ΚΕΦΑΛΑΙΟ 4 ΒΑΣΙΚΕΣ ΤΕΧΝΙΚΕΣ ΣΥΝΤΑΚΤΙΚΗΣ ΑΝΑΛΥΣΗΣ

Στόχος

Στόχος του Κεφαλαίου αυτού είναι να μάθουμε τις βασικότερες από τις τεχνικές και τις μεθοδολογίες συντακτικής ανάλυσης των κατηγοριών bottom up και top down που χρησιμοποιούνται σε μεταγλωττιστές

Προσδοκώμενα αποτελέσματα

Όταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να

- Εξηγήσετε τις διαφορές στη λειτουργία μεταξύ των μεθόδων bottom up και top down,
- Εξηγήσετε αν μια γραμματική είναι κατάλληλη για ένα συντακτικό αναλυτή της κατηγορίας bottom up και top down,
- Κατασκευάσετε ένα συντακτικό αναλυτή του τύπου operator precedence,
- Μετατρέψετε μια γραμματική ώστε να είναι κατάλληλη για ένα αναλυτή τύπου recursive descent και LL(1) χωρίς οπισθοανίχνευση,
- Κατασκευάσετε ένα συντακτικό αναλυτή του τύπου recursive descent,
- Περιγράψετε τον τρόπο λειτουργίας ενός αναλυτή τύπου LL(1)

Έννοιες κλειδιά

bottom up και top down ανάλυση, shift-reduce, οπισθοανίχνευση, operator precedence, recursive descent, LL(1),

Εισαγωγικές παρατηρήσεις

Είδαμε στο προηγούμενο κεφάλαιο πως μια CF-γραμματική μπορεί να χρησιμοποιηθεί για να ορίσει την σύνταξη μιας γλώσσας προγραμματισμού. Τώρα θα δούμε **πώς να ελέγξουμε ότι μια συμβολοσειρά είναι μια πρόταση μιας δοθείσης γραμματικής** και πώς, αν χρειάζεται, να δημιουργήσουμε ένα δένδρο ανίχνευσης για την συμβολοσειρά.

Στο κεφάλαιο αυτό θα υποθέσουμε ότι η είσοδος του Συντακτικού Αναλυτή είναι μια ακολουθία από tokens τα οποία παραδίδονται το ένα μετά το άλλο από τον Λεξικό Αναλυτή.

Θα συζητήσουμε κατ' αρχήν στην ενότητα 4.2 μια bottom-up μέθοδο που καλείται μετακίνησε-μείωσε ("shift-reduce") διότι μετακινεί (shift) τα σύμβολα εισόδου σε μια στοίβα (Stack) μέχρις ότου το δεξιό μέρος ενός κανόνα εμφανιστεί στην κορυφή της στοίβας. Αυτό κατόπιν αντικαθίσταται -μειώνεται (reduced)- από το μη τερματικό σύμβολο στα αριστερά του κανόνα. Στη συνέχεια

στην ενότητα 4.3 θα μελετήσουμε την μέθοδο Operator Precedence και θα δούμε πώς μπορούμε να κατασκευάσουμε ένα τέτοιο συντακτικό αναλυτή. Στην ενότητα 4.4 θα μελετήσουμε τα προβλήματα που εμφανίζονται στην κατηγορία top down και τρόπους αντιμετώπισης τους. Τέλος στις ενότητες 4.5 και 4.6 θα μελετήσουμε δυο τρόπους υλοποίησης της κατηγορίας top down δηλαδή τις μεθόδους Recursive Descent και LL(1) ή Predictive Parsing.

ΕΝΟΤΗΤΑ 4.1 ΕΙΣΑΓΩΓΗ ΣΕ ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ

Οι δυο πιο κοινές μορφές συντακτικής ανάλυσης είναι η **Operator Precedence** και η **Recursive Descent (Αναδρομική Κατάβαση).** Η πρώτη είναι κατάλληλη κυρίως για αριθμητικές εκφράσεις, η δε δεύτερη χρησιμοποιεί μια συλλογή από αναδρομικές ρουτίνες για να κάνει την Συντακτική Ανάλυση εφόσον η γραμματική η ίδια είναι αναδρομική.

Σε πολλούς Συντακτικούς Αναλυτές χρησιμοποιείται Operator Precedence ανίχνευση για τις εκφράσεις και Αναδρομική Κατάβαση για την υπόλοιπη γλώσσα.

Δυο νεώτερες μέθοδοι έχουν γίνει δημοφιλείς, κυρίως σε συνδυασμό με κάποια εργαλεία (βοηθήματα) που τις κάνουν ευκολόχρηστες. Η πρώτη από αυτές, **LL** ανίχνευση, είναι μια παραλλαγή της Αναδρομικής Κατάβασης που βασίζεται σε πίνακες. Η δεύτερη καλείται **LR** και επίσης βασίζεται σε πίνακες. Η δημιουργία των δένδρων ανίχνευσης -πραγματική ή νοητή- μπορεί να γίνει είτε από κάτω προς τα επάνω (Bottom-up) είτε από επάνω προς τα κάτω (Top-Down). Η μέθοδος LR που αναφέραμε παραπάνω είναι μια Bottom-up, όπως είναι και η Operator Precedence. Η Αναδρομική Κατάβαση όπως και η LL είναι Top-Down μέθοδοι.

Top-Down και Bottom-up είναι λοιπόν οι δύο βασικοί τρόποι που χρησιμοποιούμε για να κάνουμε συντακτική ανίχνευση μιας πρότασης μιας γλώσσας.

ΕΝΟΤΗΤΑ 4.2 SHIFT-REDUCE ΑΝΙΧΝΕΥΣΗ

Είναι μια bottom-up μέθοδος που δημιουργεί το δένδρο ανίχνευσης από μια συμβολοσειρά εισόδου w και προχωρεί προς την ρίζα. Ας υποθέσουμε για παράδειγμα ότι μας δίνεται η γραμματική

$S \rightarrow aAcBe$
$A \rightarrow Ab|b$

B→d

και η συμβολοσειρά εισόδου w=abbcde, και ότι θέλουμε να μειώσουμε το w στο αρχικό σύμβολο S. Η διαδικασία διακρίνεται στις εξής μειώσεις:

abbcde=>aAbcde=>aAcde=>aAcBe=>S.

