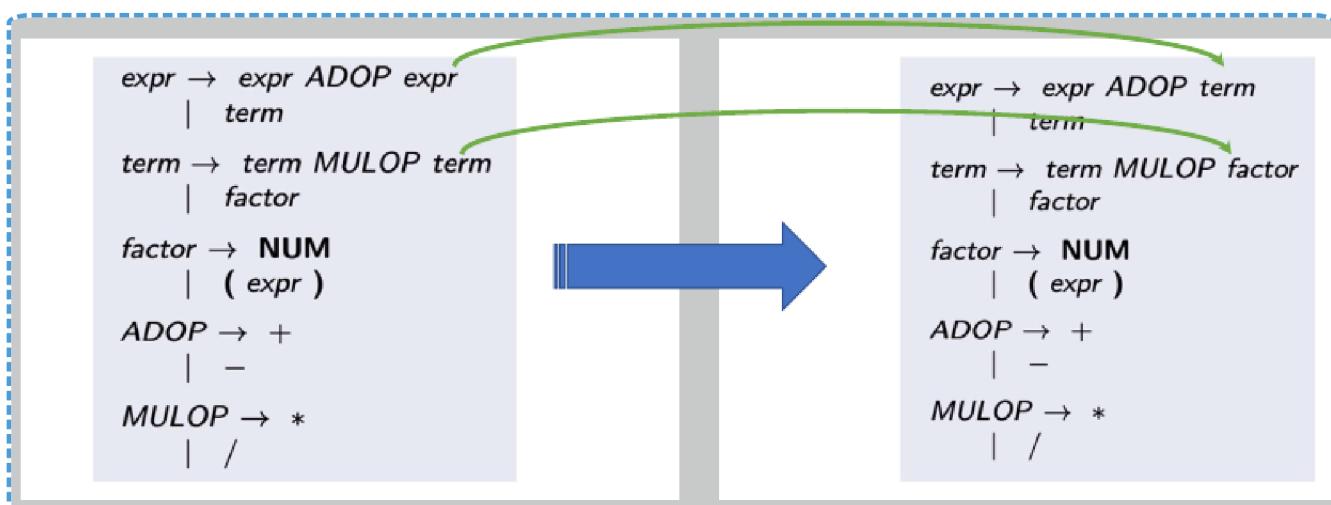
Παράδειγμα

- ΓΧΣ με πρόβλημα στον ορισμό της προσεταιριστικότητας των τελεστών

Δέντρα για δύο αριστερές παραγωγές της πρότασης 27-5-8 (η σωστή σημασία δίνεται από το δεξί δέντρο)



- Για να διορθώσουμε το πρόβλημα ασάφειας προσεταιριστικότητας τελεστών:
 - Ένας κανόνας ΓΧΣ παράγει δέντρο με αριστερή προσεταιριστικότητα αν αυτός είναι αποκλειστικά αριστερά αναδρομικός



Ασάφεια Μετέωρου Else

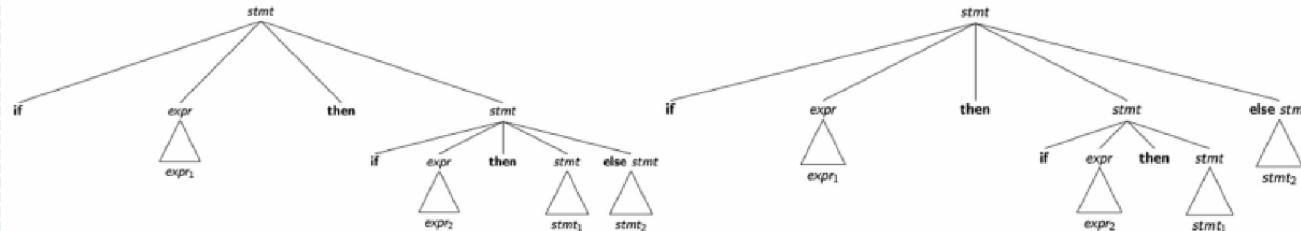
Παράδειγμα

- Διφορούμενη ΓΧΣ με το πρόβλημα του μετέωρου else

$$\begin{aligned}stmt \rightarrow & \text{ if } expr \text{ then } stmt \\& | \quad \text{if } expr \text{ then } stmt \text{ else } stmt\end{aligned}$$

Δύο δέντρα με διαφορετική σημασία για το πρόγραμμα

$$\text{if } expr_1 \text{ then if } expr_2 \text{ then } stmt_1 \text{ else } stmt_2$$



Διόρθωση ΓΧΣ που έχει το πρόβλημα ασάφειας μετέωρου else:

- Ισχύει – σε όλες τις γλ. προγραμματισμού – η **αρχή του πλησιέστερου if**
Αυτό σημαίνει ότι όταν δε γίνεται ένθεση εντολών if με χρήση διαχωριστών της γλώσσας, τότε κάθε else θα πρέπει να αναφέρεται στο πλησιέστερο προαναφερόμενο if.

$$\begin{aligned}stmt \rightarrow & \text{ if } expr \text{ then } stmt \\& | \quad \text{if } expr \text{ then } stmt \text{ else } stmt\end{aligned}$$



$$\begin{aligned}stmt \rightarrow & \text{ bal_stmt} \\& | \quad \text{unbal_stmt} \\bal_stmt \rightarrow & \text{ if } expr \text{ then } bal_stmt \text{ else } bal_stmt \\unbal_stmt \rightarrow & \text{ if } expr \text{ then } stmt \\& | \quad \text{if } expr \text{ then } bal_stmt \text{ else } unbal_stmt\end{aligned}$$

Κανονικές Εκφράσεις

Κανονικές Εκφράσεις (Ορισμός)

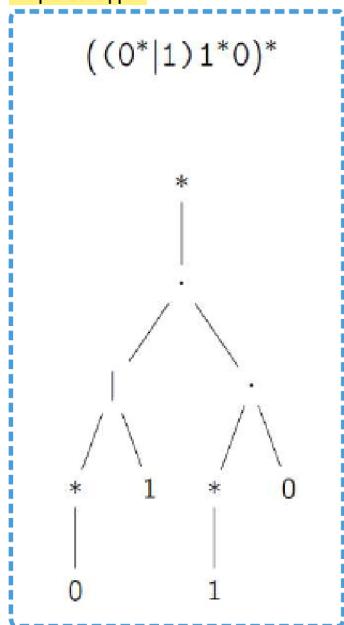
- Δοθέντος πεπερασμένου αλφαριθμητικού Σ , **Κανονικές Εκφράσεις** είναι οι σταθερές
 - \emptyset : το **κενό σύνολο**
 - ϵ : το σύνολο με την **κενή συμβολοσειρά**
 - a : το σύνολο με τη συμβολοσειρά a , όπου $a \in \Sigma$
- Δοθέντων των **Κανονικών Εκφράσεων** R και S , ορίζουμε επίσης τις ακόλουθες πράξεις ως K.E.
 - $R \cdot S = \{ab \mid a \in R, b \in S\}$: η **Παράθεση** της R με την S
 - $R|S = \{a \mid a \in R \cup S\}$: η **Διάζευξη** της R με την S
 - R^* : η **Kleene Star** της R , το ελάχιστο υπερσύνολο της R που περιλαμβάνει την ϵ και είναι **κλειστό προς την παράθεση**
- Η **Kleene Star** R^* εκφράζει το σύνολο όλων των συμβολοσειρών που σχηματίζονται με **παράθεση** πεπερασμένου αριθμού (ή μηδέν) συμβολοσειρών της R
 - πχ
 $\{\text{"ab"} \mid \text{"c"}\}^* = \{\epsilon, \text{"ab"}, \text{"c"}, \text{"abab"}, \text{"abc"}, \text{"cab"}, \text{"cc"}, \text{"ababab"}, \dots\}$
- Προτεραιότητα Τελεστών**: υποθέτουμε πως η προτεραιότητα πράξεων πάει ως εξής:
 1. Kleene Star
 2. Παράθεση
 3. Διάζευξη

K.E. -> Ελάχιστο ΝΠΑ

K.E. -> Συντακτικό Δέντρο

- Απλό

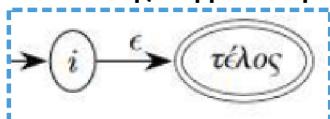
Παράδειγμα



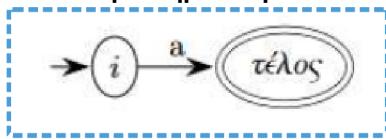
Συντακτικό Δέντρο -> ΜΝΠΑ

- Ανάπτυξη Thompson

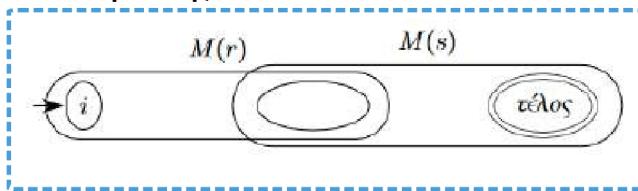
- ΜΝΠΑ Κενής Συμβολοσειράς ϵ



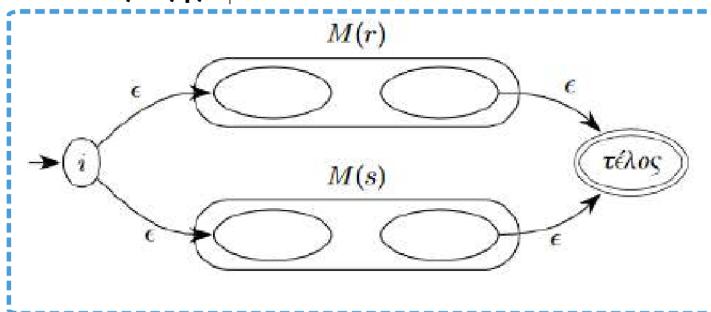
- ΜΝΠΑ Χαρακτήρα - Πρότυπο a



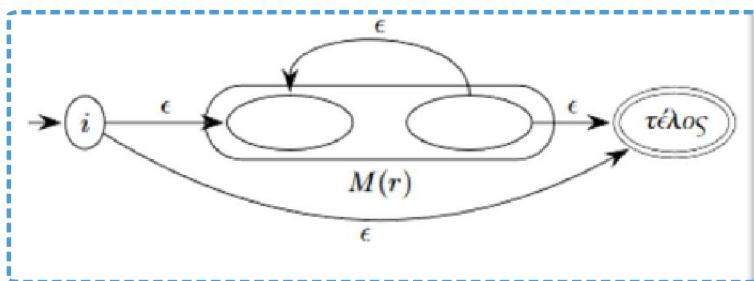
- ΜΝΠΑ Παράθεσης $r \cdot s$



- ΜΝΠΑ Διάζευξης $r | s$

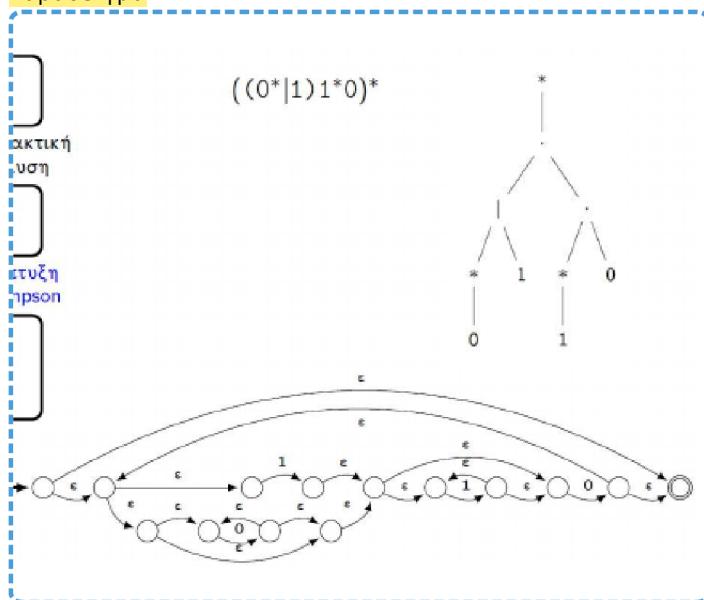


- ΜΝΠΑ Kleene Star r^*



- Συνδιάζοντας τα παραπάνω ΜΝΠΑ χτίζουμε το τελικό ΜΝΠΑ από το φύλλα πρός τη ρίζα του Συντακτικού Δέντρου

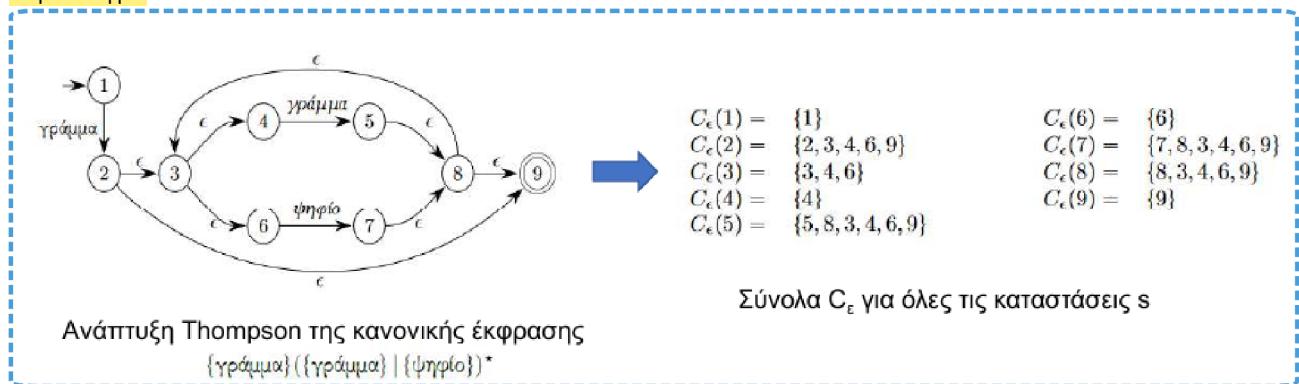
Παράδειγμα



ΜΝΠΑ -> ΝΠΑ

- Βρίσκουμε την **ε-Κλειστότητα** κάθε κατάστασης του ΜΝΠΑ
 - **ε-Κλειστότητα** της s : $C_\epsilon(s)$ το σύνολο όλων των καταστάσεων που μπορούν να προσεγγιστούν αποκλειστικά με ϵ -μεταβάσεις από την s

Παράδειγμα

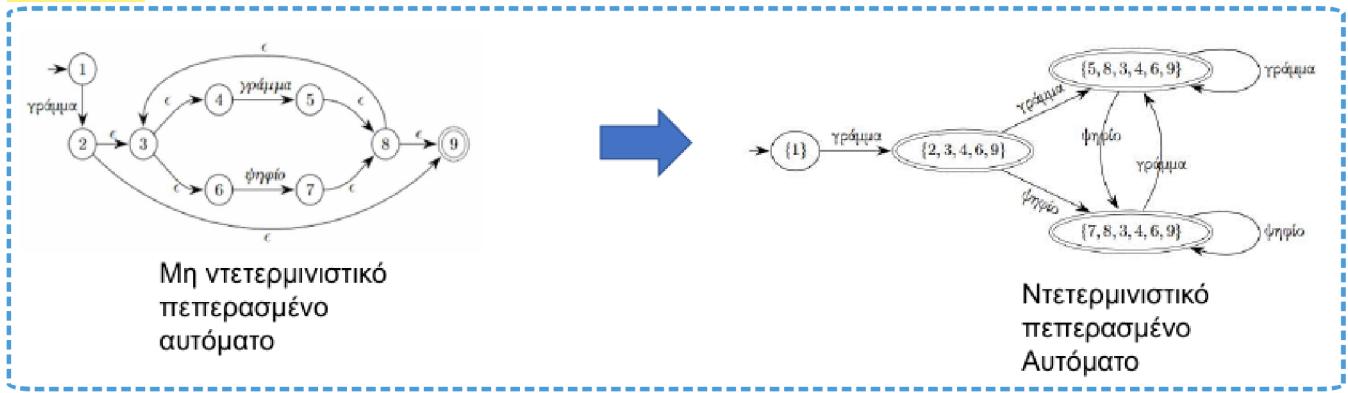


- Το νέο σύνολο καταστάσεων είναι η **ε-κλειστότητα** κάθε κατάστασης του αρχικού Αυτόματου, δηλαδή σύνολο από καταστάσεις
- Η νέα **Συνάρτηση Μετάβασης** ορίζεται ως

$$\Delta(R, s) = \bigcup_{r \in R} C_\epsilon(\delta(r, s))$$

- Για κάθε σύνολο από καταστάσεις R , η μετάβαση του μέσω κάποιου συμβόλου s , οδηγεί στην ένωση, από τις **ε-κλειστότητες**, των καταστάσεων στις οποίες οδηγούμασταν, από τις καταστάσεις του R
- Η **Αρχή** του ΝΠΑ είναι η **ε-κλειστότητα** της αρχής του ΜΝΠΑ
- Οι **τελικές καταστάσεις** του ΝΠΑ είναι όλες οι καταστάσεις που περιέχουν τελικές καταστάσεις του ΜΝΠΑ