Αν είχαμε αντικαταστήσει το b με το A στο aAbcde θα βρίσκαμε aAAcde το οποίο δεν μπορούμε να μειώσουμε σε S. Χρειάζεται λοιπόν να ορίσουμε το **handle**. Απλά (!) λέμε ότι, μια υπο-συμβολοσειρά η οποία είναι το δεξί μέρος ενός κανόνα, τέτοια ώστε η αντικατάστασή της από το αριστερό μέρος του κανόνα, **τελικά οδηγεί** σε μια μείωση στο αρχικό σύμβολο, **με την αντίστροφη διαδικασία μιας δεξιότερης παραγωγής, καλείται handle!!** Ηιο τυπικά βέβαια μπορούμε να δώσουμε τον εξής **ορισμό**

Handle μιας δεξιά-προτασιακής μορφής γ **είναι** ένας κανόνας A→β και μια θέση της γ όπου η συμβολοσειρά β μπορεί να ευρεθεί και αντικατασταθεί από το A για να δημιουργήσει την προηγούμενη δεξιά-προτασιακή μορφή, σε μια δεξιότερη παραγωγή του γ.

Δεξιά-προτασιακή μορφή είναι μία προτασιακή μορφή που δημιουργείται από μια δεξιά παραγωγή όπως αυτή έχει περιγραφεί στην ενότητα 3.2.

Παράδειγμα 1 / Κεφ. 4

Έστω η γραμματική:

(1) E→E + E

(2) E→E * E (2)

(3) E→(E)

(4) E→**id**

Παρατηρήστε την παρακάτω δεξιότερη παραγωγή

E => <u>E+E</u> => E+<u>E*E</u> => E+E*<u>id</u>$_3$ => E+<u>id</u>$_2$*id$_3$ =><u>id</u>$_1$+id$_2$*id$_3$

Το handle σε κάθε δεξιά-προτασιακή μορφή είναι υπογραμμισμένο.

Η γραμματική (2) είναι βέβαια διφορούμενη. Έτσι, αν ξεκινήσουμε από τον κανόνα E=>E*E δημιουργούνται άλλα handles. Σημειώνουμε ότι δεν αναφέρουμε ακόμα πώς βρίσκουμε τα handles, αυτό θα γίνει στην ενότητα 4.3.

Η υλοποίηση της shift-reduce ανίχνευσης μπορεί να πραγματοποιηθεί εύκολα με την βοήθεια μιας στοίβας (stack). Δύο προβλήματα εμφανίζονται εδώ. Πώς να εντοπίσουμε ένα handle σε μια δεξιά-προτασιακή μορφή, και ποιόν κανόνα να διαλέξουμε στην περίπτωση που υπάρχουν περισσότεροι από ένας.

Χρησιμοποιούμε μια στοίβα και ένα buffer που περιέχει τη συμβολοσειρά εισόδου. Το σύμβολο $| σημειώνει το τέλος της στοίβας και το δεξιό τέλος του buffer.

Stack	Input Buffer
$|	W $|

Ο shift-reduce Συντακτικός Αναλυτής λειτουργεί μεταφέροντας μηδέν ή περισσότερα σύμβολα στη στοίβα, μέχρις ότου ένα handle β σχηματισθεί στην κορφή της. Κατόπιν μειώνει το β στο αριστερό μέρος του κατάλληλου κανόνα. Η διαδικασία επαναλαμβάνεται έως ότου ανακαλυφθεί λάθος ή η στοίβα περιέχει το αρχικό σύμβολο της γραμματικής και ο Input buffer είναι άδειος.

Stack	Input Buffer
$| S	$|

Παράδειγμα 2 / Κεφ. 4

Ας δούμε στο Σχήμα 4.1 βήμα-βήμα την λειτουργία του shift-reduce Αναλυτή ακολουθώντας την γραμματική (2) και συμβολοσειρά εισόδου την $id_1+id_2*id_3$.

	Stack	Input	Δράση
(1)	$|	$id_1+ id_2 * id_3$|	μετέφερε
(2)	$| id_1	$+ id_2 * id_3$|	μείωσε με E→id
(3)	$| E	$+ id_2 * id_3$|	μετέφερε
(4)	$| E +	$id_2 * id_3$|	μετέφερε
(5)	$| E + id_2	$* id_3$|	μείωσε με E→id
(6)	$| E +E	$* id_3$|	μετέφερε
(7)	$| E + E *	id_3|	μετέφερε
(8)	$| E + E * id_3	$|	μείωσε με E→id
(9)	$| E + E * E	$|	μείωσε με E→E*E
(10)	$| E + E	$|	μείωσε με E→E+E
(11)	$| E	$|	accept

Σχήμα 4.1 Η λειτουργία του shift-reduce Αναλυτή με είσοδο $id_1+id_2*id_3$

Αν και οι βασικές λειτουργίες του Αναλυτή είναι shift και reduce, υπάρχουν ακόμα δύο, **accept** και **error**. Η λειτουργία accept αναφέρεται σε επιτυχή αναγνώριση της συμβολοσειράς εισόδου από τον αναλυτή. Η δε λειτουργία error αντιστοιχεί σε κλήση ρουτίνας διώρθωσης ή αναφοράς στντακτικού λάθους.

ΕΝΟΤΗΤΑ 4.3 OPERATOR PRECEDENCE ΑΝΙΧΝΕΥΣΗ

Για ορισμένη μικρή κλάση γραμματικών μπορούμε εύκολα να κατασκευάσουμε με το χέρι, ένα shift-reduce Αναλυτή. Οι γραμματικές αυτές δεν έχουν κανόνα με δεξιό μέρος ε (κενό) ή δυο συνεχόμενα Μη Τερματικά, και καλούνται Operator Γραμματικές.

Παράδειγμα 3 / Κεφ. 4

Από τις παρακάτω γραμματικές η πρώτη δεν είναι Operator Γραμματική, ενώ η δεύτερη είναι Operator Γραμματική.

(1) $E \rightarrow EAE \mid (E) \mid - E \mid id$
$A \rightarrow + \mid - \mid * \mid / \mid \uparrow$

(2) $E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid E \uparrow E \mid (E) \mid - E \mid id$

Σε **Operator Precedence ανίχνευση** χρησιμοποιούμε τρεις σχέσεις προτεραιότητας μεταξύ των τερματικών συμβόλων. Οι σχέσεις αυτές κατευθύνουν την επιλογή των handles, και είναι οι : $<^{\bullet}$, $=^{\bullet}$, και $^{\bullet}>$. Η σχέση $a <^{\bullet} b$ δηλώνει ότι "το a δίνει προτεραιότητα στο b", η $a=^{\bullet}b$ ότι "έχουν την ίδια προτεραιότητα", και η σχέση $a^{\bullet}>b$ ότι "το a έχει προτεραιότητα από το b". Οι σχέσεις αυτές δεν έχουν καμιά σχέση με τα <, = ,> των αριθμητικών η λογικών εκφράσεων. Είναι δυνατόν σε μια γλώσσα να έχουμε: $a <^{\bullet} b$ και $a ^{\bullet}> b$, ή ακόμη και καμιά από τις $^{\bullet}>$, $=^{\bullet}$, $<^{\bullet}$ανάμεσα σε δυο τερματικά a και b.

Οι σχέσεις $^{\bullet}>$, $=^{\bullet}$, $<^{\bullet}$ορίζονται πιο τυπικά από τους παρακάτω κανόνες.

1) $a =^{\bullet} b$ αν υπάρχει κανόνας με δεξιό μέρος το **αaβbγ**, όπου β είναι είτε ε (κενό) είτε ένα Μη Τερματικό. Δηλαδή $a =^{\bullet} b$ αν είναι γειτονικά στο δεξί μέρος ενός κανόνα ή χωρίζονται από ένα Μη Τερματικό.

2) $a <^{\bullet} b$ αν για κάποιο Μη Τερματικό A υπάρχει κανόνας με δεξί μέρος της μορφής **αaAβ**, και $A \Rightarrow^{+} \gamma b \delta$, όπου το γ είναι είτε ε είτε μοναδικό Μη Τερματικό.

3) $a ^{\bullet}> b$ αν για κάποιο Μη Τερματικό A υπάρχει κανόνας με δεξί μέρος της μορφής **αAbβ**, και $A \Rightarrow^{+} \gamma a \delta$, όπου δ είναι είτε ε είτε μοναδικό Μη Τερματικό. Επιπλέον ειδικά για το σύμβολο $| ορίζεται ότι, $a ^{\bullet}> \$|$ αν $S \Rightarrow^{+} \gamma a \delta$ και το δ είναι είτε ε είτε μοναδικό Μη Τερματικό, όπως και ότι $\$| <^{\bullet} b$ αν $S \Rightarrow^{+} \gamma b \delta$ και γ είναι ε ή μοναδικό Μη Τερματικό.

Ορισμός. Operator-Precedence (O-P) γραμματική είναι μια operator γραμματική στην οποία για οποιοδήποτε ζευγάρι τερματικών συμβόλων a και b μόνο μια από τις σχέσεις $a \stackrel{\bullet}{>} b$, $a \stackrel{\bullet}{=} b$, και $a \stackrel{\bullet}{<} b$ είναι αληθής.

Δραστηριότητα 1 / Κεφ. 4

Σας δίνεται η γραμματική (5), και σας ζητούνται τα παρακάτω

$$E \rightarrow E+E \mid E*E \mid \mathbf{(}E\mathbf{)} \mid \mathbf{id} \qquad (5)$$

(α) Προσπαθήστε καταρχήν να δείξετε ότι η γραμματική αυτή δεν είναι Operator-Precedence γραμματική (αυτό μπορείτε να το διαπιστώσετε για παράδειγμα, εφαρμόζοντας τους ορισμούς των κανόνων προτεραιότητας, και βλέποντας ότι ισχύει $+ \stackrel{\bullet}{>} +$ και $+ \stackrel{\bullet}{<} +$ ταυτόχρονα. Παρατηρώντας την γραμματική θα πρέπει να δείτε ότι είναι διφορούμενη).

(β) Στη συνέχεια μετασχηματίστε την γραμματική σε Operator Precedence και μη-διφορούμενη εφαρμόζοντας τους κανόνες αποδιφοροποίησης που μάθατε στο κεφάλαιο 3.

(γ) Τέλος, κατασκευάστε τον πίνακα των σχέσεων προτεραιότητας μεταξύ όλων των τερματικών συμβόλων της γραμματικής.

Προσπαθήστε χωρίς να κοιτάξετε τις απαντήσεις που ακολουθούν στη συνέχεια.

(α) Η γραμματική (5) είναι Operator γραμματική αλλά δεν είναι Operator-Precedence γραμματική διότι μπορούμε να διαπιστώσουμε ότι ισχύουν δύο σχέσεις προτεραιότητας μεταξύ + και +. Συγκεκριμένα από τον κανόνα (3) αν αAbβ είναι το E+E (δηλ. α = ε, A = E, b = +, και β = E), το δε A => γaδ θα μπορούσε να είναι το E => E+E (όπου γ=E, a=+, και δ=E), τότε $+ \stackrel{\bullet}{>} +$. Αν χρησιμοποιήσουμε τον κανόνα (2) για τις ίδιες συμβολοσειρές θα διαπιστώσουμε ότι $+ \stackrel{\bullet}{<} +$.