Παράδειγμα

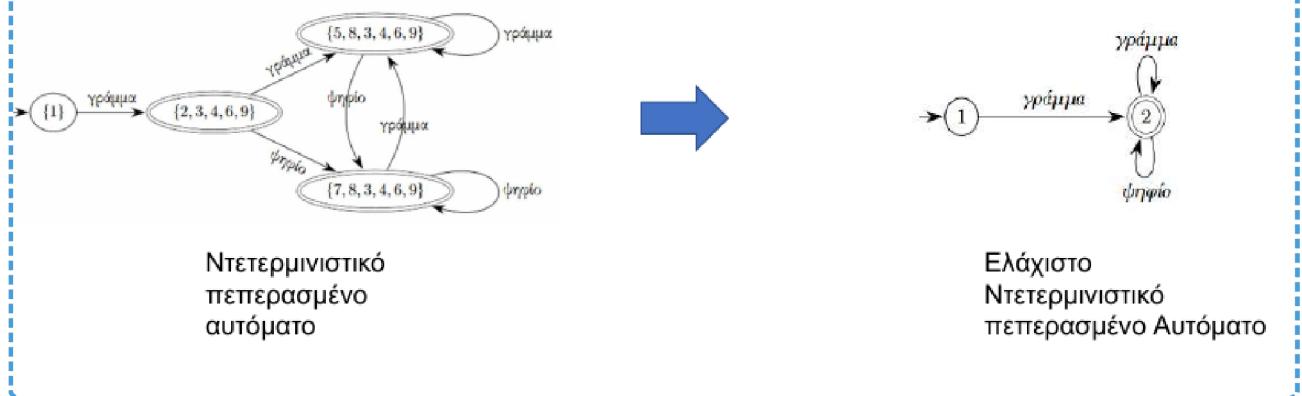


NPA -> Ελάχιστο NPA

- Χωρίζουμε τις καταστάσεις σε δύο κλασεις A, B , όπου
 - A οι μη-τελικές καταστάσεις του NPA
 - B οι τελικές καταστάσεις του NPA
- Αναδρομικά ελέγχουμε αν σε κάθε κλάση, όλες οι μεταβάσεις, οδηγούν στην ίδια κλαση
 - δλδ αν $\forall a \in A, \Delta(a, s) \in A \wedge \Delta(a, s) \in B, \forall s \in \Sigma$ η A είναι εντάξει
- Αν ένα υποσύνο μίας κλάσεις έχει διαφορετικά αποτελέσματα, δημιουργούμε νέα κλαση και επαναλαμβάνουμε τη διαδικασία, μέχρι κάθε κλαση να χαρακτηρίζεται από όμοιες μεταβάσεις σε άλλες κλάσεις
- Το Ελαχιστοποιημένο NPA έχει μία κατάσταση για κάθε κλαση του NPA, και η μετάβαση από κάθε κατάσταση είναι προφανής

Παράδειγμα

- Παράδειγμα για K.E. $\{γράμμα\} (\{γράμμα\} \mid \{\phiημίο\})^*$



Ορισμός Σύνταξης Γλ.Πρ.

- Για τον ορισμό της συντακτικής δομής των γλ.πρ., σρησιμοποιούνται **Γραμματικές Χωρίς Συμφρζόμενα**
 - Τερματικά σύμβολα είναι τα αναγνωρηστικά (γράφονται με **bold**)
 - Μη τερματικά σύμβολα εκφράζουν τις συντακτικές δομές της γλώσσας (assig-stmt, expr, term...)
 - (γράφονται με *italics*)
 - Η Αρχή της ΓΧΣ αντιστοιχεί σε ένα οποιοδήποτε πρόγραμμα (σλυμβολο στο αριστερό μέρος του πρώτου κανόνα)

Παράδειγμα

$assign-stmt \rightarrow var := expr$	(1)
$expr \rightarrow term$	(2)
$expr + term$	(3)
$expr - term$	(4)
$term \rightarrow prim Expr$	(5)
$term \times prim Expr$	(6)
$term / prim Expr$	(7)
$prim Expr \rightarrow var$	(8)
NUM	(9)
$(expr)$	(10)
$var \rightarrow ID$	(11)
$ID [sbscr-lst]$	(12)
$sbscr-lst \rightarrow expr$	(13)
$sbscr-lst, expr$	(14)

Κατηγοριοποίηση Γραμματικών

Το σύνολο FIRST

- Αν \bar{u} είναι συμβολοσειρά που αποτελείται από μη-τερματικά ή/και τερματικά σύμβολα μίας γραμματικής, τότε

$$FIRST(\bar{u}) = \{t | \bar{u} \Rightarrow^* t\bar{o}\}$$

το σύνολο των t (τερματικών συμβόλων ή ϵ) που μπορεί να εμφανίζονται στην αρχή όλων των συμβολοσειρών, που παράγονται από τη \bar{u} μέσω κανόνων

Αλγόριθμος Υπολογισμού FIRST

- Είσοδος: Γραμματική $G = (N, T, S, P)$ και μία προτασιακή μορφή \bar{u}
- Έξοδος: το σύνολο $FIRST(\bar{u})$

```

1: if ( $\bar{u} = x_1 \in N \cup T \cup \{\epsilon\}$ ) then
2:   για  $x_1 \in T$ , έχουμε  $FIRST(\bar{u}) = \{x_1\}$ 
3:   για  $x_1 = \epsilon$ , έχουμε  $FIRST(\bar{u}) = \{\epsilon\}$ 
4:   για  $x_1 = X \in N$ , με  $X \rightarrow \bar{u}_1 | \bar{u}_2 | \dots | \bar{u}_k$ 
      έχουμε  $FIRST(\bar{u}) = FIRST(\bar{u}_1) \cup FIRST(\bar{u}_2) \cup \dots \cup FIRST(\bar{u}_k)$ 
5: if ( $\bar{u} = x_1 x_2 \dots x_n$ , με  $x_i \in N \cup T$ ) then
6:    $FIRST(\bar{u}) = \{\}; j=0;$ 
7:   repeat
8:      $j=j+1;$ 
9:     προσθέτουμε στο  $FIRST(\bar{u})$  το  $FIRST(x_j) - \{\epsilon\}$ ;
10:    until  $x_j$  δεν είναι απαλείψιμο (δεν παράγει το  $\epsilon$ ) ή  $j = n$ 
11:    if (η  $x_1 x_2 \dots x_n$  είναι απαλείψιμη) then
12:      προσθέτουμε το  $\epsilon$  στο  $FIRST(\bar{u})$ 

```

- Αν το \bar{u} είναι τερματικό ή ϵ , το $FIRST(\bar{u}) = \{\bar{u}\}$
- Αν το \bar{u} είναι μη-τερματικό, το $FIRST(\bar{u})$ είναι η ένωση των $FIRST$ όλων των συμβολοσειρών που μπορούν να παραχθούν με έναν κανόνα από το \bar{u}
- Αν το \bar{u} είναι σύζευξη από τερματικά ή/και μη τερματικά $x_1 x_2 \dots x_n$
 - Παίρνουμε με τη σειρά κάθε επιμέρους σύμβολο x_j και προσθέτουμε στο $FIRST(\bar{u})$, το $FIRST(x_j) - \{\epsilon\}$
 - Μέχρι να βρούμε x_j που να μην παράγει το ϵ , ή να τελειώσουν τα σύμβολα
 - Αν η \bar{u} παράγει το ϵ , το προσθέτουμε στη $FIRST(\bar{u})$

Το Σύνολο FOLLOW

- Αν X ένα μη-τερματικό σύμβολο, τότε $FOLLOW(X)$ το σύνολο των τερματικών συμβόλων που μπορεί να εμφανιστούν αμέσως μετά από αυτό, σε όλες τις περιπτώσεις παραγώμενων συμβόλοσειρών

Αλγόριθμος Υπολογισμού

- Είσοδος: Γραμματική $G = (N, T, S, P)$ και ένα μη-τερματικό X
- Έξοδος: το σύνολο $FOLLOW(X)$

```

1: if ( $X = S$ ) then
2:   $ ∈ FOLLOW( $X$ ), όπου $ συμβολίζει το τέλος συμβόλοσειράς
3: for all  $p \in P$ , κανόνας  $Y \rightarrow \bar{m}X\bar{o}$ , δηλ. με  $X$  στο δεξί του μέρος do
4:   if ( $\bar{o}$  αρχίζει από  $c \in T$ ) then
5:      $c \in FOLLOW(X)$ 
6:   if ( $\bar{o}$  αρχίζει από  $Z \in N$ ) then
7:     προσθέτουμε στο  $FOLLOW(X)$  το  $FIRST(\bar{o}) - \{\epsilon\}$ ;
8:   if ( $\bar{o} = \epsilon$  ή  $\bar{o} \Rightarrow^* \epsilon$ ) then
9:     προσθέτουμε στο  $FOLLOW(X)$  το  $FOLLOW(Y)$ ;
```

- Αν X είναι η αρχή της γραμματικής, τότε βάζουμε το \$ στο $FOLLOW(X)$
- Για κάθε κανόνα της γραμματικής, μορφής $Y \rightarrow \bar{m}X\bar{o}$ (με το X στο δεξί του μέρος)
 - αν το \bar{o} αρχίζει με τερματικό, βάλε αυτό το τερματικό στο $FOLLOW(X)$
 - αν το \bar{o} αρχίζει με μη-τερματικό, βάλε το $FIRST(\bar{o}) - \{\epsilon\}$ στο $FOLLOW(X)$
 - αν το \bar{o} έχει ή παράγει ϵ , βάλε το $FOLLOW(Y)$ στο $FOLLOW(X)$

Παράδειγμα

Σύνολα FIRST και FOLLOW

E	$\rightarrow TT_r$	$FIRST(E) = \{‘(, “αριθμός”\}$
T_r	$\rightarrow ‘+’TT_r$	$FIRST(T) = \{‘(, “αριθμός”\}$
	$ ‘-’TT_r$	$FIRST(F) = \{‘(, “αριθμός”\}$
	$ \epsilon$	$FIRST(T_r) = \{‘+’, ‘-’, \epsilon\}$
T	$\rightarrow FF_r$	$FIRST(F_r) = \{‘*’, ‘/’, \epsilon\}$
F_r	$\rightarrow ‘*’FF_r$	Υποδεικνύει το τέλος του προγράμματος.
	$ ‘/’FF_r$	$FOLLOW(E) = \{$, ‘)’\}$
	$ \epsilon$	$FOLLOW(T) = \{‘+’, ‘-’, $, ‘)’\}$
F	$\rightarrow ‘(’E‘)’$	$FOLLOW(F) = \{‘*’, ‘/’, $, ‘)’, ‘+’, ‘-’, $, ‘)’\}$
	$ “αριθμός”$	$FOLLOW(F_r) = \{‘+’, ‘-’, $, ‘)’\}$

19

Γραμματικές LL(1)

- Μία γραμματική είναι $LL(1)$ αν-ν
 - Για κάθε κανόνα $X \rightarrow \bar{u}_1|\bar{u}_2|\dots|\bar{u}_n$ ισχύει

$$FIRST(\bar{u}_i) \cap FIRST(\bar{u}_j) = \emptyset$$

για όλα τα $i, j = 1\dots n$ με $i \neq j$

- Για κάθε μη-τερματικό X , τέτοιο ώστε $\epsilon \in FIRST(X)$ ισχύει

$$FIRST(X) \cap FOLLOW(X) = \emptyset$$

Απομάκρυνση Αριστερής Αναδρομικότητας

Άμεση Αριστερή Αναδρομικότητα

- Πρόβλημα με κανόνα παραγωγής του τύπου

$$X \rightarrow X\bar{u}_1|X\bar{u}_2|\dots|X\bar{u}_n|\bar{o}_1|\bar{o}_2|\dots|\bar{o}_m$$

- \bar{u}_i : συμβολοσειρές από τερματικά ή/και μη-τερματικά
- \bar{o}_j : συμβολοσειρές από τερματικά ή/και μη-τερματικά που δεν ξεκινούν από X
- Η λύση δίνεται με την μετατροπή της αριστερής αναδρομής σε δεξιά ως εξής:

$$X \rightarrow \bar{o}_1 X' |\bar{o}_2 X' | \dots | \bar{o}_m X'$$

$$X' \rightarrow \bar{u}_1 X' |\bar{u}_2 X' | \dots | \bar{u}_n X' | \varepsilon$$

- όπου X' ένα νέο μη-τερματικό σύμβολο
- Η παραπάνω αντικατάσταση δεν επηρεάζει τη γλώσσα της γραμματικής

Έμμεση Αριστερή Αναδρομικότητα

- Πρόβλημα με κανόνες παραγωγής του τύπου

$$X \rightarrow Y\bar{u}_2| \dots$$

$$Y \rightarrow X\bar{u}_1| \dots$$

- Για να εφαρμοστεί ο αναγκαίος μετασχηματισμός, προϋποθέτει η γραμματική
 - Να μην περιέχει κυκλικούς κανόνες παραγωγής
 - πχ $M \rightarrow Q, Q \rightarrow R, R \rightarrow M$
 - Να μην περιέχει κανόνες-ε
 - Σύμφωνα με θεώρημα, όταν στη γραμματική υπάρχουν κανόνες-ε, είναι γενικά δυνατή η απομάκρυνσή τους με διατύπωση ισοδύναμης γραμματικής

Είσοδος: γραμματική G χωρίς κυκλικούς κανόνες παραγωγής και κανόνες-ε

Έξοδος: ισοδύναμη γραμματική με την G χωρίς αριστερή αναδρομή

- έστω τα μη τερματικά X_1, X_2, \dots, X_n της G με δείκτες που αντιστοιχούν στη σειρά εμφάνισης του κανόνα αντικατάστασης τους στη γραμματική
- for $i:=1$ to n do
- for $j:=1$ to $i-1$ do
- αντικατέστησε κάθε κανόνα της μορφής

$$X_i \rightarrow X_j \bar{o}$$
 με τους κανόνες

$$X_i \rightarrow \bar{u}_1 \bar{o} \mid \bar{u}_2 \bar{o} \mid \dots \mid \bar{u}_k \bar{o}$$
 για όλους τους κανόνες της μορφής

$$X_j \rightarrow \bar{u}_1 \mid \bar{u}_2 \mid \dots \mid \bar{u}_k$$
- απομόρφυνε την άμεση αριστερή αναδρομή στους κανόνες για το X_i ;

- Πάρε **ΜΕ ΤΗ ΣΕΙΡΑ** τα μη-τερματικά και
 - Έλεγξε αν έχουν κανόνες, που το δεξιά τους μέρος, ξεκινά με μη-τερματικό, που έχει ήδη ελεγχθεί
 - Αντικατέστησε κάθε κανόνα της μορφής

$$X_i \rightarrow X_j \bar{o}$$

με τους κανόνες

$$X_i \rightarrow \bar{u}_1 \bar{o} \mid \bar{u}_2 \bar{o} \mid \dots \mid \bar{u}_k \bar{o}$$

για κάθε κανόνα (του προηγούμενου μη-τερματικού) της μορφής

$$X_j \rightarrow \bar{u}_1 \mid \bar{u}_2 \mid \dots \mid \bar{u}_k$$

- Απομάκρυνε την άμεση αριστερή αναδρομή για τους κανόνες του X_i

Παράδειγμα

Απομάκρυνση έμμεσης και άμεσης αριστερής αναδρομής

$$X \rightarrow Y "p" \mid X "p" \mid "r"$$

$$Y \rightarrow Y "q" \mid X "q" \mid "s"$$



Απομάκρυνση άμεσης αριστερής αναδρομής κανόνα $X \rightarrow Y "p"$



Απομάκρυνση άμεσης αριστερής αναδρομής κανόνα $Y \rightarrow Y "q"$ και $Y \rightarrow Y "p" X' "q"$

$$X \rightarrow Y "p" X' \mid "r" X'$$

$$X' \rightarrow "p" X' \mid \epsilon$$

$$Y \rightarrow Y "q" \mid X "q" \mid "s"$$

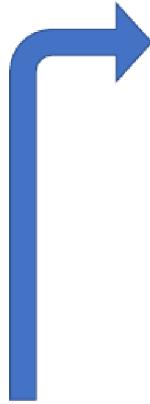


Απομάκρυνση έμμεσης αριστερής αναδρομής κανόνα $Y \rightarrow X "q"$

$$X \rightarrow Y "p" X' \mid "r" X'$$

$$X' \rightarrow "p" X' \mid \epsilon$$

$$Y \rightarrow Y "q" \mid Y "p" X' "q" \mid "r" X' "q" \mid "s"$$



$$X \rightarrow Y "p" X' \mid "r" X'$$

$$X' \rightarrow "p" X' \mid \epsilon$$

$$Y \rightarrow "r" X' "q" Y' \mid "s" Y'$$

$$Y' \rightarrow "q" Y' \mid "p" X' "q" Y' \mid \epsilon$$

7

Μετασχηματισμός Γραμματικής σε LL(1) - Αριστερή Παραγοντοποίηση

- Υπάρχουν γραμματικές που εμφανίζουν σε έναν τουλάχιστον κανόνα της γραμματικής, δύο ή περισσότερες εναλλακτικές με το ίδιο πρόθεμα. Για να εκτελέσουμε προβλέπουσα ανάλυση, αυτοί οι κανόνες πρέπει να εξαλειφθούν

• Αριστερή Παραγοντοποίηση Παράδειγμα

- H

$$\begin{aligned} C &\rightarrow 'if' E 'then' C \\ &\mid 'if' E 'then' C 'else' C \\ &\mid 'άλλο' \end{aligned}$$

- Γίνεται

$$\begin{aligned} C &\rightarrow 'if' E 'then' C C' \\ &\mid 'άλλο' \\ C' &\rightarrow 'else' C \\ &\mid \epsilon \end{aligned}$$

- δλδ Βρίσκουμε τα δεξιά μέρη με το κοινό πρόθεμα και τα αποσπούμε, αφήνοντας μόνο το κοινό πρόθεμα στον αρχικό κανόνα, και δημιουργόντας νέο μη-τερματικό για κάθε διαφορετικό επίθεμα, το οποίο μπορεί να γίνει είτε το επίθεμα, είτε ϵ

Γραμματικές LR(1)

Για να είναι μία γραμματική $LR(1)$ πρέπει:

- Αν υπάρχει στοιχείο της μορφής $[Y \rightarrow \bar{u} \cdot k\bar{o}, a] \in I_i$ στο σύνολο I_i για κάποιο τερματικό k , τότε δεν θα υπάρχει στο ίδιο σύνολο το συμπληρωμένο στοιχείο $[X \rightarrow \bar{e} \cdot, k] \in I_i$ (shift-reduce conflict)
- Ένα σύνολο στοιχείων I_i δεν μπορεί να περιέχει πανω από ένα συμπληρωμένα στοιχεία της μορφής $[X \rightarrow \bar{e} \cdot, a]$ και $[Z \rightarrow \bar{w} \cdot, a]$ (reduce-reduce conflict)

Γραμματικές SLR(1)

- Οι γραμματικές $SLR(1)$ είναι και (κανονική) $LR(1)$ - $SLR(1) \subset LR(1)$, αλλά η ανάλυση $LR(1)$ μπορεί να έχει περισσότερες καταστάσεις από την $SLR(1)$
- Οι γραμματικές $LR(1)$ δεν είναι απαραίτητα και $SLR(1)$

Αλγόριθμοι Συντακτικής Ανάλυσης

- **Καθοδική Συντακτική Ανάλυση**
 - *Rīza* (αρχή γραμματικής) προς φύλλα (αναγνωριστικά τερματικών)
 - Προϋποθέτει μη-αριστερά αναδρομική γραμματική
 - Αντιστοιχεί σε αριστερές παραγωγές της γραμματικής
- **Ανοδική Συντακτική Ανάλυση**
 - Φύλλα (αναγνωριστικά τερματικών) προς φύλλα (αρχή γραμματικής)
 - Εφαρμόζεται σε ευρύτερη οικογένεια γραμματικών
 - Αντιστοιχεί σε δεξιές παραγωγές της γραμματικής
- **Αλγόριθμοι Οπισθοδρόμησης**
 - Επιλεγεται εξ αρχής σειρά εφάρμοσης κανόνων γραμματικής και αν η ανάλυση φτάσει σε σημείο που δεν μπορεί να προχωρήσει
 - Αναιρείται ένας ή περισσότεροι κανόνες και εφαρμόζονται διαφορετικοί για τα ίδια σύμβολα
 - Υψηλό υπολογιστικό κόστος
 - Καθυστερημένη ανίχνευση λαθών
 - **DFS εφαρμογή κανόνων γραμματικής**
- **Αλγόριθμοι Πρόγνωσης**
 - Διαβάζονται ένα ή περισσότερα τερματικά σύμβολα και
 - με βάση αυτά αποφασίζεται ποιούς κανόνας θα χρησιμοποιηθεί, από αυτούς που αναφέρονται σε ένα συγκεκριμένο μη-τερματικό
 - Συγκριτικά αποδοτικότερη μεταγλώττιση και πιο απλή ανάκαμψη από λάθη

Ονοματολογία Αλγορίθμων Συντακτικής Ανάλυσης

Ένας αλγόριθμος ΣΑ παίρνει όνομα της μορφής

$$XY(n)$$

- Όπου
 - X αναφέρεται στη φορά ανάγνωσης των συμβόλων, που αντιστοιχούν στα αναγνωριστικά συμβολοσειρών του πηγαίου προγράμματος
 - $X = L$ δηλώνει πως η είσοδος αναγνωριστικών στον αλγόριθμο γίνεται από τα αριστερά προς τα δεξιά
 - Y αναφέρεται στον τύπο παραγωών που ακολουθεί ο αλγόριθμος
 - $Y = L$ δηλώνει αριστερές παραγωγές (χαρακτηριστικό της καθοδικής ανάλυσης)
 - $Y = R$ δηλώνει δεξιές παραγωγές (χαρακτηριστικό της ανοδικής ανάλυσης)
 - n αναφέρεται στον αριθμό τερματικών συμβόλων που διαβάζονται για να αποφασιστεί ο κανόνας της γραμματικής που θα χρησιμοποιηθεί

Καθοδική ΣΑ

Καθοδική ΣΑ με Οπισθοδρόμηση

1. Για ένα τερματικό σύμβολο, εφαρμόζεται ο πρώτος κανόνας που αναφέρεται σε αυτό
2. Στη συμβολοσειρά που προκύπτει επιλέγεται το πρώτο από τα αριστερά μη τερματικό και εφαρμόζεται ο πρώτος κανόνας που αναφέρεται σε αυτό
3. Επαναλαμβάνουμε το βήμα 2, για κάθε μη τερματικό σύμβολο που ακολουθεί μέχρι να παραχθεί
 - μία σειρά που αντιστοιχεί στη συμβολοσειρά εισόδου
 - μία σειρά τερματικών που διαφέρει από το αντίστοιχο τμήμα της εισόδου

Όταν συμβεί αυτό αναιρείται ο τελευταίος κανόνας που εφαρμόστηκε και επιλέγεται ο επόμενος για το αντίστοιχο μη τερματικό

Αν εξαντληθούν οι κανόνες για το συγκεκριμένο μη τερματικό, αναιρείται ο επόμενος πιο πρόσφατος κανόνας

Οι αναιρέσεις συνεχίζουν μέχρι να μπορεί να συνεχίσει η ανάλυση, ή να επιστρέψουμε στην αρχή της γραμματικής, που σημαίνει πως βρέθηκε συντακτικό λάθος

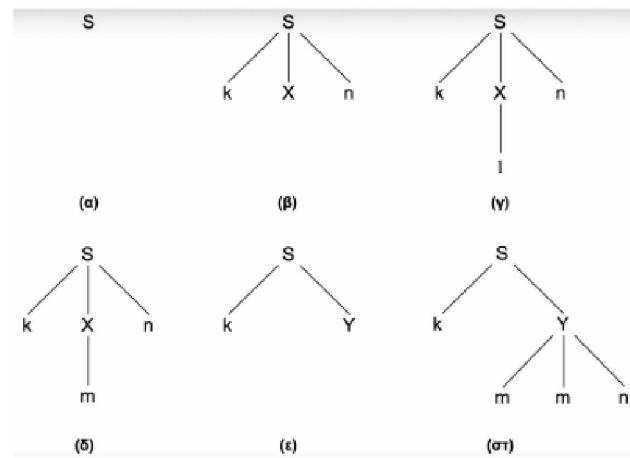
- Στην ουσία είναι **DFS** αναζήτηση για τη σωστή παραγωγή μίας συμβολοσειράς

Παράδειγμα

Καθοδική ΣΑ με οπισθοδρόμηση

- Παραγωγή του προγράμματος: **kmmn**
- $S \xrightarrow{(\beta)} kXn \xrightarrow{(\gamma)} kln$ αναίρεση τελευταίου κανόνα
- $S \xrightarrow{(\beta)} kXn \xrightarrow{(\delta)} kmn$ αναίρεση προτελευταίου κανόνα
- $S \xrightarrow{(\varepsilon)} kY \xrightarrow{(\sigma\tau)} kmmn$

S	\rightarrow	"k"	X	"n"
		"k"	Y	
X	\rightarrow	"l"		"m"
Y	\rightarrow	"mmn"		
		"nnm"		



- Εξαιρετικά αργός
- Ανίχνευση λάθους όταν έχει δοκιμαστεί κάθε πιθανή παραγωγή
- Τα περισσότερα (αν όχι όλα) χαρακτηριστικά που μπορεί να είναι απιθυμητά σε μία γλ.πρ., μπορούν να περιγραφούν από γραμματικές με απαιτήσεις περιορισμένης ή ανύπαρκτης οπισθοδρόμησης

Καθοδική ΣΑ Προβλεπουσας Αναδρομικής Κατάβασης

- Κάθε κανόνας της γραμματικής, για ένα μη τερματικό σύμβολο, εκφράζεται από μία **διαδικασία** (*function*)
 - Το **αριστερό** μέρος είναι το **όνομα**, το **δεξί** καθορίζει τον **κώδικα**
- Μία σειρά συμβόλων εκφράζεται από την κατά σειρά κλήση των αντίστοιχων συναρτήσεων
- Η ΣΑ προβλεπουσας αναδρομικής κατάβασης αποτελείται από
 - Μία **καθολική μεταβλητή**, με την τιμή του **αναγνωριστικού** που διαβάστηκε πιο πρόσφατα
 - Μία **διαδικασία** που ελέγχει αν το τρέχον αναγνωριστικό είναι ίδιο με κάποιο αναμενόμενο, και αν είναι καλεί τη διαδικασία ανάγνωσης του επόμενου αναγνωριστικού, που ενημερώνει την καθολική μεταβλητή
 - Τις διαδικασίες ανάλυσης για μη τερματικά σύμβολα της γραμματικής
 - Την κύρια συνάρτηση, που διαβάζει το πρώτο αναγνωριστικό, και έπειτα καλεί τη διαδικασία ανάλυσης για το μη τερματικό της αρχής της γραμματικής

Παράδειγμα

Έστω η ΓΧΣ:

$$E \rightarrow TT_r$$

$$T_r \rightarrow ' + ' TT_r$$

$$| ' - ' TT_r$$

$$|\varepsilon$$

$$T \rightarrow FF_r$$

$$F_r \rightarrow ' * ' FF_r$$

$$| ' \backslash ' FF_r$$

$$|\varepsilon$$

$$F \rightarrow ' (' E ') '$$

$$| ' \text{αριθμός} '$$

Αυτό μεταφράζεται σε κώδικα

```

// Ena gia kathe typo anagnwristikou
typedef enum {PLUS,MINUS,MULT,DIV,LPAR,RPAR,NUMBER,END_FILE} TokenType;
// H katholiki metavlth epomenou anagnoristikou
TokenType token;

void error() {
    printf("Syntax Error \n");
    exit(0);
}

//Synarthsh elegxou
void match(TokenType expexted_token) {
    if (token == expected_token)
        token = getToken(); //Lexikos analyths
    else
        error();
}

//Synarthseis analyshs mh termatikwn
void E();

void F() { // F →
    switch(token){
        case LPAR: // '('
            token = getToken();
            E();
            match(RPAR);
            break;
        case NUMBER: // 'arithmos'
            token = getToken();
            break;
        default:
            error();
    }
}

void Fr() { //Fr →
    switch(token){
        case MULT: // '*'FFr
            token = getToken();
            F();
            Fr();
            break;
        case DIV: // '/'FFr
            token = getToken();
            F();
            Fr();
            break;
        default: // ε
            return;
    }
}

void T() { //T → FFr
    F();
    Fr();
}

```

```

void Tr() { //Tr →
    switch(token) {
        case PLUS: // '+' TTr
            token = getToken();
            T();
            Tr();
        case MINUS: // '-' TTr
            token = getToken();
            T();
            Tr();
        default: // ε
            return;
    }
}

void E() { //E → TTr
    T();
    Tr();
}

void main() {
    token = getToken();
    E();
    if (Token != END_FILE)
        return error();
}

```

- Όπως όλοι οι αλγόριθμοι καθοδικής ανάλυσης, δεν εφαρμόζεται σε **αριστερά αναδρομικές γραμματικές**
 - χρειάζεται μετασχηματισμός της γραμματικής σε **μη-αριστερά αναδρομική**
- Η προβλεπουσα αναδρομική κατάβαση πρέπει να επιλέγει σε κάθε βήμα τον σωστό κανόνα μεταξύ των πιθανών επιριπτώσεων (έτσι ώστε να αποφέυγεται η οπισθοδρόμηση)
 - η επιλογή του κανόνα γίνεται με βάση το τερματικό σύμβολο που βρίσκεται σε κάθε θέση
- Απαραίτητη προϋπόθεση για να εφαρμοστεί, είναι η γραμματική να είναι $LL(1)$
- Αν δεν είναι τότε πρέπει να μετασχηματιστέι σε $LL(1)$
- Η γενική μορφή μιας συνάρτησης ανάλυσης είναι η εξής

Αν για το μη τερματικό σύμβολο X ορίζεται στη γραμματική ο χανόνας

$$X \rightarrow \bar{u}_1 | \bar{u}_2 | \dots | \bar{u}_k$$

και αν υποθέσουμε ότι η \bar{u}_i είναι απολεύψημη, τότε η διαδικασία που αντιστοιχεί στο μη τερματικό σύμβολο X έχει την μορφή:

- 1: $X()$:
- 2: **switch** (token)
- 3: **case** $c \in FIRST(\bar{u}_1)$:
- 4: κλήση διαδικασιών για τα σύμβολα του \bar{u}_1
- 5: **case** $c \in FIRST(\bar{u}_2)$:
- 6: κλήση διαδικασιών για τα σύμβολα του \bar{u}_2
-
- 7: **case** $c \in FIRST(\bar{u}_i) \cup FOLLOW(X)$:
- 8: κλήση διαδικασιών για τα σύμβολα του \bar{u}_i ή (αντίστοιχα) των διαδικασιών για τα σύμβολα που ακολουθούν μετά το X
- 9: **default**:
- 10: λάθος;
- 11: **end switch**
- 12: Τέλος X.

- Σε γενικούς όρους δουλεύει ως εξής
 - Όταν εκτελείται η διαδικασία, για ένα μη-τερματικό X , ελέγχεται αν το τρέχον αναγνωριστικό εμφανίζεται σε ένα $FIRST$ για το δεξί μέρος κάποιου κανόνα του X
 - Αν εμφανίζεται, τότε εφαρμόζεται αυτός ο κανόνας
 - Διαφορετικά υπάρχει συντακτικό λάθος, εκτός και αν περιέχεται παραλέιψη συμβολοσειρά στο δεξί μέρος του X , οπότε ελέγχεται αν το αναγνωριστικό βρίσκεται στο $FOLLOW(X)$
- Στην προβλέπουσα αναδρομική κατάβαση, τα συντακτικά λάθη εντοπίζονται άμεσα, και αυτό διευκολύνει την ανάκαμψη, που είναι πιο κατάλληλη για κάθε περίπτωση
 - Όμως η απαίτηση να μην είναι αριστερά αναδρομική η γραμματική, αποκλειεί την επιθυμητή ιδιότητα της αριστερής προσεταιριστικότητας
 - Το πρόβλημα αντιμετωπίζεται με προσεκτική σχεδίαση της αλληλεπίδρασης συναρτήσεων, που αντιστοιχούν στα μη τερματικά σύμβολα της γραμματικής

Καθοδική ΣΑ Προβλεπουσας Αναδρομικής Κατάβασης με Ανάκαμψη σε Κατάσταση "Πανικού"

- Μετά από εντοπισμό λάθους, "αγνοούνται" λεξικές μονάδες μέχρι να βρεθεί μία από την οποία μπορεί να συνεχίσει με ασφάλεια η ανάλυση (**λεξικές μονάδες συγχρονισμού**)
- Πρέπει να επιλεγούν οι κατάλληλες λεξικές μονάδες συγχρονισμού για κάθε πιθανή κατάσταση από την οποία μπορεί να περάσει η ανάλυση
 - Υποψήφιες είναι οι λεξικές μονάδες του συνόλου $FOLLOW$ του μη-τερματικού, στο δεξί μέρος ενός κανόνα παραγωγής
- Ιδιέταιρα χρήσιμα είναι και τα σύνολα $FIRST$, που λαμβάνονται υπόψη για να αποτρέπεται η προσπέραση λεξικών μονάδων με καίρια σημασία, εξαιτίας πιθανής απουσίας του τερματικού συμβόλου που αναμένεται (όπως το άνοιγμα μίας έφρασης, όταν λείπει το ";")

Παράδειγμα

Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη σε κατάσταση «πανικού»

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#define MAXTOKENLEN 40
typedef enum {
    PLUS, MINUS,
    MULT, DIV,
    LPAR, RPAR,
    NUMBER, END_FILE
} TokenType;

TokenType token;
char tokenString[MAXTOKENLEN+1];
struct stoken_set {
    TokenType stoken;
    struct stoken_set *next;
    struct stoken_set *prev;
};
typedef struct stoken_set *STOKEN;
```

H ignore προσπερνά έναν αριθμό λεξικών μονάδων μέχρι να διαβαστεί μια μονάδα συγχρονισμού ή το \$

```
void ignore (STOKEN first_stoken) {
    STOKEN temp;
    int found=0;
    while ((token!=END_FILE) && (found==0)) {
        temp=first_stoken;
        while (found==0
            && temp!=NULL && temp->next!=NULL)
        {
            temp=temp->next;
            if (temp->stoken==token)
                found=1;
        }
        if (found==0)
            token=getToken();
    }
}
```

H check ελέγχει αν η τιμή της token περιλαμβάνεται , σε ένα σύνολο FIRST, που διαβιβάζεται ως παράμετρος