(β) Η (5) όμως μπορεί να μετασχηματισθεί στην παρακάτω γραμματική η οποία είναι Operator Precedence και μη-διφορούμενη.

$$\begin{aligned} &E \rightarrow E + T \mid T \\ &T \rightarrow T * F \mid F \\ &F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id} \end{aligned} \qquad (6)$$

(γ) Θα κατασκευάσουμε τώρα τον **πίνακα των σχέσεων προτεραιότητας** όλων των τερματικών συμβόλων για την γραμματική (6). Ο πίνακας αυτός είναι εκείνος που θα κατευθύνει τον Συντακτικό Αναλυτή στην επιλογή των handles.

Κατ' αρχήν βρίσκουμε για κάθε μη τερματικό, εκείνα τα τερματικά που μπορούν να είναι πρώτα ή τελευταία τερματικά σε μια συμβολοσειρά η οποία παράγεται από αυτό το μη τερματικό. Παρατηρούμε ότι όλες οι παραγωγές από το F θα έχουν τα σύμβολα (ή id σαν πρώτο τερματικό, και τα σύμβολα) ή id σαν τελευταίο. Μια παραγωγή από το T μπορεί να αρχίζει T => T*F, που δείχνει ότι το * μπορεί να είναι και πρώτο και τελευταίο τερματικό σύμβολο από τις παραγωγές του T. Ακόμη, μια παραγωγή του T μπορεί να αρχίζει με T => F, που δείχνει ότι κάθε πρώτο ή τελευταίο τερματικό που παράγεται από το F, είναι επίσης ένα πρώτο ή τελευταίο τερματικό που παράγεται από το T. Έτσι το T έχει τα *, (, id σαν πρώτα και τα *,), id σαν τελευταία τερματικά σε μια παραγωγή του. Παρόμοια για το E βρίσκουμε ότι τα +, *, (, id μπορούν να είναι πρώτα και τα *, +,), id τελευταία τερματικά που προκύπτουν από τις παραγωγές του E.

Έχουμε λοιπόν τον πίνακα του Σχήματος 4.2,

Μη Τερματικό	Πρώτα Τερματικά	Τελευταία Τερματικά
E	*, +, (, id	*, +,), id
T	*, (, id	*,), id
F	(, id	), id

Σχήμα 4.2 Πίνακας με τα πρώτα και τελευταία τερματικά σύμβολα των E, T και F

Για να υπολογίσουμε τώρα την σχέση $=^{\bullet}$, ψάχνουμε τα δεξιά μέλη της (6) για δυο τερματικά σύμβολα με ένα μη τερματικό ή τίποτα στην μέση. Βρίσκουμε λοιπόν από τον κανόνα $F \rightarrow$ **(** E **)** ότι ($=^{\bullet}$).

Κατόπιν εξετάζουμε την σχέση $<^{\bullet}$. Ψάχνουμε τα δεξιά μέρη των κανόνων για ένα τερματικό στο αριστερό μέρος ενός μη τερματικού για να παίξουν τους ρόλους των a και A του κανόνα (2). Για κάθε τέτοιο ζευγάρι, το a σχετίζεται με $<^{\bullet}$. με κάθε τερματικό που μπορεί να είναι πρώτο σε κάποια συμβολοσειρά παραγόμενη από το A. Στην (6) εξετάζουμε τα + και T στο δεξί μέρος E+T, τα * και F στο T*F, και τα (και E στο (E). Το ζευγάρι +:T δίνει + $<^{\bullet}$ *, + $<^{\bullet}$ (, και + $<^{\bullet}$ id. Το ζευγάρι *:F δίνει * $<^{\bullet}$ (, και * $<^{\bullet}$ id. Το ζευγάρι (:E δίνει ($<^{\bullet}$* , ($<^{\bullet}$ +, ($<^{\bullet}$ C, και ($<^{\bullet}$ id. Τέλος προσθέτουμε τις σχέσεις $| $<^{\bullet}$ *, $| $<^{\bullet}$ +, $| $<^{\bullet}$ (, και $| $<^{\bullet}$ id, μια και το $| πρέπει να συνδέεται με την σχέση $<^{\bullet}$ με όλα τα δυνατά πρώτα τερματικά παραγόμενα από το αρχικό σύμβολο E.

Συμμετρικά κατασκευάζουμε την σχέση $\bullet>$. Ψάχνουμε τις δεξιές πλευρές για ένα μη τερματικό στα αριστερά ενός τερματικού, για να παίξουν τους ρόλους των A και b του κανόνα (3). Τότε κάθε τερματικό που μπορεί να είναι τελευταίο σε μια συμβολοσειρά που παράγεται από το A, σχετίζεται με $\bullet>$ με το b. Έτσι στην (6) τα ζευγάρια που αντιστοιχούν στα A και b είναι E:+, T:*, και E:).

Άρα θα έχουμε τις σχέσεις $*\bullet>+$, $+\bullet>+$, $)\bullet>+$, $id\bullet>+$, $*\bullet>*$, $)\bullet>*$, $id\bullet>*$, $*\bullet>)$, $+\bullet>)$, $)\bullet>)$, και $id\bullet>)$.

Ακόμη προσθέτουμε τις σχέσεις $*\bullet>\$$, $+\bullet>\$$, $)\bullet>\$$, και $id\bullet>\$$, σύμφωνα με τον κανόνα (3). Έχουμε λοιπόν τον πίνακα του Σχήματος 4.3 .

	+	*	(	)	id	$
+	$\bullet>$	$<\bullet$	$<\bullet$	$\bullet>$	$<\bullet$	$\bullet>$
*	$\bullet>$	$\bullet>$	$<\bullet$	$\bullet>$	$<\bullet$	$\bullet>$
(	$<\bullet$	$<\bullet$	$<\bullet$	=	$<\bullet$	e_4
)	$\bullet>$	$\bullet>$	e_3	$\bullet>$	e_3	$\bullet>$
id	$\bullet>$	$\bullet>$	e_3	$\bullet>$	e_3	$\bullet>$
$	$<\bullet$	$<\bullet$	$<\bullet$	e_2	$<\bullet$	e_1

Σχήμα 4.3 Ο πίνακας προτεραιοτήτων της O-P γραμματικής (6).