```
void check ( STOKEN FIRST_first, STOKEN FIRST_last,
            STOKEN FOLLOW_first) {
    STOKEN temp=FIRST_first;
    int found=0;
    while (temp->stoken!=token && temp->next!=NULL) {
        temp=temp->next;
        if (temp->stoken!=token) {
            printf("SYNTAX ERROR\n");
            temp->next=FOLLOW_first;
            ignore(FIRST_first);
        }
    }
}
```

Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη σε κατάσταση «πανικού»

E	$\rightarrow TT_r$
T_r	$\rightarrow '+' TT_r$
	$ '-' TT_r$
	$ \epsilon$
T	$\rightarrow FF_r$
F_r	$\rightarrow '*' FF_r$
	$ '/' FF_r$
	$ \epsilon$
F	$\rightarrow (' E)'$
	$ "αριθμός"$

```

void error() { //syntax error handling
    printf("SYNTAX ERROR\n");
}

void match(TokenType expected_token) {
    if (token==expected_token)
        token=getToken(); //lexical analysis
    else
        error();
}

int E(STOKEN, STOKEN);

```

Η check μπορεί να καλείται από τις συναρτήσεις μη τερματικών δύο φορές. Την πρώτη φορά ελέγχεται αν η token ανήκει στο σύνολο FIRST του μη τερματικού συμβόλου και αν δεν ανήκει, τότε έχουμε λάθος και ενεργοποιείται η διαδικασία ανάκαμψης.

Τη δεύτερη φορά ελέγχεται αν η token ανήκει στο σύνολο FOLLOW του μη τερματικού, κάτι που συμβαίνει είτε μετά από επιτυχή εφαρμογή του κανόνα, είτε μετά την ανάκαμψη από λάθος.

Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη σε κατάσταση «πανικού»

E	$\rightarrow TT_r$	Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη
T_r	$\rightarrow '+' TT_r$	
	$ '-' TT_r$	int F(STOKEN synch_token_first, STOKEN synch_token_last) {
	$ \epsilon$	int val=999; int found=0;
		STOKEN templ, temp2, temp temp=synch_token_first;
T	$\rightarrow FF_r$	templ=(STOKEN) malloc(sizeof(struct stoken_set)); temp2=(STOKEN) malloc(sizeof(struct stoken_set));
F_r	$\rightarrow '*' FF_r$	templ->stoken=LPAR; temp2->stoken=NUMBER; temp2->prev=NULL; temp2->next=NULL; temp->next=temp2;
	$ '/' FF_r$	temp->prev=NULL; temp->next=temp2;
	$ \epsilon$	temp2->prev=templ;
F	$\rightarrow (' E)'$	check(templ,temp2,synch_token_first);
	$ "αριθμός"$	while (temp->stoken!=token && temp->next!=NULL) { temp=temp->next; }

```

} } }

void F() {
    switch(token) {
        case LPAR:
            E();
            match(RPAR);
            break;
        case NUMBER:
            token=getToken();
            break;
        default:
            error();
    }
}

if (temp->stoken!=token) {
    switch(token) {
        case LPAR:
            token=getToken();
            templ->stoken=RPAR;
            templ->prev=NULL;
            templ->next=NULL;
            val=E(templ,temp1);
            match(NUMBER);
            break;
        case NUMBER:
            val=atoi(tokenString);
            token=getToken();
            break;
        default:
            error();
    }
    templ->stoken=LPAR;
    temp2->stoken=NUMBER;
    templ->prev=NULL;
    temp2->next=NULL;
    templ->next=temp2;
    temp2->prev=templ;
    check(synch_token_first,synch_token_last,templ);
}
return val;
}

```

10

Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη σε κατάσταση «πανικού»

E	$\rightarrow TT_r$	Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη
T_r	$\rightarrow +'TT_r$	
T	$ \epsilon$	<pre>int Fr(int val, STOKEN synch_token_first, STOKEN synch_token_last) { int found=0; STOKEN temp; temp=synch_token_first; while (temp->stoken!=token && temp->next!=NULL) { temp=temp->next; } if (temp->stoken!=token) { switch(token) { case MULT: token=getToken(); F(); Fr(); break; case DIV: token=getToken(); F(); Fr(); break; default: return; } } return val; }</pre>
	$ ^'TT_r$	
F_r	$\rightarrow FF_r$	<pre>if (temp->stoken!=token) { switch(token) { case MULT: token=getToken(); val=val*F(synch_token_first, synch_token_last); val=Fr(val,synch_token_first, synch_token_last); break; case DIV: token=getToken(); val=val/F(synch_token_first, synch_token_last); val=Fr(val,synch_token_first, synch_token_last); break; } }</pre>
	$ \epsilon$	
F	$\rightarrow ('E')$	<p>Τα <code>synch_token_first</code> και <code>synch_token_last</code> προσαρμόζουν κατάλληλα το σύνολο των λεξικών μονάδων συγχρονισμού.</p>
	$ "αριθμός"$	

```
void Fr() {
    switch(token) {
        case MULT:
            token=getToken();
            F();
            Fr();
            break;
        case DIV:
            token=getToken();
            F();
            Fr();
            break;
        default:
            return;
    }
}

if (temp->stoken!=token) {
    switch(token) {
        case MULT:
            token=getToken();
            val=val*F(synch_token_first,
                       synch_token_last);
            val=Fr(val,synch_token_first,
                   synch_token_last);
            break;
        case DIV:
            token=getToken();
            val=val/F(synch_token_first,
                       synch_token_last);
            val=Fr(val,synch_token_first,
                   synch_token_last);
            break;
    }
}
return val;
}
```

11

Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη σε κατάσταση «πανικού»

E	$\rightarrow TT_r$	Προβλέπουσα αναδρομική κατάβαση με ανάκαμψη
T_r	$\rightarrow +'TT_r$	
T	$ \epsilon$	<pre>int E(STOKEN synch_token_first, STOKEN synch_token_last) { int val=-999; int found=0; STOKEN templ, temp2, temp; temp=synch_token_first;</pre>
	$ ^'TT_r$	
F_r	$\rightarrow FF_r$	<pre> templ=(STOKEN) malloc(sizeof(struct stoken_set)); check(synch_token_first,synch_token_last,templ); temp2=(STOKEN) malloc(sizeof(struct stoken_set)); templ->stoken=LPAR; templ->prev=NULL; templ->next=NULL; temp2->stoken=NUMBER; temp2->prev=NULL; temp2->next=NULL; templ->next=temp2; temp2->prev=templ;</pre>
	$ \epsilon$	
F	$\rightarrow ('E')$	<pre> templ->next=temp2; temp2->prev=templ; check(templ,temp2,synch_token_first); while (temp->stoken!=token && temp->next!=NULL) { temp=temp->next; } }</pre>
	$ "αριθμός"$	

```
void E() {
    T();
    Tr();
}

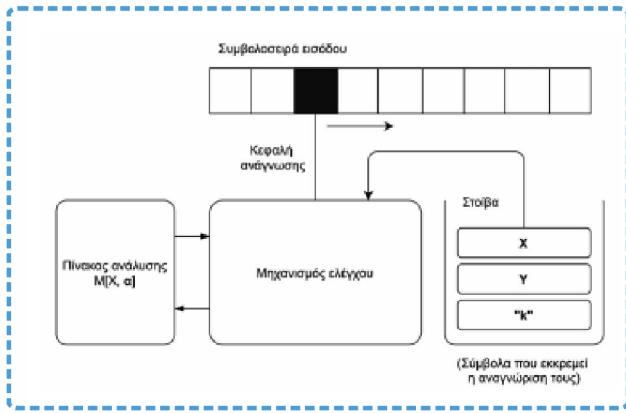
void main() {
    token=getToken(); //lexica
    E();
    if (token != END_FILE) return error();
}
```

Τυπώνεται το αποτέλεσμα της αρ. έκφρασης ή ότι υπάρχει συντ. λάθος

```
void main() {
    token=getToken(); //lexical analysis
    printf("%d", E());
    if (token != END_FILE) return error();
}
```

12

Ανάλυση LL(1)



- Στην ανάλυση $LL(1)$, για κάθε συνδιασμό μη-τερματικού συμβόλου X και τερματικού συμβόλου a , ο αλγόριθμος αποφασίζει την ενέργεια που θα εφαρμώσει, με βάση έναν πίνακα $M[X, a]$
- Ο πίνακας υπολογίζεται εφαρμόζοντας έναν αλγόριθμο στους κανόνες της γραμματικής
- Το μη τερματικό σύμβολο της επόμενης παραγωγής, δίνεται κάθε φορά από την κορυφή της στοίβας που συντηρείται από τον αλγόριθμο ανάλυσης
- Λειτουργεί ως εξής
 - Η στοίβα αρχικοποιείται με το σύμβολο $\$$ και ακολουθείται από το σύμβολο αρχής της γραμματικής
 - Αν στην κορυφή της στοίβας έχουμε τερματικό και είναι ίδιο με αυτό στην κεφαλή ανάγνωσης (τρέχων αναγνωριστικό), αφαιρείται από τη στοίβα και διαβάζεται το επόμενο αναγνωριστικό. Αν τα δύο αναγνωριστικά δεν ταυτίζονται, τότε έχουμε λάθος
 - Αν στην κορυφή της στοίβας έχουμε μη-τερματικό, επιλέγεται από τον πίνακα ανάλυσης M , το δεξιό μέρος του κανόνα που αντιστοιχεί στο μη-τερματικό και στο τερματικό που υποδεικνύει η κεφαλή ανάγωνσης

Αν η θέση του πίνακα έχει τιμή *error* (κενή θέση), τότε έχουμε λάθος. Διαφορετικά αντικαθίσταται το μη τερματικό της κορυφής της στοίβας με τα σύμβολα του δεξιού μέρους του κανόνα που επιλέχθηκε

- Η ανάλυση ολοκληρώνεται με επιτυχία, όταν προσεγγιστεί το τέλος της συμβολοσειράς εισόδου, και η στοίβα περιέχει μόνο το $\$$

Παράδειγμα

Καθοδική ανάλυση $LL(1)$

Πίνακας Ανάλυσης M	E	$\rightarrow TT_r$						
	T_r	$\rightarrow '+' TT_r$						
			$'-' TT_r$					
			ϵ					
	T	$\rightarrow FF_r$						
	F_r	$\rightarrow '*' FF_r$						
			$'/' FF_r$					
			ϵ					
	F	$\rightarrow (' E ')$						
			"αριθμός"					

E	$T T_r$	$'+'$	$'-'$	$'*'$	$'/'$	$('$	$)'$	$\$$
T_r	$'+' TT_r$	$'-' TT_r$				ϵ	ϵ	
T	FF_r				FF_r			
F_r	ϵ	ϵ	$'*' FF_r$	$'/' FF_r$		ϵ	ϵ	
F	"αριθμός"				$(' E ')$			

ΣΤΟΙΒΑ	ΣΥΜΒΟΛΟΣΕΙΡΑ	ΚΑΝΟΝΑΣ
\$E	27 - 5 * 8\$	
\$T_r T	27 - 5 * 8\$	$E \rightarrow TT_r$
\$T_r F_r F	27 - 5 * 8\$	$T \rightarrow FF_r$
\$T_r F_r "αριθμός"	27 - 5 * 8\$	$F \rightarrow "αριθμός"$
\$T_r F_r	-5 * 8\$	
\$T_r	-5 * 8\$	$F_r \rightarrow \epsilon$
\$T_r T' -'	-5 * 8\$	$T_r \rightarrow '-' TT_r$
\$T_r T	5 * 8\$	
\$T_r F_r F	5 * 8\$	$T \rightarrow FF_r$
\$T_r F_r "αριθμός"	5 * 8\$	$F \rightarrow "αριθμός"$
\$T_r F_r	*8\$	
\$T_r F_r F '*'	*8\$	$F_r \rightarrow '*' FF_r$
\$T_r F_r F	8\$	
\$T_r F_r "αριθμός"	8\$	$F \rightarrow "αριθμός"$
\$T_r F_r	8\$	
\$T_r	8\$	$F_r \rightarrow \epsilon$
\$	8\$	$T_r \rightarrow \epsilon$

- Κανόνες ανάλυσης $LL(1)$ – αριστερή παραγωγή της 27-5*8
- $E \Rightarrow TT_r \Rightarrow FF_r T_r \Rightarrow "αριθμός" F_r T_r \Rightarrow "αριθμός" T_r \Rightarrow "αριθμός" - T$
 $\Rightarrow "αριθμός" - FF_r T_r \Rightarrow "αριθμός" - "αριθμός" F_r T_r \Rightarrow$
 $\Rightarrow "αριθμός" - "αριθμός" * F_r T_r \Rightarrow$
 $\Rightarrow "αριθμός" - "αριθμός" * "αριθμός" F_r T_r \Rightarrow$
 $\Rightarrow "αριθμός" - "αριθμός" * "αριθμός"$

28

Αλγόριθμος Υπολογισμού Πίνακα Ανάλυσης

Είσοδος: μία γραμματική G

Έξοδος: ο πίνακας M της ανάλυσης LL(1)

```

1: for all κανόνες  $X \rightarrow \bar{u}$  do
2:   for all τερματικά  $b \in FIRST(\bar{u})$  do
3:     προσθέτουμε τον κανόνα  $X \rightarrow \bar{u}$  στη θέση  $M[X, b]$ ;
4:   if  $\epsilon \in FIRST(\bar{u})$  then
5:     for all τερματικά  $c \in FOLLOW(X)$  do
6:       προσθέτουμε τον κανόνα  $X \rightarrow \bar{u}$  στη θέση  $M[X, c]$ ;
7:   if  $\$ \in FOLLOW(X)$  then
8:     προσθέτουμε τον κανόνα  $X \rightarrow \bar{u}$  στη θέση  $M[X, \$]$ ;
9: στις θέσεις του  $M$  που δεν έχει οριστεί τική, εγχωρείται η τική error;
```

- Για κάθε κανόνα της γραμματικής $X \rightarrow \bar{u}$
 - για κάθε τερματικό $b \in FIRST(\bar{u})$
 - προσθεσε τον κανόνα $X \rightarrow \bar{u}$ στη θέση $M[X, b]$
 - αν $\epsilon \in FIRST(\bar{u})$, βάλε τον κανόνα $X \rightarrow \bar{u}$ στη θέση $M[X, c]$, για κάθε τερματικό $c \in FOLLOW(X)$
 - αν $\$ \in FOLLOW(X)$, βάλε τον κανόνα $X \rightarrow \bar{u}$ στη θέση $M[X, \$]$
- Στις θέσεις που μένουν κενές εννοείται πως γράφουν **error**

Παράδειγμα

$$\begin{array}{lcl}
E & \rightarrow & TT_r \\
T_r & \rightarrow & '+' TT_r \\
& | & '-' TT_r \\
& | & \epsilon \\
T & \rightarrow & FF_r \\
F_r & \rightarrow & '*' FF_r \\
& | & '/' FF_r \\
& | & \epsilon \\
F & \rightarrow & '(' E ')' \\
& | & "αριθμός"
\end{array}$$

$$\begin{array}{ll}
FIRST(E) = \{ '(', "αριθμός" \} & FOLLOW(E) = \{ \$, ')' \} \\
FIRST(T) = \{ '(', "αριθμός" \} & FOLLOW(T) = \{ '+', '−', \$, ')' \} \\
FIRST(F) = \{ '(', "αριθμός" \} & FOLLOW(F_r) = \{ \$, ')' \} \\
FIRST(T_r) = \{ '+', '−', \epsilon \} & FOLLOW(F) = \{ '*', '/', '+', '−', \$, ')' \} \\
FIRST(F_r) = \{ '*', '/', \epsilon \} & FOLLOW(F_r) = \{ '+', '−', \$, ')' \}
\end{array}$$

	"αριθμός"	'+'	'−'	'*'	'/'	'('	')'	\$
E	TT_r					TT_r		
T_r		$+' TT_r$	$− TT_r$				ϵ	ϵ
T	FF_r					FF_r		
F_r		ϵ	ϵ	$* FF_r$	$/ FF_r$		ϵ	ϵ
F	"αριθμός"					$(E)'$		

Ανάλυση LL(1) με Ανάκαμψη σε Κατάσταση "Πανικού"

- Στην $LL(1)$ εντοπίζεται λάθος όταν στην κορυφή της στοίβας έχουμε ένα μη-τερματικό X , τέτοιο ώστε το τρέχον σύμβολο εισόδου να μην ανήκει στο $FIRST(X)$ και αν σε αυτό συμβαίνει να περιέχεται το ϵ , ούτε στο σύνολο $FOLLOW(X)$ (κενές θέσεις του πίνακα)
- Η ανάκαμψη σε κατάσταση "πανικού" υλοποιείται προσδιορίζοντας την κατάλληλη ενέργεια για κάθε περίπτωση κενής θέσης στον πίνακα ανάλυσης
 - pop:** Αφαίρεση του X από τη στοίβα
 - scan:** προσπέρασμα συμβόλων στην είσοδο μέχρι τον εντοπισμό λεξικής μονάδας που μπορεί να χρησιμοποιηθεί για την επανεκκινηση της ανάλυσης
 - Εισαγωγή ενός νέου μη-τερματικού στη στοίβα

Αλγόριθμος

- Ξεκινώντας από τον απλό πίνακα ανάλυσης $LL(1)$
- Σε κάθε θέση $M[X, a]$ (που είναι κενή), με $a \in FOLLOW(X) \cup \{\$\}$, βάλε την ενέργεια **pop**
- Σε κάθε θέση $M[X, a]$ (που είναι κενή), με $a \notin FIRST(X) \cup FOLLOW(X) \cup \{\$\}$, βάλε την ενέργεια **scan**
- Αν αδειάσει η στοίβα, ενώ δεν έχει ολοκληρωθεί ακόμα η ανάγνωση της συμβολοσειράς εισόδου, εισάγεται στη στοίβα το σύμβολο της αρχής, και αγνοούνται όλα τα επόμενα, μέχρι να αναγνωστεί ένα σύμβολο που να ανήκει στο $FIRST$ της αρχής

Παράδειγμα

Ανάκαμψη σε κατάσταση «πανικού» σε ανάλυση $LL(1)$																																																							
E	$\rightarrow TT_r$																																																						
T_r	$\rightarrow '+' TT_r$																																																						
		$'-' TT_r$																																																					
		ϵ																																																					
T	$\rightarrow FF_r$																																																						
F_r	$\rightarrow '*' FF_r$																																																						
		$'/' FF_r$																																																					
		ϵ																																																					
F	$\rightarrow (' E ')$																																																						
		"αριθμός"																																																					
$FIRST(E) = \{(' , "αριθμός"\}$ $FOLLOW(E) = \{$, '\}$ $FIRST(T) = \{(' , "αριθμός"\}$ $FOLLOW(T) = \{+' , '-' , $, '\}\}$ $FIRST(F) = \{(' , "αριθμός"\}$ $FOLLOW(F_r) = \{$, '\}\}$ $FIRST(T_r) = \{+' , '-' , \epsilon\}$ $FOLLOW(F) = \{*' , '/' , '+' , '-' , $, '\}\}$ $FIRST(F_r) = \{*' , '/' , \epsilon\}$ $FOLLOW(F_r) = \{+' , '-' , $, '\}\}$																																																							
<table border="1"> <tr> <td>"αριθμός"</td><td>+'</td><td>'-'</td><td>**'</td><td>/'</td><td>('</td><td>)'</td><td>\$</td></tr> <tr> <td>$E$</td><td>$TT_r$</td><td>scan</td><td>scan</td><td>scan</td><td>scan</td><td>TT_r</td><td>pop</td></tr> <tr> <td>T_r</td><td>scan</td><td>$+' TT_r$</td><td>$'-' TT_r$</td><td>scan</td><td>scan</td><td>ϵ</td><td>ϵ</td></tr> <tr> <td>T</td><td>FF_r</td><td>pop</td><td>pop</td><td>scan</td><td>scan</td><td>FF_r</td><td>pop</td></tr> <tr> <td>F_r</td><td>scan</td><td>ϵ</td><td>$**' FF_r$</td><td>$/' FF_r$</td><td>scan</td><td>ϵ</td><td>ϵ</td></tr> <tr> <td>F</td><td>"αριθμός"</td><td>pop</td><td>pop</td><td>pop</td><td>pop</td><td>$(' E ')$</td><td>pop</td></tr> </table>								"αριθμός"	+'	'-'	**'	/'	(')'	\$	E	TT_r	scan	scan	scan	scan	TT_r	pop	T_r	scan	$+' TT_r$	$'-' TT_r$	scan	scan	ϵ	ϵ	T	FF_r	pop	pop	scan	scan	FF_r	pop	F_r	scan	ϵ	$**' FF_r$	$/' FF_r$	scan	ϵ	ϵ	F	"αριθμός"	pop	pop	pop	pop	$(' E ')$	pop
"αριθμός"	+'	'-'	**'	/'	(')'	\$																																																
E	TT_r	scan	scan	scan	scan	TT_r	pop																																																
T_r	scan	$+' TT_r$	$'-' TT_r$	scan	scan	ϵ	ϵ																																																
T	FF_r	pop	pop	scan	scan	FF_r	pop																																																
F_r	scan	ϵ	$**' FF_r$	$/' FF_r$	scan	ϵ	ϵ																																																
F	"αριθμός"	pop	pop	pop	pop	$(' E ')$	pop																																																

Εκτέλεση

Παράδειγμα

ΣΤΟΙΒΑ	ΣΥΜΒΟΛΟΣΕΙΡΑ	ΚΑΝΟΝΑΣ
$\$E$	$(27 - *)\$$	
$\$T_r T$	$(27 - *)\$$	$E \rightarrow TT_r$
$\$T_r F_r F$	$(27 - *)\$$	$T \rightarrow FF_r$
$\$T_r F_r)' E ' ($	$(27 - *)\$$	$F \rightarrow (^(' E '))$
$\$T_r F_r)' E$	$(27 - *)\$$	
$\$T_r F_r)' T_r T$	$(27 - *)\$$	$E \rightarrow TT_r$
$\$T_r F_r)' T_r F_r F$	$(27 - *)\$$	$T \rightarrow FF_r$
$\$T_r F_r)' T_r F_r "αριθμός"$	$(27 - *)\$$	$F \rightarrow "αριθμός"$
$\$T_r F_r)' T_r F_r$	$(27 - *)\$$	
$\$T_r F_r)' T_r$	$(27 - *)\$$	$F_r \rightarrow \epsilon$
$\$T_r F_r)' T_r T$	$(27 - *)\$$	$T_r \rightarrow (^-' TT_r)$
$\$T_r F_r)' T_r T$	$(27 - *)\$$	
$\$T_r F_r)' T_r T$	$(27 - *)\$$	$scan$
$\$T_r F_r)' T_r$	$(27 - *)\$$	pop
$\$T_r F_r)'$	$(27 - *)\$$	$T_r \rightarrow \epsilon$
$\$T_r$	$(27 - *)\$$	
$\$$	$(27 - *)\$$	$F_r \rightarrow \epsilon$
		$T_r \rightarrow \epsilon$

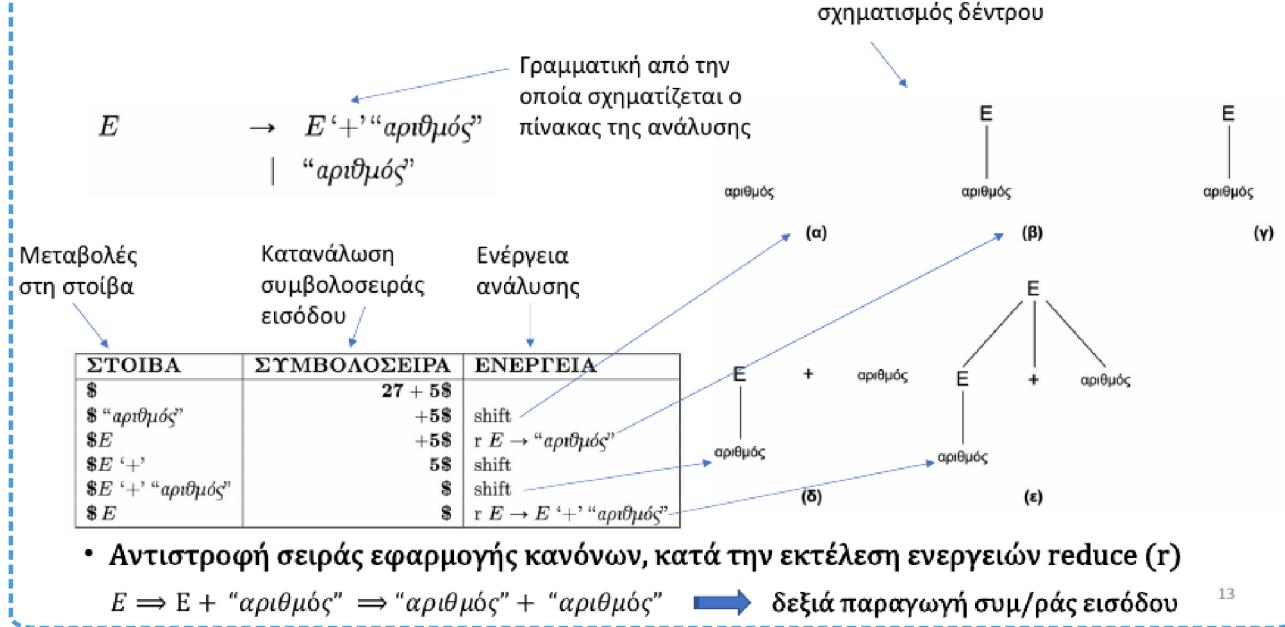
Η ανάλυση της συμβολοσειράς $(27 - *)$ ολοκληρώνεται μετά από την ανάγνωση και του τελευταίου χαρακτήρα, ενώ αν δεν υπήρχαν στον πίνακα ανάλυσης οι ενέργειες ανάκαμψης από λάθος αυτό δε θα μπορούσε να γίνει.
 Η ανάλυση επανέρχεται σε κανονική λειτουργία (κελά με ενέργειες ανάλυσης) μετά από την εκτέλεση δύο ενεργειών ανάκαμψης.
 Για να αποφευχθούν τα διαγνωστικά μηνύματα από διαδιδόμενα λάθη, θα πρέπει να επιτρέπεται η παραγωγή τους μόνο μετά από την επαναφορά της ανάλυσης σε κανονική λειτουργία και την εκτέλεση μιας ή δύο ενεργειών ανάλυσης.

Ανοδική ΣΑ

- Η συμβολοσειρά εισόδου διαβάζεται από τα αριστερά προς τα δεξιά, και στην πορεία απλοποιείται προς το σύμβολο της αρχής της γραμματικής, με εφαρμογή των κανόνων της
- Το παραγόμενο δέντρο αναπτύσσεται σταδιακά, από τα αριστερά προς τα δεξιά και από τα φύλλα προς τη ρίζα
- Η ανοδική ΣΑ μπορεί να ταυτιστεί με την ανάλυση LR

Παράδειγμα

Παράδειγμα ανοδικής ΣΑ



Ανάλυση LR

Βασικές Έννοιες

- Σε κάθε βήμα της ανοδικής ΣΑ εκτελείται μία από τις εξής **Ενέργειες**
 - Απλοποίηση (reduce)**: σύμφωνα με κάποιον κανόνα της γραμματικής (πχ $X \rightarrow u_1 u_2 \dots u_n$) αντικαθιστούνται τα σύμβολα του δεξιού μέρους, από την κορυφή της στοίβας, με το μητερματικό του δεξιού
 - Εισαγωγή Αναγνωστικού (shift)** στη στοίβα της ανάλυσης
 - Αποδοχή (accept)** της συμβολοσειράς εισόδου
- Όταν εκτελείται απολοποίηση με κανόνα-ε ($X \rightarrow \epsilon$), τότε το X εισάγεται στη στοίβα
- Η ανοδική ΣΑ είναι μία αλληλουχία ενεργειών shift και reduce, μέχρι να εμφανιστέι λάθος, ή να προσεγγιστεί κατάσταση αποδοχής (accept)
- Ενεργό Πρόθεμα**: τα σύμβολα που περιέχονται στη στοίβα, σε κάποιο βήμα της εκτέλεσης της ανοδικής ΣΑ
- Λαβή απλοποίησης**: μία ακολουθία συμβόλων στην κορυφή της στοίβας, που ταιριάζει στο δεξί μέρος ενός κανόνα, και η εφαρμογή του αποδίδει ένα βήμα δεξιάς παραγωγής
 - Η ταύτιση με το δεξί μέρος κανόνα δεν αρκεί για να είναι μία συμβολοσειρά λαβή
 - Μόνο όταν ο κανόνας δημιουργεί δεξιά παραγωγή με την εφαρμογή του, θεωρούμε τη συμβολοσειρά, λαβή
- Ο κύρια λειτουργία ενός αλγορίθμου ανοδικής ΣΑ, είναι η **αναγνώριση της επόμενης λαβής**
- Σε μία ανοδική ΣΑ, ο αλγόριθμος μπορεί να βρεθεί σε κατάσταση, που η επόμενη ενέργεια δεν είναι ξεκαθαρη, και πρέπει να επιλαγεί ανάμεσα σε πολλές υποψήφιες
 - Όταν η επιλογή είναι ανάμεσα σε ενέργειες shift και reduce, έχουμε σύγκρουση εισαγωγής-απλοποίησης (shift-reduce conflict)
 - Πιο σπάνια έχουμε και σύγκρουση απλοποίησης-απλοποίησης (reduce-reduce conflict)
- Σε περιπτώσεις shift-reduce conflict προτιμάται η shift, χωρίς να είναι εγκυημένα η σωστή απάντηση
- Σε περιπτώσεις reduce-reduce conflict προτιμάται ο πρώτος από τους δύο κανόνες, χωρίς παλι να είναι εγκυημένα ο σωστός

Γενική Λειτουργία

- Παράγεται ο **Πίνακας Ενέργειών**
 - (Κατάσταση - Επόμενο Σύμβολο) -> Επόμενη Ενέργεια
- Παράγεται ο **Πίνακας Μεταβάσεων**
 - (Κατάσταση - Σύμβολο) -> Κατάσταση
- Ξεκινάμε με κατάσταση 0
- Επόμενη Ενέργεια** (πιν. Ενεργ.) <- πιο δεξιά κατάσταση + επόμενο σύμβολο
- Shift**:
 - Προσθέτουμε νέα ρίζα (με όνομα το σύμβολο εισαγωγής)
 - Κατάσταση νέας ρίζας (πιν. Μετ.) <- πιο δεξιά κατάσταση + σύμβολο εισαγωγής
- Reduce** $X \rightarrow \bar{u}$:
 - νέος κόμβος για το X , συνδέεται ως πρόγονος στις πιο δεξιές ρίζες που αντιστοιχούν στο \bar{u}
 - Κατάσταση ρίζας X (πιν. Μετ.) <- αμέσως αριστ. κατάσταση από συνδεδεμένους κόμβους + σύμβολο εισαγωγής
- Στη στοίβα διατηρείται μαζί με κάθε σύμβολο (ρίζα) και η αντίστοιχη κατάστασή
 - Σε ενέργεια shift, καταχωρείται στην κορυφή της στοίβας, πίσω από το σύμβολο που εισάγεται
 - Σε ενέργεια reduce η λαβή που έχει σχηματιστέι στην κορυφή της στοίβας, αντικαθιστάται αό το μητερματικό στο αριστερό μέρος του κανόνα, μαζί με τη νέα κατάσταση
- Είναι απαραίτητη μία λειτουργεία ανάκαμψης από λάθη

Παράδειγμα

Ανάλυση LR(1)

Πίνακας ενεργειών

	“αριθμός”	ϵ^+	ϵ^-	ϵ^*	$\epsilon/$	(\cdot)	$(\cdot)^*$	$\$$
state 0	s							
state 1		s	s					accept
state 2		r 3	r 3	s				r 3 r 3
state 3		r 6	r 6	r 6	r 6			r 6 r 6
state 4	s					s		
state 5		r 8	r 8	r 8	r 8		r 8	r 8
state 6	s					s		
state 7	s					s		
state 8	s					s		
state 9	s					s		
state 10		s	s				s	
state 11		r 1	r 1	s	s		r 1	r 1
state 12		r 2	r 2	s	s		r 2	r 2
state 13		r 4	r 4	r 4	r 4		r 4	r 4
state 14		r 5	r 5	r 5	r 5		r 5	r 5
state 15		r 7	r 7	r 7	r 7		r 7	r 7

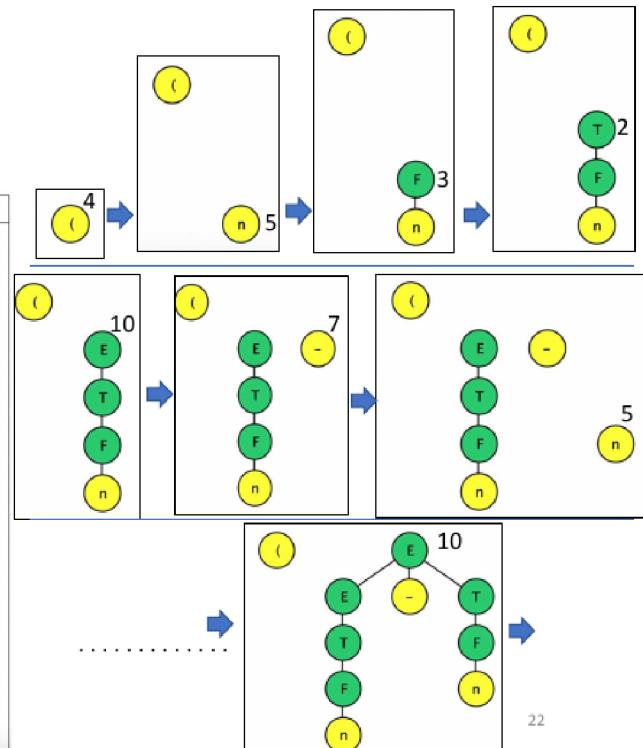
(1) E	$\rightarrow E^+ T$
(2)	$ E^- T$
(3)	$ T$
(4) T	$\rightarrow T^{**} F$
(5)	$ T^*/ F$
(6)	$ F$
(7) F	$\rightarrow (' E ')$
(8)	$ \text{“αριθμός”}$

Πίνακας μεταβάσεων

	“αριθμός”	ϵ^+	ϵ^-	ϵ^*	$\epsilon/$	(\cdot)	$(\cdot)^*$	E	T	F
state 0	5									
state 1		6	7							
state 2				8	9					
state 3										
state 4	5							4	10	2 3
state 5										
state 6	5							4		11 3
state 7	5							4		12 3
state 8	5							4		13
state 9	5							4		14
state 10		6	7							15
state 11				8	9					
state 12				8	9					
state 13										
state 14										
state 15										

Ανάλυση LR(1)

ΣΤΟΙΒΑ	ΣΥΜΒΟΛΟΣΕΙΡΑ	ΕΝΕΡΓΕΙΑ
\$0	(27 - 5) * 8\$	
\$0 '4	27 - 5) * 8\$	shift
\$0 '4 “αριθμός”5	-5) * 8\$	shift
\$0 '4 F3	-5) * 8\$	r 8
\$0 '4 T2	-5) * 8\$	r 6
\$0 '4 E10	-5) * 8\$	r 3
\$0 '4 E10	5) * 8\$	shift
\$0 '4 E10 '7	5) * 8\$	shift
\$0 '4 E10 '7 “αριθμός”5) * 8\$	shift
\$0 '4 E10 '7 F3) * 8\$	r 8
\$0 '4 E10 '7 T12) * 8\$	r 6
\$0 '4 E10) * 8\$	r 2
\$0 '4 E10 ')15	*8\$	shift
\$0 F3	*8\$	r 7
\$0 T2	*8\$	r 6
\$0 T2 '8	8\$	shift
\$0 T2 '8 “αριθμός”5	\$	shift
\$0 T2 '8 F13	\$	r 8
\$0 T2	\$	r 4
\$0 E1	\$	r 3
\$0 E1	\$	accept



Κατασκευή Πινάκων Ανάλυσης LR

Σύνολο Καταστάσεων

- Για κάθε παραλλαγή της ανάλυσης LR πρέπει να υπολογιστεί ένα σύνολο καταστάσεων, από το οποίο προκύπτουν οι πίνακες

LR(0)

- Στοιχείο $LR(0)$ (Ορισμός)**
 - Κάθε στοιχείο του $LR(0)$ ορίζεται από έναν κανόνα παραγωγής και μία τελεία σε κάποια θέση του δεξιού μέρους του κανόνα, που καταγράφει ένα ενδιάμεσο βήμα στην αναγνώριση των συμβόλων που αυτό περιλαμβάνει
- Έστω ο κανόνας $X \rightarrow u v z$
 - με u, v, z οποιαδήποτε τερματικά ή μη τερματικά σύμβολα της γραμματικής
- Τα στοιχεία $LR(0)$ που αναφέρονται στον συγκεκριμένο κανόνα είναι
 - $[X \rightarrow \cdot u v z] <-$ αρχικό στοιχείο $LR(0)$ - τελεία δεξιά