Παρατηρώντας την μεθοδολογία του παραπάνω παραδείγματος βλέπουμε ότι κατασκευάσαμε κατ' αρχήν δυο σύνολα LEADING (A) και TRAILLING (A) για κάθε μη τερματικό A, οριζόμενα από τις σχέσεις:

LEADING(A) = {a|A => γaδ, όπου γ είναι ε ή μοναδικό μη τερματικό}

TRAILLING(A) = {a|A => γaδ, όπου δ είναι ε ή μοναδικό μη τερματικό}

Ακολούθως καθορίσαμε την σχέση $<\bullet$, κοιτάζοντας στα δεξιά μέρη για συνεχόμενα σύμβολα ...aA..., και σχετίσαμε το $a<\bullet b$ για κάθε b στο σύνολο LEADING(A). Παρόμοια μας βοήθησαν τα TRAILLING σύνολα για να καθορίσουμε τις σχέσεις $\bullet>$.

Άσκηση αυτοαξιολόγησης 1 / Κεφ 4

Στον πίνακα προτεραιοτήτων του Σχήματος 4.3 υπάρχουν θέσεις με περιεχόμενα τους δείκτες e1, e2, e3 και e4. Οι δείκτες αυτοί αναφέρονται σε

ρουτίνες διόρθωσης λαθών μια και οι σχετικές θέσεις του πίνακα υποδηλώνουν συντακτικά λάθη. Προσπαθήστε να γράψετε μικρά κομμάτια ψευδοκώδικα που θα μπορούσε να χρησιμοποιήσει ο συντακτικός αναλυτής για να εκδώσει σχετικά διαγνωστικά μηνύματα και να διορθώσει τα αντίστοιχα λάθη.

Τους αλγόριθμους για την αυτόματη κατασκευή των συνόλων LEADING και TRAILLING και την κατασκευή του πίνακα προτεραιοτήτων μπορείτε να τους βρείτε στην βιβλιογραφία (πχ [2,3]). Εμείς τώρα θα δώσουμε ένα αλγόριθμο shift-reduce που θα ανιχνεύσει όλες τις προτάσεις (sentences) και μόνο προτάσεις.

Αλγόριθμος

Είσοδος: Οι σχέσεις προτεραιότητας (ο πίνακας), μιας O-P γραμματικής και μια συμβολοσειρά εισόδου από τερματικά σύμβολα της γραμματικής.

Έξοδος: Ουσιαστικά δεν υπάρχει έξοδος. Αλλιώς μπορούμε να θεωρήσουμε σαν έξοδο αυτή την ίδια την ακολουθία των βημάτων της shift-reduce.

Μέθοδος: Έστω a1 a2 ...an $ το input string. Αρχικά η στοίβα περιέχει $. Να εκτελεσθεί το πρόγραμμα του Σχήματος 4.4

.

```
repeat forever
   (1) if μόνο $ είναι στη στοίβα και στην είσοδο
   (2) then accept and break
       else
         begin
   (3)     έστω a το κορυφαίο τερματικό σύμβολο στη στοίβα
           και b το τρέχον σύμβολο στην είσοδο;
   (4)     if a <• b or a =• b
              then μετακίνησε το  b στην κορυφή της στοίβας
   (5)        else if a •> b      then    /* μείωσε */
   (6)           repeat  βγάλε το κορυφαίο στοιχείο από την στοίβα
   (7)           until το κορυφαίο τερματικό στοιχείο της στοίβας έχει την σχέση
                        <•  με το τερματικό που εξήχθη τελευταίο
   (8)                        else κάλεσε ρουτίνα διόρθωσης λαθών
         end /*begin*/
```

Σχήμα 4.4 Πρόγραμμα πυρήνας για O-P ανίχνευση.

Εάν θέλουμε να δημιουργήσουμε ένα δένδρο ανίχνευσης καθώς προχωράει η συντακτική ανάλυση, τότε πρέπει να δημιουργούμε ένα κόμβο για κάθε τερματικό που μεταφέρεται στη στοίβα στην γραμμή (4). Κατόπιν, όταν η ανακύκλωση των γραμμών (6)-(7) μειώνει με κάποιο κανόνα, δημιουργούμε ένα

κόμβο του οποίου παιδιά είναι οι κόμβοι που αντιστοιχούν στο οτιδήποτε εξήχθη από τη στοίβα. Μετά την γραμμή (7) τοποθετούμε στη στοίβα ένα δείκτη στον κόμβο που δημιουργήθηκε. Αυτό σημαίνει ότι μερικά από τα "σύμβολα" που εξήχθηκαν από την γραμμή (6) θα είναι δείκτες σε κόμβους. Η σύγκριση της γραμμής (7) συνεχίζεται να γίνεται μόνο μεταξύ τερματικών, οι δείκτες εξάγονται χωρίς καμιά σύγκριση.

ΕΝΟΤΗΤΑ 4.4 TOP-DOWN ΑΝΙΧΝΕΥΣΗ

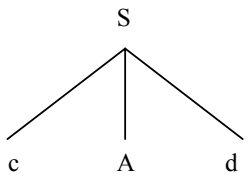
Κατ' αρχήν θα αναφερθούμε σε μια γενική μορφή Top-Down ανίχνευσης η οποία χρειάζεται οπισθο-ανίχνευση (backtracking), και στην συνέχεια θα αναφερθούμε στη μέθοδο Αναδρομικής Κατάβασης η οποία καταργεί την ανάγκη για backtracking. Η Top-Down ανίχνευση μπορεί να θεωρηθεί σαν προσπάθεια προσδιορισμού μιας αριστερότερης παραγωγής (leftmost derivation) μιας συμβολοσειράς εισόδου, ή ακόμη και σαν προσπάθεια κατασκευής ενός δένδρου ανίχνευσης για την συμβολοσειρά εισόδου αρχίζοντας από την ρίζα και δημιουργώντας του κόμβους σε preorder διαπέραση (traversal).