 - $[X \rightarrow u \cdot v z]$
 - $[X \rightarrow u v \cdot z]$
 - $[X \rightarrow u v z \cdot] <-$ συμπληρωμένο στοιχείο $LR(0)$ - τελεία αριστερά
- πχ. το 2ο στοιχείο $LR(0)$ περιγράγει την κατάσταση που έχει ήδη αναγνωριστεί στην είσοδο, συμβολοσειρά που παράγεται από το u και υπάρχει πιθανότητα αναγνώρισης στην είσοδο, συμβολοσειράς που παράγεται από το vz
- Ο όρος $LR(0)$ εκφράζει ότι κάθε στοιχείο δεν εξαρτάται από την τιμή κάποιου συμβόλου εισόδου
- Ο κανόνας $X \rightarrow \epsilon$ παράγει το $LR(0)$ σύνολο με μόνο στοιχείο το $[X \rightarrow \cdot]$
- Ξεκινάμε επεκτείνοντας την γραμματική με ένα κανόνα (0) που ορίζεται ως αρχή της γραμματικής και παίρνει τη μορφή

$$(0)S \rightarrow E$$

, με E την παλιά αρχή της γραμματικής. Μία συμβολοσειρά εισόδου γίνεται αποδεκτή, όταν ο αναλυτής θα μπορούσε να εκτελέσει απλοποίηση για τον κανόνα (0)

- Αντιστοιχίζουμε την αρχική κατάσταση (state 0) το στοιχείο

$$[S \rightarrow \cdot E]$$

που εκφράζει ότι δεν έχει αναγνωριστεί κάποιο μη-κενό ενεργό πρόθεμα

- Εφαρμόζουμε τους εξής κανόνες για κάθε σύνολο στο εξής:
 - Επέκταση μη-τερματικών (ε-κλείσιμο)** : για κάθε στοιχείο $[Y \rightarrow \bar{u} \cdot X \bar{o}]$ (με \cdot πριν μη-τερματικό), προσθέτουμε στο ίδιο σύνολο και καθε στοιχείο της μορφής $[X \rightarrow \cdot \bar{u}]$ για κάθε κανόνα της μορφής $X \rightarrow \bar{u}$, δηλαδή κάθε κανόνα με το μη-τερματικό X στο αριστερό του μέρος, με την \cdot στην αρχή
 - Δημιουργία νέων συνόλων (ανάγνωση)**: για κάθε σύμβολο (τερματικό ή μη) με \cdot πριν από αυτό σε κάποιο στοιχείο του συνόλου, δημιουργησε ένα νέο σύνολο, και αρχικοποίησέ το με, βάζοντας την \cdot μετά από αυτό, για κάθε στοιχείο που εμφανίζεται
 - Αυτό μπορεί να γραφτεί και ως $\text{ανάγνωση}(I_i, X) = I_j$, όπου
 - I_i : το παρόν σύνολο
 - X : το σύμβολο που "πυροδώτησε" τη δημιουργία νέου συνόλου
 - I_j : το νέο σύνολο
 - Σημείωσε τις παραπάνω πληροφορίες γιατί θα χρειαστούν αργότερα

Παράδειγμα

(0)	S	$\rightarrow E$
(1)	E	$\rightarrow E '+' T$
(2)		$ E '-' T$
(3)		$ T$
(4)	T	$\rightarrow T '*' F$
(5)		$ T '/' F$
(6)		$ F$
(7)	F	$\rightarrow ('(E)'$
(8)		$ "αριθμός"$

Η κατασκευή συνεχίζεται, οπότε με τις πράξεις ανάγνωση συμβόλου και ε-κλείσιμο, παίρνουμε τα σύνολα στοιχείων:

state 0 επόμενα σύμβολα: E - state 1 T - state 2 F - state 3 $($ - state 4 αριθμός - state 5	$[S \rightarrow \cdot E]$ [$E \rightarrow \cdot E + T$] [$E \rightarrow \cdot E - T$] [$E \rightarrow \cdot T$] [$T \rightarrow \cdot T * F$] [$T \rightarrow \cdot T / F$] [$T \rightarrow \cdot F$] [$F \rightarrow \cdot (E)$] [$F \rightarrow \cdot αριθμός$]	state 4 επόμενα σύμβολα: E - state 10 T - state 2 F - state 3 $($ - state 4 αριθμός - state 5	$[F \rightarrow (\cdot E)]$ [$E \rightarrow \cdot E + T$] [$E \rightarrow \cdot E - T$] [$E \rightarrow \cdot T$] [$T \rightarrow \cdot T * F$] [$T \rightarrow \cdot T / F$] [$T \rightarrow \cdot F$] [$F \rightarrow \cdot (E)$] [$F \rightarrow \cdot αριθμός$]	state 8 επόμενα σύμβολα: F - state 13 $($ - state 4 αριθμός - state 5	$[T \rightarrow T * \cdot F]$ [$T \rightarrow \cdot F$] [$F \rightarrow \cdot αριθμός$]	state 12 επόμενα σύμβολα: $*$ - state 8 $/$ - state 9	$[E \rightarrow E - \cdot T]$ [$T \rightarrow T \cdot * F$] [$T \rightarrow T \cdot / F$]
state 1 επόμενα σύμβολα: $+$ - state 6 $-$ - state 7	$[S \rightarrow E]$ [$E \rightarrow E \cdot + T$] [$E \rightarrow E \cdot - T$]	state 5	$[F \rightarrow αριθμός]$	state 9 επόμενα σύμβολα: F - state 14 $($ - state 4 αριθμός - state 5	$[T \rightarrow T \cdot / F]$ [$F \rightarrow \cdot (E)$] [$F \rightarrow \cdot αριθμός$]	state 13	$[T \rightarrow T \cdot F]$
state 2 επόμενα σύμβολα: $*$ - state 8 $/$ - state 9	$[E \rightarrow T]$ [$T \rightarrow T \cdot * F$] [$T \rightarrow T \cdot / F$]	state 6 επόμενα σύμβολα: T - state 11 F - state 3 $($ - state 4 αριθμός - state 5	$[E \rightarrow E + \cdot T]$ [$T \rightarrow T \cdot F$] [$T \rightarrow T / F$] [$T \rightarrow \cdot F$] [$F \rightarrow \cdot (E)$] [$F \rightarrow \cdot αριθμός$]	state 10 επόμενα σύμβολα: $)$ - state 15 $+$ - state 6 $-$ - state 7	$[F \rightarrow (E \cdot)]$ [$E \rightarrow E \cdot + T$] [$E \rightarrow E \cdot - T$]	state 14	$[T \rightarrow T \cdot F]$
state 3	$[T \rightarrow F]$	state 7 επόμενα σύμβολα: T - state 12 F - state 3 $($ - state 4 αριθμός - state 5	$[E \rightarrow E - \cdot T]$ [$T \rightarrow T \cdot F$] [$T \rightarrow T / F$] [$T \rightarrow \cdot F$] [$F \rightarrow \cdot (E)$] [$F \rightarrow \cdot αριθμός$]	state 11 επόμενα σύμβολα: $*$ - state 8 $/$ - state 9	$[E \rightarrow E + \cdot T]$ [$T \rightarrow T \cdot * F$] [$T \rightarrow T \cdot / F$]	state 15	$[F \rightarrow (E \cdot)]$

26

- Για να είναι μία γλώσσα $LR(0)$ είναι αναγκαίο:
 - Να μην υπάρχει σύνολο στοιχείων που περιέχει ταυτόχρονα ένα συμπληρωμένο στοιχείο (\cdot στο τέλος) και ένα στοιχείο της μορφής $[Y \rightarrow \bar{u} \cdot k\bar{o}]$, με k τερματικό
 - Κάθε σύνολο στοιχείων να περιέχει το πολύ ένα συμπληρωμένο στοιχείο

LR(1)

- **Στοιχείο LR(1) (Ορισμός)**

- Κάθε στοιχείο $LR(1)$ έχει τη γενική μορφή

$$[A \rightarrow X_1 \dots X_i \cdot X_{i+1} \dots X_j, a]$$

και περιγράφει ότι έχουν σχηματιστεί στη στοίβα τα $X_1 \dots X_i$ και αναμένεται να σχηματιστούν τα $X_{i+1} \dots X_j$ και μετά να απλοποιηθούν, στην περίπτωση που ακολουθεί το τερματικό a (*lookahead symbol*) στην είσοδο

- Το lookahead symbol μπορεί να είναι οποιοδήποτε τερματικό ή το $\$$ και επηρεάζει μόνο όταν βρίσκεται σε $LR(1)$ στοιχεία της μορφής $[A \rightarrow X_1 \dots X_j, a]$, (όταν η \cdot είναι στο τέλος), όταν δηλαδή μπορεί να γίνει απλοποίηση μόνο με επόμενο στοιχείο εισόδου a
- Μπορεί για ένα στοιχείο να έχουμε περισσότερα από ένα lookahead symbols, οπότε για ευκολεία τα απαριθμούμε σε μία λίστα, διαχωρίζοντάς τα με /
- δηλαδή τα $[A \rightarrow \bar{u}, a]$, $[A \rightarrow \bar{u}, b]$ και $[A \rightarrow \bar{u}, c]$, μπορουν να γραφτούν

$$[A \rightarrow \bar{u}, a/b/c]$$

- Το lookahead ενός στοιχείου $LR(1)$ θα είναι πάντα ένα υποσύνολο του A

Για να παράξουμε τα σύνολα στοιχείων $LR(1)$ μίας γραμματικής

1. Προσθέτουμε κανόνα (0) της μορφής $S \rightarrow E$, με E την προηγούμενη αρχή της γραμματικής
2. Αρχικοποιούμε το σύνολο I_0 με το στοιχείο $[S \rightarrow \cdot A, \$]$
3. Επεκτείνουμε κάθε σύνολο και δημιουργούμε νέα σύνολα ακολουθώντας τους εξής κανόνες
 - **Επέκταση μη-τερματικών (ε-κλείσιμο)**: για κάθε στοιχείο της μορφής $[Y \rightarrow \bar{u} \cdot X\bar{o}, a] \in I_i$ (με \cdot πριν από κάποιο μη-τερματικό X) προσθέτουμε στο ίδιο σύνολο I_i ένα νέο στοιχείο για κάθε κανόνα $X \rightarrow \bar{w}$ (με το X στο αριστερό του μέρος), τοποθετώντας την \cdot στην αρχή του αριστερού του μέρους και με lookahead όλα τα στοιχεία του $FIRST(\bar{o}a)$, δηλαδή

$$[X \rightarrow \cdot \bar{w}, a_1/a_2/\dots/a_n]$$

- όπου $FIRST(\bar{o}a) = a_1, a_2, \dots, a_n$ (και $\$$ όπου βγαίνει κενό)
- **Δημιουργία νέων συνόλων (ανάγνωση)**: για κάθε σύμβολο (τερματικό ή μη) με \cdot πριν από αυτό σε κάποιο στοιχείο του συνόλου, δημιουργούμε ένα νέο σύνολο, και αρχικοποίησέ το με, βάζοντας την \cdot μετά από αυτό, για κάθε στοιχείο που εμφανίζεται
 - Αυτό μπορεί να γραφτεί και ως $ανάγνωση(I_i, X) = I_j$, όπου
 - I_i : το παρόν σύνολο
 - X : το σύμβολο που "πυροδώτησε" τη δημιουργία νέου συνόλου
 - I_j : το νέο σύνολο
 - Σημείωσε τις παραπάνω πληροφορίες γιατί θα χρειαστούν αργότερα

Παράδειγμα

(0)	S	\rightarrow	A
(1)	A	\rightarrow	$X X$
(2)	X	\rightarrow	"a" X
(3)	X	\rightarrow	"b"

state 0 επόμενα σύμβολα: A - state 1 X - state 2 a - state 3 b - state 4	$[S \rightarrow \cdot A, \$]$ $[A \rightarrow \cdot XX, \$]$ $[X \rightarrow \cdot aX, a/b]$ $[X \rightarrow \cdot b, a/b]$	state 5	$[A \rightarrow XX\cdot, \$]$
state 1	$[S \rightarrow A\cdot, \$]$	state 6 επόμενα σύμβολα: X - state 9 a - state 6 b - state 7	$[X \rightarrow a \cdot X, \$]$ $[X \rightarrow \cdot aX, \$]$ $[X \rightarrow \cdot b, \$]$
state 2 επόμενα σύμβολα: X - state 5 a - state 6 b - state 7	$[A \rightarrow X \cdot X, \$]$ $[X \rightarrow \cdot aX, \$]$ $[X \rightarrow \cdot b, \$]$	state 7	$[X \rightarrow b\cdot, \$]$
state 3 επόμενα σύμβολα: X - state 8 a - state 3 b - state 4	$[X \rightarrow a \cdot X, a/b]$ $[X \rightarrow \cdot aX, a/b]$ $[X \rightarrow \cdot b, a/b]$	state 8	$[X \rightarrow aX\cdot, a/b]$
state 4	$[X \rightarrow b\cdot, a/b]$	state 9	$[X \rightarrow aX\cdot, \$]$

Πίνακες

LR(0)

Είσοδος: μία γραμματική G με αρχή το σύμβολο S

Έξοδος: πίνοκας ενεργειών M και μεταβάσεων N ανάλυσης $LR(0)$

- 1: Δημιουργείται το σύνολο $I = \{I_0, I_1, \dots, I_n\}$;
- 2: **for** $i := 0$ **to** n **do**
- 3: **if** $([X \rightarrow \bar{e}] \in I_i \text{ και } X \neq S)$ **then**
- 4: **for all** σύμβολο εισόδου α **do**
- 5: $M[i, \alpha] := \text{reduce } X \rightarrow \bar{e};$
- 6: **if** $([S \rightarrow \bar{e}\cdot] \in I_i)$ **then**
- 7: $M[i, \$] := \text{accept};$
- 8: **for all** $([Y \rightarrow \bar{u} \cdot k \bar{o}] \in I_i \text{ για } k \text{ τερματικό})$ **do**
- 9: **if** $(\text{ανάγνωση}(I_i, k) = I_j)$ **then**
- 10: $M[i, k] := \text{shift};$
- 11: $N[i, k] := j;$
- 12: **for all** μη τερματικό σύμβολο X **do**
- 13: **if** $(\text{ανάγνωση}(I_i, X) = I_j)$ **then**
- 14: $N[i, X] := j;$

- Για τον πίνακα **Μεταβάσεων** N :

- Για κάθε σύνολο I_i οι μεταβάσεις του είναι οι **αναγνώσεις ανάγνωση** $(I_i, X) = I_j$ μέσω των συμβόλων X στα σύνολα I_j

- Για τον πίνακα **Ενεργειών** M :

- Για κάθε στοιχείο του N που δεν είναι κενό, βάλε στην αντίστοιχη θέση του M το **shift**
- Αν ένα σύνολο I_i έχει στοιχείο της μορφής $[X \rightarrow \bar{e}\cdot]$ (με την \cdot στο τέλος), αν το X είναι το η αρχή της γραμματικής, S , βάλε **accept** στο σύμβολο $\$$, διαφορετικά, βάλε **reduce** $X \rightarrow \bar{e}$ σε όλα τα σύμβολα (που δεν έχουν ήδη **shift**)

Παράδειγμα (στον πίνακα ενεργειών σκέψου όπου υπάρχει μία ενέργεια r να επεκτείνεται σε όλη τη γραμμή)

Οτούχειαν:					
state 0	$[S \rightarrow \cdot E]$ επόμενα σύμβολα: $E - state 1$ $[E \rightarrow \cdot E + T]$ $E - state 2$ $[E \rightarrow \cdot T]$ $F - state 3$ $(- state 4$ $[T \rightarrow \cdot T * F]$ $[T \rightarrow \cdot T / F]$ αριθμός - state 5 $[T \rightarrow \cdot F]$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alphaριθμός]$	state 4 επόμενα σύμβολα: $E - state 10$ $[E \rightarrow \cdot E + T]$ $T - state 2$ $[E \rightarrow \cdot T]$ $F - state 3$ $(- state 4$ $[T \rightarrow \cdot T * F]$ $[T \rightarrow \cdot T / F]$ αριθμός - state 5 $[T \rightarrow \cdot F]$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alphaριθμός]$	state 8 επόμενα σύμβολα: $F - state 13$ $(- state 4$ αριθμός - state 5 $[T \rightarrow \cdot F]$ $[T \rightarrow \cdot T * F]$ $[T \rightarrow \cdot T / F]$	state 12 επόμενα σύμβολα: $* - state 8$ $/ - state 9$ $[E \rightarrow E - T]$ $[T \rightarrow T * F]$ $[T \rightarrow T / F]$	
state 1	$[S \rightarrow E \cdot]$ επόμενα σύμβολα: $+ - state 6$ $-- state 7$	state 5 $[F \rightarrow \alphaριθμός]$	state 9 επόμενα σύμβολα: $F - state 14$ $(- state 4$ αριθμός - state 5 $[T \rightarrow \cdot F]$ $[T \rightarrow \cdot T * F]$ $T - state 11$ $[T \rightarrow \cdot T / F]$ $F - state 3$ $[T \rightarrow \cdot F]$ $(- state 4$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alphaριθμός]$	state 13 $[T \rightarrow T * F]$	
state 2 επόμενα σύμβολα: $* - state 8$ $/ - state 9$	$[E \rightarrow T \cdot]$ $[T \rightarrow T \cdot * F]$ $[T \rightarrow T \cdot / F]$	state 6 επόμενα σύμβολα: $T - state 12$ $[T \rightarrow \cdot T * F]$ $F - state 3$ $[T \rightarrow \cdot F]$ $(- state 4$ $[F \rightarrow \cdot (E)]$ αριθμός - state 5 $[F \rightarrow \cdot \alphaριθμός]$	state 10 επόμενα σύμβολα: $) - state 15$ $+ - state 6$ $- - state 7$ $[F \rightarrow (E \cdot)]$ $[E \rightarrow E \cdot + T]$ $[E \rightarrow E \cdot - T]$	state 14 $[T \rightarrow T / F]$	
state 3	$[T \rightarrow F \cdot]$	state 7 επόμενα σύμβολα: $F - state 3$ $(- state 4$ αριθμός - state 5 $[T \rightarrow \cdot F]$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alphaριθμός]$	state 11 επόμενα σύμβολα: $* - state 8$ $/ - state 9$ $[E \rightarrow E + T \cdot]$ $[T \rightarrow T \cdot * F]$ $[T \rightarrow T \cdot / F]$	state 15 $[F \rightarrow (E \cdot)]$	

26

	<i>“αριθμός”</i>	<i>+</i>	<i>-</i>	<i>*?</i>	<i>/?</i>	<i>()?</i>	<i>\$</i>
state 0	s					s	
state 1		s	s				accept
state 2		r 3	r 3	s		r 3	r 3
state 3		r 6	r 6	r 6	r 6	r 6	r 6
state 4	s					s	
state 5		r 8	r 8	r 8	r 8	r 8	r 8
state 6	s					s	
state 7	s					s	
state 8	s					s	
state 9	s					s	
state 10		s	s			s	
state 11		r 1	r 1	s	s	r 1	r 1
state 12		r 2	r 2	s	s	r 2	r 2
state 13		r 4	r 4	r 4	r 4	r 4	r 4
state 14		r 5	r 5	r 5	r 5	r 5	r 5
state 15		r 7	r 7	r 7	r 7	r 7	r 7

SLR(1)

Είσοδος: μία γραμματική G με αρχή το σύμβολο S

Έξοδος: πίνακας ενεργειών M και μεταβάσεων N ανάλυσης $LR(0)$

```

1: Δημιουργείται το σύνολο  $I = \{I_0, I_1, \dots, I_n\}$ ;
2: for i:=0 to n do
3:   for all  $([X \rightarrow \bar{e}] \in I_i \text{ με } X \neq S)$  do
4:     for all σύμβολο εισόδου  $\alpha \in FOLLOW(X)$  do
5:        $M[i, \alpha] := \text{reduce } X \rightarrow \bar{e};$ 
6:     if  $([S \rightarrow \bar{e}] \in I_i)$  then
7:        $M[i, \$] := \text{accept};$ 
8:     for all  $([Y \rightarrow \bar{u} \cdot k \bar{o}] \in I_i \text{ για } k \text{ τερματικό})$  do
9:       if  $(\alpha \in FOLLOW(Y))$  then
10:         $M[i, k] := \text{shift};$ 
11:         $N[i, k] := j;$ 
12:      for all μη τερματικό σύμβολο  $X$  do
13:        if  $(\alpha \in FOLLOW(X))$  then
14:           $N[i, X] := j;$ 

```

βασική διαφορά σε σχέση με τον αλγόριθμο για την κατασκευή των πινάκων της ανάλυσης $LR(0)$

31

- Για τον πίνακα **Μεταβάσεων** N :

- Για κάθε σύνολο I_i οι μεταβάσεις του είναι οι αναγνώσεις ανάγνωσης $\alpha \in FOLLOW(I_i) = I_j$ μέσω των συμβόλων X στα σύνολα I_j

- Για τον πίνακα **Ενεργειών** M :

- Για κάθε στοιχείο του N που δεν είναι κενό, βάλε στην αντίστοιχη θέση του M το *shift*
- Αν ένα σύνολο I_i έχει στοιχείο της μορφής $[X \rightarrow \bar{e}]$ (με την \cdot στο τέλος), αν το X είναι το η αρχή της γραμματικής, S , βάλε *accept* στο σύμβολο $\$$, διαφορετικά, βάλε *reduce* $X \rightarrow \bar{e}$ σε όλα τα σύμβολα του $FOLLOW(X)$ (που δεν έχουν ήδη *shift*)

Παράδειγμα

ΟΤΟΥΧΙΩΝ:

state 0	$[S \rightarrow \cdot E]$ επόμενα σύμβολα: $[E \rightarrow \cdot E + T]$ $E - \text{state 1}$ $[E \rightarrow \cdot E - T]$ $T - \text{state 2}$ $[E \rightarrow \cdot T]$ $F - \text{state 3}$ $[T \rightarrow \cdot T * F]$ $(- \text{state 4}$ $[T \rightarrow \cdot T / F]$ $\alpha \text{ριθμός} - \text{state 5}$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alpha \text{ριθμός}]$	state 4 επόμενα σύμβολα: $[E \rightarrow \cdot E + T]$ $E - \text{state 10}$ $[E \rightarrow \cdot E - T]$ $T - \text{state 2}$ $[E \rightarrow \cdot T]$ $F - \text{state 3}$ $[T \rightarrow \cdot T * F]$ $(- \text{state 4}$ $[T \rightarrow \cdot T / F]$ $\alpha \text{ριθμός} - \text{state 5}$ $[T \rightarrow \cdot F]$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alpha \text{ριθμός}]$	$[F \rightarrow (\cdot E)]$ $[E \rightarrow \cdot E + T]$ $[E \rightarrow \cdot E - T]$ $[E \rightarrow \cdot T]$ $[T \rightarrow \cdot T * F]$ $[T \rightarrow \cdot T / F]$ $[F \rightarrow \cdot F]$ $[F \rightarrow \cdot (E)]$ $[F \rightarrow \cdot \alpha \text{ριθμός}]$
state 1	$[S \rightarrow E \cdot]$ επόμενα σύμβολα: $[E \rightarrow E \cdot + T]$ $+ - \text{state 6}$ $- - \text{state 7}$	state 5 $[F \rightarrow \alpha \text{ριθμός}]$	$[T \rightarrow \alpha \text{ριθμός}]$
state 2	$[E \rightarrow T \cdot]$ επόμενα σύμβολα: $[T \rightarrow T \cdot * F]$ $* - \text{state 8}$ $/ - \text{state 9}$	state 6 επόμενα σύμβολα: $[T \rightarrow E + T]$ $E - \text{state 11}$ $[T \rightarrow E - T]$ $F - \text{state 3}$ $(- \text{state 4}$ $[F \rightarrow \cdot (E)]$ $\alpha \text{ριθμός} - \text{state 5}$ $[F \rightarrow \cdot \alpha \text{ριθμός}]$	$[T \rightarrow T \cdot F]$ επόμενα σύμβολα: $[T \rightarrow \cdot F]$ $F - \text{state 13}$ $(- \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$
state 3	$[T \rightarrow F \cdot]$	state 7 επόμενα σύμβολα: $[T \rightarrow E + T]$ $E - \text{state 12}$ $[T \rightarrow E - T]$ $F - \text{state 3}$ $(- \text{state 4}$ $[F \rightarrow \cdot (E)]$ $\alpha \text{ριθμός} - \text{state 5}$ $[F \rightarrow \cdot \alpha \text{ριθμός}]$	$[T \rightarrow T \cdot * F]$ επόμενα σύμβολα: $[T \rightarrow \cdot F]$ $F - \text{state 12}$ $(- \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$

state 8 επόμενα σύμβολα: $[T \rightarrow E + T]$ $E - \text{state 13}$ $[T \rightarrow E - T]$ $F - \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$	$[T \rightarrow T \cdot F]$ επόμενα σύμβολα: $[T \rightarrow \cdot F]$ $F - \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$	state 12 επόμενα σύμβολα: $[T \rightarrow E - T]$ $E - \text{state 8}$ $[T \rightarrow T \cdot F]$ $/ - \text{state 9}$	$[E \rightarrow E - T]$ επόμενα σύμβολα: $[T \rightarrow T \cdot * F]$ $* - \text{state 8}$ $[T \rightarrow T \cdot / F]$
state 9 επόμενα σύμβολα: $[F \rightarrow \cdot (E)]$ $F - \text{state 14}$ $(- \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$	$[T \rightarrow T \cdot F]$ επόμενα σύμβολα: $[F \rightarrow \cdot (E)]$ $F - \text{state 14}$ $(- \text{state 4}$ $\alpha \text{ριθμός} - \text{state 5}$	state 13 $[T \rightarrow T \cdot * F]$	state 13 $[T \rightarrow T \cdot * F]$
state 10 επόμενα σύμβολα: $[E \rightarrow E \cdot + T]$ $+ - \text{state 15}$ $- - \text{state 6}$ $- - \text{state 7}$	$[F \rightarrow (E \cdot)]$ επόμενα σύμβολα: $[E \rightarrow E \cdot + T]$ $+ - \text{state 15}$ $- - \text{state 6}$ $- - \text{state 7}$	state 14 $[T \rightarrow T \cdot / F]$	state 14 $[T \rightarrow T \cdot / F]$
state 11 επόμενα σύμβολα: $[T \rightarrow T \cdot * F]$ $* - \text{state 8}$ $/ - \text{state 9}$	$[E \rightarrow E + T \cdot]$ επόμενα σύμβολα: $[T \rightarrow T \cdot * F]$ $* - \text{state 8}$ $/ - \text{state 9}$	state 15 $[F \rightarrow (E \cdot)]$	state 15 $[F \rightarrow (E \cdot)]$

26

	“ $\alpha \text{ριθμός}$ ”	$\cdot +$	$\cdot -$	$\cdot *$	$\cdot /$	$\cdot ($	$\cdot)$	$\$$
state 0	s					s		
state 1	s	s						accept
state 2	r 3	r 3	s			r 3	r 3	
state 3	r 6	r 6	r 6	r 6		r 6	r 6	
state 4	s				s			
state 5	r 8	r 8	r 8	r 8		r 8	r 8	
state 6	s				s			
state 7	s				s			
state 8	s				s			
state 9	s				s			
state 10	s	s				s		
state 11	r 1	r 1	s	s		r 1	r 1	
state 12	r 2	r 2	s	s		r 2	r 2	
state 13	r 4	r 4	r 4	r 4		r 4	r 4	
state 14	r 5	r 5	r 5	r 5		r 5	r 5	
state 15	r 7	r 7	r 7	r 7		r 7	r 7	

	“ $\alpha \text{ριθμός}$ ”	$\cdot +$	$\cdot -$	$\cdot *$	$\cdot /$	$\cdot ($	$\cdot)$	E	T	F
state 0	5							4	1	2
state 1			6	7						
state 2					8	9				
state 3										
state 4	5							4	10	2
state 5										
state 6	5							4	11	3
state 7	5							4	12	3
state 8	5							4		13
state 9	5							4		14
state 10		6	7						15	
state 11					8	9				
state 12					8	9				
state 13										
state 14										
state 15										

26

(Κανονική) LR(1)

Είσοδος: μία γραμματική G με αρχή το σύμβολο S
Έξοδος: πίνακας ενεργειών M και μεταβάσεων N ανάλυσης $LR(1)$