Ας πάρουμε την γραμματική:

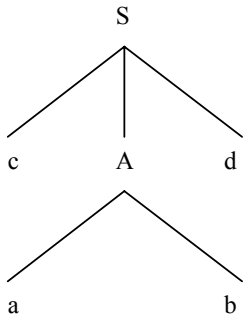
$S \rightarrow cAd$ (7)

$A \rightarrow ab|a$

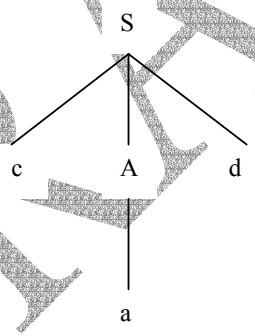
και την συμβολοσειρά εισόδου w=cad. Κάνουμε χρήση ενός δείκτη στο w, αρχίζοντας από το σύμβολο c. Στην αρχή φτιάχνουμε ένα δένδρο με ένα μόνο κόμβο που επιγράφεται από το S και στη συνέχεια χρησιμοποιούμε τον πρώτο κανόνα της (7) για να αναπτύξουμε το δένδρο:



Το c συμπίπτει με το πρώτο σύμβολο του w, άρα προχωράμε τον δείκτη στο a που είναι το δεύτερο σύμβολο του w, και θεωρούμε το επόμενο φύλλο που είναι το A. Αναπτύσσουμε το A σύμφωνα με τον πρώτο εναλλακτικό κανόνα του A, και έχουμε:



Το δεύτερο σύμβολο a του w συμπίπτει με το a του αναπτύγματος του A. Κατόπιν κοιτάζουμε το επόμενο σύμβολο d του w και το οποίο δεν συμπίπτει με το b. Αναφέρουμε την αποτυχία και ψάχνουμε για επόμενο εναλλακτικό κανόνα του A, ο οποίος θα μπορούσε να μας δώσει ταυτότητα μεταξύ w και των φύλλων του δένδρου. Πηγαίνοντας προς τα πίσω (backtracking) πρέπει να επαναφέρουμε τον δείκτη του w στην θέση 2, την οποία είχε όταν πήγαμε για πρώτη φορά στο A. Κάνουμε λοιπόν χρήση του κανόνα $A \rightarrow a$ τώρα και έχουμε ταυτότητα. Το τελικό δένδρο είναι το:



Για να υλοποιήσουμε ένα τέτοιο Συντακτικό Αναλυτή μπορούμε να δημιουργήσουμε μια ρουτίνα για κάθε μη τερματικό σύμβολο. Οι ρουτίνες αυτές είναι κατά κανόνα αναδρομικές. Στην περίπτωση της (7), η οποία δημιουργεί δυο μόνο συμβολοσειρές, δεν υπάρχει ανάγκη για αναδρομή.

Παράδειγμα 4 / Κεφ.4

Σαν παράδειγμα δίνουμε σε μορφή ψευδοκώδικα στο Σχήμα 4.5 παρακάτω τις ρουτίνες για τους κανόνες S και A της γραμματικής (7)

```
proc S( ); /*  S → cAd  */
begin
        if in_sym ='c'
        then begin
                advance ( );
```

```
                if A ( )
                then
                    if in_sym ='d'
                    then begin
                            advance ( );
                            return true
                            end;
                end;
        return false
end;

proc A ( );
begin       /* φύλαξε τον δείκτη για οπισθοανίχνευση */
            save_pointer := in_pointer;
            if in_sym = 'a'
    then begin advance ( );
                if in_sym ='b'
                then begin advance ( ); return true end
            end;
    /* δοκίμασε εναλλακτικό κανόνα αφού πρώτα ακυρώσεις τις σημασιολογικές
    εργασίες του προηγούμενου κανόνα */
    in_pointer := save_pointer;
    if in_sym = 'a'
    then begin advance ( ); return true end
    else return false
end.
```

Σχήμα 4.5 Οι ρουτίνες της γραμματικής (7) για Top-Down ανίχνευση.

Η ρουτίνα advance() αντιστοιχεί στην κλήση του Λεκτικού Αναλυτή και προχωράει τον δείκτη του w στο επόμενο σύμβολο εισόδου. Οι ρουτίνες S και A επιστρέφουν **true** ή **false** ανάλογα με το αν βρήκαν στη συμβολοσειρά εισόδου ένα συμβολοσειρά που παράγεται από το αντίστοιχο μη τερματικό. Παρατηρούμε ότι σε περίπτωση αποτυχίας (false), κάθε ρουτίνα αφήνει τον δείκτη εισόδου όπου ήταν όταν κλήθηκε η ρουτίνα και σε περίπτωση επιτυχίας (true) μετακινεί τον δείκτη.

4.4.1 ΠΡΟΒΛΗΜΑΤΑ ΣΕ TOP-DOWN ΑΝΙΧΝΕΥΣΗ

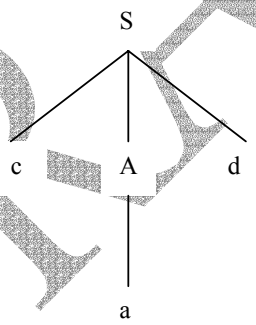
Τρία είναι τα βασικά προβλήματα που συναντάμε σε top-down ανίχνευση.

Το πρώτο αναφέρεται στην αριστερή αναδρομή (left-recursion). Μια γραμματική G λέγεται **αριστερά αναδρομική** αν έχει ένα μη τερματικό A τέτοιο ώστε να υπάρχει μια παραγωγή A => Aα για κάποιο α (δες γραμματική (8) παρακάτω). Μια αριστερά αναδρομική γραμματική μπορεί να αναγκάσει έναν

top-down αναλυτή να μπει σε μια ατελείωτη ανακύκλωση. Αυτό μπορεί να συμβεί σε λανθασμένη συμβολοσειρά εισόδου αλλά ακόμη και σε σωστή, ανάλογα με ποια σειρά θα δοκιμασθούν οι εναλλακτικοί κανόνες του Α. Στην περίπτωση λοιπόν του top-down αναλυτή πρέπει να εξαλείψουμε την αριστερή αναδρομή από την γραμματική (δες την υποενότητα 4.4.2).