```

1: Δημιουργείται το σύνολο  $I = \{I_0, I_1, \dots, I_n\}$ ; /
2: for i:=0 to n do
3:   for all  $([X \rightarrow \bar{e}, a] \in I_i \text{ με } X \neq S)$  do
4:      $M[i, a] := \text{reduce } X \rightarrow \bar{e};$ 
5:   if  $([S \rightarrow \bar{e}, \$] \in I_i)$  then
6:      $M[i, \$] := \text{accept};$ 
7:   for all  $([Y \rightarrow \bar{u} \cdot k \bar{o}, b] \in I_i \text{ για } k \text{ τερματικό})$  do
8:     if  $(\alpha\gamma\eta\omega\sigma(I_i, k) = I_j)$  then
9:        $M[i, k] := \text{shift};$ 
10:       $N[i, k] := j;$ 
11:    for all μη τερματικό σύμβολο  $X$  do
12:      if  $(\alpha\gamma\eta\omega\sigma(I_i, X) = I_j)$  then
13:         $N[i, X] := j;$ 

```

- Για τον πίνακα **Μεταβάσεων** N
 - Για κάθε σύνολο I_i οι μεταβάσεις του είναι οι αναγνώσεις $\alpha\gamma\eta\omega\sigma(I_i, X) = I_j$ μέσω των συμβόλων X στα σύνολα I_j
- Για τον πίνακα **Ενεργειών** M
 - Για κάθε στοιχείο του N που δεν είναι κενό, βάλε στην αντίστοιχη θέση του M το *shift*
 - Αν ένα σύνολο I_i έχει στοιχείο της μορφής $[X \rightarrow \bar{e}, a]$ (με την \cdot στο τέλος), αν το X είναι το η αρχή της γραμματικής, S και το $a = \$$, βάλε *accept* στο σύμβολο $\$$, διαφορετικά, βάλε *reduce* $X \rightarrow \bar{e}$ σε όλα τα σύμβολα a (που δεν έχουν ήδη *shift*)

Παράδειγμα

state 0 επόμενα σύμβολα: A - state 1 X - state 2 a - state 3 b - state 4	$[S \rightarrow \cdot A, \$]$ $[A \rightarrow \cdot XX, \$]$ $[X \rightarrow \cdot aX, a/b]$ $[X \rightarrow \cdot b, a/b]$	state 5	$[A \rightarrow XX\cdot, \$]$
state 1	$[S \rightarrow A\cdot, \$]$	state 6 επόμενα σύμβολα: X - state 9 a - state 6 b - state 7	$[X \rightarrow a \cdot X, \$]$ $[X \rightarrow \cdot aX, \$]$ $[X \rightarrow \cdot b, \$]$
state 2 επόμενα σύμβολα: X - state 5 a - state 6 b - state 7	$[A \rightarrow X \cdot X, \$]$ $[X \rightarrow \cdot aX, \$]$ $[X \rightarrow \cdot b, \$]$	state 7	$[X \rightarrow b\cdot, \$]$
state 3 επόμενα σύμβολα: X - state 8 a - state 3 b - state 4	$[X \rightarrow a \cdot X, a/b]$ $[X \rightarrow \cdot aX, a/b]$ $[X \rightarrow \cdot b, a/b]$	state 8	$[X \rightarrow aX\cdot, a/b]$
state 4	$[X \rightarrow b\cdot, a/b]$	state 9	$[X \rightarrow aX\cdot, \$]$

	“a”	“b”	\$
state 0	s	s	
state 1			accept
state 2	s	s	
state 3	s	s	
state 4	r 3	r 3	
state 5			r 1
state 6	s	s	
state 7			r 3
state 8	r 2	r 2	
state 9			r 2

	“a”	“b”	A	X
state 0	3	4	1	2
state 1				
state 2	6	7		5
state 3	3	4		8
state 4				
state 5				
state 6	6	7		9
state 7				
state 8				
state 9				

LALR(1)

Παράδειγμα

Ανάλυση LALR(1)

LR(1)

state 0 επόμενα σύμβολα: A - state 1 X - state 2 a - state 3 b - state 4	[S → · A, \$] [A → · XX, \$] [X → · aX, a/b] [X → · b, a/b]	state 5	[A → XX·, \$]
state 1	[S → A·, \$]	state 6 επόμενα σύμβολα: X - state 9 a - state 6 b - state 7	[X → a · X, \$] [X → · aX, \$] [X → · b, \$]
state 2 επόμενα σύμβολα: X - state 5 a - state 6 b - state 7	[A → X · X, \$] [X → · aX, \$] [X → · b, \$]	state 7	[X → b · , \$]
state 3 επόμενα σύμβολα: X - state 8 a - state 3 b - state 4	[X → a · X, a/b] [X → · aX, a/b] [X → · b, a/b]	state 8	[X → aX·, a/b]
state 4	[X → b · , a/b]	state 9	[X → aX·, \$]

- Παρατηρούμε ότι κάποιες από τις καταστάσεις της LR(1) ανάλυσης έχουν σημαντικές ομοιότητες μεταξύ τους, π.χ. οι 3 και 6. Αυτές οι καταστάσεις διαφέρουν μόνο ως προς τα σύνολα των συμβόλων lookahead. Το ίδιο επίσης ισχύει για τις καταστάσεις 4 & 7, καθώς και για 8 & 9.
- Στην συγκεκριμένη γραμματική με την ανάλυση LALR(1) έχουμε πολύ πιο μικρό χώρο καταστάσεων για ισοδύναμη λειτουργία.
- Χρειάζεται όμως προσοχή γιατί το παραπάνω δεν ισχύει πάντα.

LALR(1)

state 3-6	[X → a · X, a/b/\$] [X → · aX, a/b/\$] [X → · b, a/b/\$]
state 4-7	[X → b · , a/b/\$]
state 8-9	[X → aX·, a/b/\$]

Συστήματα Τύπων

- **Τύπος**
 - Ορίζει ένα σύνολο (*εύρος*) πιθανών τιμών και ένα σύνολο λειτουργιών (*πράξεων*) που έχει νόημα για αυτές

Δομητές Τύπων

- **Απλοί τύποι:** κάθε γλώσσα προγραμματισμού διαθέτει ένα σύνολο προκαθορισμένων τύπων με βάση το οποίο δομούνται όλοι οι υπόλοιποι
- **Δομητές τύπων:** είναι οι τρόποι με τους οποίους μπορούν να συνδυαστούν οι απλοί τύποι για να δημιουργήσουν νέους
- **Καρτεσιανό γινόμενο:** $Z \times Y = \{(z, y) | z \in Z, y \in Y\}$
 - με Z και Y να είναι (απλοί) τύποι
 - ουσιαστικά η διαδικασία που δημιουργεί *structs* και *classes*
- **Ένωση:** ένας τύπος που μπορεί να παίρνει τιμές από το εύρος άλλων (απλών) τύπων
 - **Διακριτή Ένωση:** προσθέτει μία ετκικέτα για κάθε στοιχείο, ώστε να διακρίνεται από ποιό σύνολο προέρχεται
 - **Μη-Διακριτή Ένωση:** κάνει το σύστημα τύπων *μη-ασφαλές*
- **Υποσύνολο:** Δηλώνει πως ένας σύνθετος τύπος κληρονομεί τις πράξεις του από ένα άλλο (απλό) τύπο, αλλά δεν μπορεί να πάρει τιμές από όλο το εύρος του.
 - **Παράδειγμα Ada**

```
sybtype IntDigit_Type is integer range 0 ... 9;
```

- **Πίνακες και Συναρτήσεις:** $f: Z \rightarrow Y$
 - με Z και Y κανονικούς τύπους
 - f ένα πίνακα με δείκτες τύπου Z και τιμές τύπου Y
 - μία αντιστοίχιση τιμών Z σε τιμές Y
 - **Παράδειγμα C**

```
typedef int (*IntFunction) (int);
```

- **Δείκτες και Αναδρομικοί Τύποι:** δημιουργούν το σύνολο όλων των διευθύνσεων, για τον τύπο που προσδιορίζεται
 - **Παράδειγμα C**

```
typedef int* IntPtr;
```

Ισοδυναμία Τύπων

- Πότε μπορούμε να πούμε ότι δύο μεταβκητές έχουν τον ίδιο τύπο;
 - **Δομική Ισοδυναμία:** αν έχουν ίδια δομή, δηλαδή έχουν κατασκευαστεί από τους ίδιους απλούς τύπους, με τους ίδιους δομητές, τότε δύο τύποι είναι ισοδύναμοι
 - **C** σε περιπτώσεις που δεν είναι *struct* ή *union*
 - **Java** σε πίνακες (με ειδικούς κανόνες)
 - **Ονομαστική Ισοδυναμία:** αν δύο τύποι έχουν το ίδιο όνομα, τότε είναι ισοδύναμοι
 - **C** σε *structs* και *unions*
 - **Ada**
 - **Java** σε *classes* και *interfaces*

Θέσεις και Κανόνες Τύπων

Περιβάλλον

- (απλή) **Θέση Τύπου** ονομάζεται μία παράσταση της μορφής

$$e : \tau$$

- και σημαίνει ότι η **έκφραση** e αποτιμάται σε αποτέλεσμα **τύπου** τ
- και έμμεσα ότι η e είναι καλώς ορισμένου τύπου (περνά από τον έλεγχο τύπων)

- **Περιβάλλον**: ένα σύνολο από απλές δηλώσεις τύπων, της μορφής

$$\Gamma = x_1 : \tau_1, x_2 : \tau_2, \dots, x_n : \tau_n$$

- Αν μία μεταβλητή x ορίζεται ως τύπου τ σε ένα περιβάλλον Γ , μπορούμε να γράψουμε $\Gamma \vdash x : \tau$

Θέση Τύπου

- **Θέση Τύπου**: μία παράσταση της μορφής

$$\Gamma \vdash e : \tau$$

- που σημαίνει πως αν όλες οι ελεύθερες μεταβλητές της **έκφρασης** e ορίζονται στο περιβάλλον Γ , τότε η έκφραση e αποτιμάται σε τιμή τύπου τ

Κανόνας Τύπων

- **Κανόνας Τύπων**: μία παράσταση της μορφής

$$\frac{\text{υπόθεση}_1 \dots \text{υπόθεση}_n}{\text{συμπέρασμα}}$$

- που σημαίνει πως αν μπορούν να αποδειχθούν οι υποθέσεις τότε θα ισχύει το συμπέρασμα
- Ο πιο βασικός κανόνας τύπων είναι το

$$\frac{}{\Gamma \vdash id : t} (Id)$$

- Μπορούμε να τον χρησιμοποιούμε αν $id : \tau \in \Gamma$
- λέγεται (Id)
- και ανήκει σε μία ειδική κατηγορία κανόνων που λέγονται **Αξιώματα**, γιατί δεν έχουν υποθέσεις

Παραδείγματα

- το **Axioma**

$$\frac{}{\Gamma \vdash n : int} (Int)$$

- $\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 + e_2 : int} (Add)$

$$\frac{}{\Gamma \vdash \text{true}:bool} (\text{True})$$

$$\frac{}{\Gamma \vdash \text{false}:bool} (\text{False})$$

$$\frac{\Gamma \vdash e : \text{bool} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash (e?e_1 : e_2) : \tau} (\text{If})$$

Επέκταση Περιβάλλοντος

- Σε ένα κανόνα τύπων, μπορεί αντί για περιβάλλοντος Γ να δούμε ένα επεκταμένο περιβάλλοντος

$$\Gamma, x_1 : \tau_1, \dots, x_n : \tau_n$$

που είναι το περιβάλλον Γ μαζί με τους κανόνες $x_i : \tau_i$

- Πιο συγκεκριμένα η έκφραση $\Gamma, x : \tau$ σημαίνει
 - $\Gamma \cup \{x : \tau\}$ όταν $\exists \tau' \neq \tau | x : \tau' \in \Gamma$
 - $\Gamma - \{x : \tau'\} \cup \{x : \tau\}$ όταν $\exists \tau' \neq \tau | x : \tau' \in \Gamma$
- Με άλλα λόγια ο κανόνας που εισάγεται υπερισχύει αν υπάρχει άλλος κανόνας για την μεταβλητή x στο Γ

Παράδειγμα

$$\frac{\Gamma, \text{id} : \tau \vdash e : \tau}{\Gamma, \text{id} : \tau \vdash \text{id} = e : \tau} \text{ (Var - assign)}$$

$$\frac{\Gamma \vdash e_3 : \tau \quad \Gamma \vdash e_2 : \text{int} \quad \Gamma \vdash e_1 : \text{array}[\tau]}{\Gamma \vdash e_1[e_2] = e_3 : \tau} \text{ (Array - assign)}$$

Κανόνες Συνάρτησης

- Μία συνάρτηση ορίζεται $\tau_n f(\tau_1 id_1, \dots, \tau_n id_n)$
 - Ο κανόνας που ελέγχει την εγκυρότητα της δήλωσης μίας συνάρτησης είναι ο
- $$\frac{\Gamma, \text{id}_1 : \tau_1, \dots, \text{id}_k : \tau_k \vdash e : \tau_r}{\Gamma \vdash (\lambda[id_i : \tau_i]_{i=1,\dots,k}. e) : \tau_1 \times \tau_2 \dots \times \tau_k \rightarrow \tau_r} \text{ (Fun - body)}$$
- Όπου
 - Η $(\lambda[id_i : \tau_i]_{i=1,\dots,k}. e)$ είναι μία λ -έκφραση, δηλαδή μία ανώνυμη συνάρτηση με παραμέτρους id_i τύπων τ_i και με σώμα e
 - Ο $\tau_1 \times \tau_2 \dots \times \tau_k \rightarrow \tau_r$ είναι μία αντιστοίχιση (συνάρτηση) από το καρτεσιανό γινόμενο τύπων τ_i σε τύπο τ_r
 - Ο κανόνας που ελέγχει την εγκυρότητα της κλήσης μίας συνάρτησης είναι ο
- $$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \dots \times \tau_k \rightarrow \tau_r \quad \Gamma \vdash [e_i : \tau_i]_{i=1,\dots,k}}{\Gamma \vdash e(e_1, \dots, e_k) : \tau_r} \text{ (Fun - call)}$$

Τύπος void

- Υπάρχουν σε κάθε γλώσσα εκφράσεις που δεν έχουν τιμή, αυτές λέμε ότι είναι τύπου *void* και το συμβολίζουμε με

$$s : \text{void}$$

- Ο τύπος *void* χρησιμοποιείται σε κανόνες που κάνουν έλεγχο τύπου σε εκφράσεις που δεν χαρακτηρίζονται απαραίτητα από την τιμή που επιστρέφουν

Παράδειγμα

$$\frac{\Gamma \vdash e : \text{bool} \quad \Gamma \vdash s : \tau}{\Gamma \vdash \text{while } (e) s : \text{void}} \text{ (While)}$$

$$\frac{\Gamma, \text{return} : \tau_r \vdash e : \tau_r}{\Gamma, \text{return} : \tau_r \vdash \text{return } e : \text{void}} \text{ (Return)}$$

- Ελέγχει πως στην έκφραση "return e" θα το e θα είναι ίδιου τύπου με τον ορισμό της συνάρτησης, αλλά η ίδια η έκφραση δεν αποτιμάται σε κάποιον τύπο

- Κάθε ακολουθία εντολών που συγκροτεί ένα πρόγραμμα θα πρέπει να περνά από έλεγχο τύπων. Θα πρέπει λοιπόν η πρώτη εντολή να είναι καλώς ορισμένου τύπου, καθώς επίσης και οι υπόλοιπες εντολές της ακολουθίας:

$$\frac{\Gamma \vdash s_1 : \tau_1 \quad \Gamma \vdash (s_2; \dots; s_k) : \tau_k}{\Gamma \vdash (s_1; s_2; \dots; s_k) : \tau_k} (\text{Seq})$$

- Δεν καταλαβαίνω τι σημαίνει αυτό

Κλάσεις

- Όταν μία γλώσσα επιτρέπει την δημιουργία κλάσεων, τότε για κάθε κλάση ορίζεται ένα (τοπικό) περιβάλλον

Παράδειγμα

Για την κλάση

```
class C {
    int x,y;
    int get_x() {return x;}
}
```

το περιβάλλον είναι

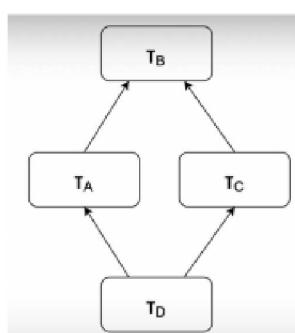
$$\Gamma_C = x: \text{int}, y: \text{int}, \text{get_x}: \text{void} \rightarrow \text{int}$$

Σχέση Υποτύπων

- Επίσης λαμβάνεται υπόψη και ιεραρχία των κλάσεων. Με τη σχέση υποτύπου

$$\tau_A \leq \tau_B$$

ορίζουμε ότι η κλάση τ_A κληρονομεί από την τ_B



Η σχέση υποτύπου έχει τις ιδιότητες:

- ανακλαστική: $\tau_A \leq \tau_A$
 - μεταβατική: αν $\tau_D \leq \tau_A$ και $\tau_A \leq \tau_B$, τότε $\tau_D \leq \tau_B$
 - αντισυμμετρική: αν $\tau_K \leq \tau_L$ και $\tau_L \leq \tau_K$, τότε $\tau_K = \tau_L$
- δηλ. είναι μία σχέση μερικής διάταξης.

- Αυτός ο ορισμός για τη σχέση υποτύπων μπορεί να επεκταθεί σε όλους τους τύπους της γλώσσας (μαζί με τους απλούς)
- Αν τ_A, τ_B είναι τύποι της γλώσσας, και τ_A είναι Απλός Τύπος ή Τύπος Πίνακα, τότε
 - $\tau_A \leq \tau_B \Rightarrow \tau_A = \tau_B$
 - $\tau_B \leq \tau_A \Rightarrow \tau_A = \tau_B$

Παράδειγμα

$$\frac{\Gamma, \text{id} : \tau_1 \vdash e : \tau_2 \quad \tau_2 \leq \tau_1}{\Gamma, \text{id} : \tau_1 \vdash \text{id} = e : \tau_1} (\text{Var - assign})$$

$$\frac{}{\Gamma \vdash \text{new } \tau : \tau} (\text{Con})$$

$$\frac{\Gamma \vdash e : \tau \quad (\text{id}, \tau') \in \Gamma_C}{\Gamma \vdash e.\text{id} : \tau'} (\text{Field-acc})$$

Συμπερασμοί Τύπων

- Όλοι οι κανόνες που ορίζουμε χρησιμοποιούνται για να αποδείξουμε ότι οι τύποι των διαφορετικών εκφράσεων μέσα σε ένα πρόγραμμα είναι έγκυροι

Παράδειγμα

$$\frac{}{\Gamma \vdash n : \text{int}} (\text{Int})$$

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 + e_2 : \text{int}} (\text{Add})$$

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}} (\text{True})$$

$$\frac{}{\Gamma \vdash \text{false} : \text{bool}} (\text{False})$$

όπως
τους
έων

$$\frac{\Gamma \vdash e : \text{bool} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash (e?e_1 : e_2) : \tau} (\text{If})$$

Με τους κανόνες, που έχουμε ορίσει, αν

$$\Gamma = b : \text{bool}, x : \text{int}$$

αποδεικνύεται ότι $\Gamma \vdash (b?2 + 1:x) : \text{int}$

$$\frac{}{\Gamma \vdash b : \text{bool}} (\text{Id}) \quad \frac{\frac{\Gamma \vdash 2 : \text{int}}{(\text{Int})} \quad \frac{\Gamma \vdash 1 : \text{int}}{(\text{Int})}}{\Gamma \vdash 2 + 1 : \text{int}} (\text{Add}) \quad \frac{\Gamma \vdash x : \text{int}}{(\text{Id})} (\text{If})$$

$$\frac{}{\Gamma \vdash (b?2 + 1 : x) : \text{int}}$$

δηλ. εφαρμόζεται ο κανόνας (If) για $\tau = \text{int}$.