Ένα δεύτερο πρόβλημα αναφέρεται στο backtracking και στις σημασιολογικές εργασίες που έγιναν πριν εντοπισθεί ότι πρέπει να γίνει backtracking. Παραδείγματος χάριν, θα πρέπει να βγάλουμε από τον πίνακα συμβόλων πληροφορίες που βάλαμε προηγουμένως. Μια και η δουλειά αυτή είναι χρονοβόρα, είναι λογικό να ψάχνουμε για top-down ανιχνευτές που δεν κάνουν backtracking (π.χ. αναδρομικής κατάβασης και predictive parsers).

Τρίτο πρόβλημα είναι ότι η σειρά με την οποία δοκιμάζονται οι εναλλακτικοί κανόνες μπορεί να έχει συνέπειες στην γλώσσα. Π.χ. αν στην γραμματική (7), χρησιμοποιούσαμε a και κατόπιν ab σαν την σειρά των εναλλακτικών του Α, μπορεί να αποτυγχάναμε στην αναγνώριση της συμβολοσειράς **cabd**, ενώ θα είχαμε επιτυχία στην περίπτωση της cad.



Δηλαδή με το δένδρο και το ca να έχει ήδη αναγνωρισθεί, η αποτυχία ταύτισης του επόμενου συμβόλου, b, υπονοεί ότι το εναλλακτικό cAd του S ήταν εσφαλμένο, και οδηγεί στην απόρριψη του cabd (διότι κάνει backtrack και πηγαίνει στο προηγούμενο φύλλο (c) για να εφαρμόσει εναλλακτικό κανόνα, άρα γυρίζει στο S το οποίο όμως δεν έχει εναλλακτικό κανόνα).

4.4.2 ΑΠΑΛΟΙΦΗ ΤΗΣ ΑΜΕΣΗΣ ΑΡΙΣΤΕΡΗΣ ΑΝΑΔΡΟΜΗΣ

Αν έχουμε τους κανόνες $A \rightarrow A\alpha \mid \beta$ όπου το β δεν αρχίζει με Α, τότε απαλείφουμε την αριστερή αναδρομή ξαναγράφοντας τους κανόνες ως εξής.

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' | \varepsilon$

Άσκηση αυτοαξιολόγησης 2 / κεφ 4

Μπορείτε να δείξετε ότι ο παραπάνω μετασχηματισμός δεν αλλάζει την γραμματική; Υπόδειξη: πρέπει να δείξετε ότι οι συμβολοσειρές που παράγονται από τον αρχικό και τον μετασχηματισμένο κανόνα είναι ίδιες.

Παράδειγμα 5 / Κεφ.4

Έστω η γραμματική (8) παρακάτω η οποία μετασχηματίζεται στην (9) απαλείφοντας τις άμεσες αριστερές αναδρομές της μορφής $A \rightarrow A\alpha$.

$E \rightarrow E + T \mid T$
$T \rightarrow T * E \mid F$ (8)
$F \rightarrow (E) \mid id$

$E \rightarrow TE'$
$E' \rightarrow + TE' \mid \varepsilon$ (9)
$T \rightarrow FT'$
$T' \rightarrow * FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

Γενικότερα για να απαλειφθεί η άμεση αριστερή αναδρομή σε όλους τους Α-κανόνες, τους ομαδοποιούμε ως εξής:

$$A \rightarrow A\,\alpha_1 \mid A\,\alpha_2 \ldots \mid A\,\alpha_k \mid \beta_1 \mid \beta_2 \ldots \mid \beta_n$$

όπου κανένα β_i δεν αρχίζει με Α, και αντικαθιστούμε τα Α με τα:

$$A \rightarrow \beta_1\,A' \mid \beta_2\,A' \mid \ldots \mid \beta_n\,A'$$

$$A' \rightarrow \alpha_1\,A' \mid \alpha_2\,A' \mid \ldots \mid \alpha_k\,A' \mid \varepsilon$$

Η διαδικασία αυτή απαλείφει όλες τις άμεσες αριστερές αναδρομές (αρκεί να μην υπάρχει α_i που είναι ε), αλλά όχι και στις αριστερές αναδρομές που περιλαμβάνουν παραγωγές δυο ή περισσοτέρων βημάτων (αυτό μπορείτε να το δείτε στα βιβλία [2] και [3]) όπως συμβαίνει στην παρακάτω γραμματική.

Π.χ. $S \rightarrow A\alpha \mid b$

$A \rightarrow Ac \mid Sd \mid \varepsilon$

ΕΝΟΤΗΤΑ 4.5 ΑΝΑΛΥΤΕΣ ΑΝΑΔΡΟΜΙΚΗΣ ΚΑΤΑΒΑΣΗΣ

Για να μη χρειάζεται backtracking, πρέπει να γνωρίζουμε, δοθέντος του τρέχοντος συμβόλου εισόδου a και του μη τερματικού Α (υπό ανάπτυξη), ποιος από τους εναλλακτικούς κανόνες $A \rightarrow \alpha_1 \mid \alpha_2 \mid .. \mid \alpha_n$ είναι ο μοναδικός ο οποίος παράγει συμβολοσειρές που αρχίζουν με a. Π.χ. στους κανόνες

S → **if** C **then** S **else** S

|**while** C **do** S

|**begin** S-list **end**

Οι αποκλειστικές λέξεις **if**, **while** και **begin** μας πληροφορούν για το ποιον εναλλακτικό κανόνα πρέπει να επιλέξουμε για να αναγνωρίσουμε ένα S. Ένας Συντακτικός Αναλυτής που χρησιμοποιεί αναδρομικές ρουτίνες για να αναγνωρίσει μια συμβολοσειρά χωρίς backtracking καλείται Αναλυτής Αναδρομικής Κατάβασης. Οι αναδρομικές ρουτίνες υλοποιούνται σε γλώσσα που επιτρέπει αναδρομή.

Παράδειγμα 6 / Κεφ 4

Παρακάτω δίνονται μερικές από τις αναδρομικές ρουτίνες της γραμματικής (9) οι οποίες αποτελούν μέρος ενός Αναλυτή Αναδρομικής Κατάβασης. Οι ρουτίνες αυτές δεν χρειάζεται να αναφέρουν επιτυχία ή αποτυχία (**true**, **false**) γιατί η ρουτίνα που τις καλεί δεν πρόκειται να δοκιμάσει άλλο εναλλακτικό κανόνα. Σε περίπτωση αποτυχίας-μη αναγνώρισης-καλείται μια ρουτίνα σφάλματος-διόρθωσης που ονομάζεται error.

```
procedure E( );                              /* E → TE'  */
begin
  T( ) ;
  ETONOS( )
end
```

```
procedure ETONOS( ) ;                        /* E' → +TE' | ε */
  if input-symbol = '+'
                then
   begin advance( ) ;
         T( ) ;
         ETONOS( ) end;
             else /* ε */;
```

```
procedure F( );                              /* F → (E) | id  */
if input-symbol = 'id'
  then advance( )
  else if input-symbol = '('
        then
          begin
             advance( );
             E( );
             if input-symbol = ')'
               then advance( ) else error( )
           end
        else error( );
```

Άσκηση αυτοαξιολόγησης 3 / Κεφ. 4

Προσπαθήστε να γράψετε τις αντίστοιχες ρουτίνες για τους κανόνες

$T \rightarrow FT'$ και
$T' \rightarrow * FT' \mid \varepsilon$
της γραμματικής (9)

Συχνά μια γραμματική που γράφει κάποιος είναι ακατάλληλη για Αναδρομική Κατάβαση έστω και αν δεν περιέχει αριστερές αναδρομές. Π.χ. στους κανόνες

$S \rightarrow$ **if** C **then** S **else** S
| **if** C **then** S

βλέποντας το σύμβολο **if** δεν μπορούμε να διαλέξουμε ένα μοναδικό κανόνα για να αναπτύξουμε σε S.

Γενικότερα, αν $A \rightarrow \alpha\beta|\alpha\gamma$ είναι δύο A-κανόνες και η συμβολοσειρά εισόδου αρχίζει με κάποια μη κενή συμβολοσειρά που παράγεται από το α, δεν ξέρουμε αν πρέπει να αναπτύξουμε το A σε αβ ή σε αγ. Η λύση είναι ο μετασχηματισμός (αριστερή παραγοντοποίηση) των κανόνων $A \rightarrow \alpha\beta|\alpha\gamma$ στους ισοδύναμους:

$A \rightarrow \alpha A'$ και
$A' \rightarrow \beta \mid \gamma$

Παράδειγμα 7 / Κεφ. 4 Έστω η γραμματική:

$S \rightarrow iCtS \mid iCtSeS \mid a$
$C \rightarrow b$

μετά την αριστερά-παραγοντοποίηση γίνεται:

$S \rightarrow iCtSS' \mid a$
$S' \rightarrow eS \mid \varepsilon$
$C \rightarrow b$

ΕΝΟΤΗΤΑ 4.6 ΑΝΑΛΥΤΕΣ LL (PREDICTIVE PARSERS)

Ένας Predictive (PP) αναλυτής είναι ένας διαφορετικός τρόπος υλοποίησης της μεθόδου Αναδρομικής Κατάβασης ο οποίος δεν εμπλέκει κλήσεις αναδρομικών ρουτινών. Οι γραμματικές που ικανοποιούν τις απαιτήσεις ώστε να είναι κατάλληλες για την υλοποίηση ενός αναλυτή Αναδρομικής Κατάβασης λέμε ότι ανήκουν στην κατηγορία LL(1) και επομένως είναι κατάλληλες και για ένα αναλυτή τύπου LL. Η εξήγηση του συμβολισμού LL(1) σημαίνει Left to right, Leftmost derivation, one look ahead symbol. Ο LL(1) ή Predictive αναλυτής (στη συνέχεια οι όροι και θα χρησιμοποιούνται σαν συνώνυμοι) υλοποιείται σαν ένα

αυτόματο στοίβας ειδικής μορφής. Ένας LL(1) αναλυτής χρησιμοποιεί ένα πίνακα ανίχνευσης (Parsing Table) και μια στοίβα. Η στοίβα περιέχει μια σειρά από σύμβολα της γραμματικής και στο κάτω μέρος του το σύμβολο $. Αρχικά η στοίβα περιέχει το $ και το αρχικό σύμβολο της γραμματικής. Ο πίνακας ανίχνευσης είναι ένας πίνακας M[A,α], όπου A = μη τερματικό και α = τερματικό ή $.

Ο Συντακτικός Αναλυτής ελέγχεται από κάποιο πρόγραμμα οδηγό που λειτουργεί ως εξής. Το πρόγραμμα προσδιορίζει το σύμβολο X στην κορυφή της στοίβας, και το τρέχον σύμβολο εισόδου a. Τα δυο αυτά σύμβολα καθορίζουν τον τρόπο λειτουργίας του αναλυτή:

1. Αν X = a = $, ο αναλυτής σταματά και ανακοινώνει επιτυχή αναγνώριση.
2. Αν X = a ≠ $, ο αναλυτής απορρίπτει το X από τη στοίβα και προχωράει τον δείκτη εισόδου στο επόμενο σύμβολο εισόδου.
3. Αν X ≠ a και X = τερματικό, καλείται μια ρουτίνα λάθους.
4. Αν το X είναι κάποιο μη τερματικό, το πρόγραμμα συμβουλεύεται το στοιχείο M[X,a], το οποίο μπορεί να είναι είτε κάποιος X-κανόνας της γραμματικής ή η επιγραφή μιας ρουτίνας λάθους-διόρθωσης. Αν M[X,a]={X→UVW}, το πρόγραμμα αντικαθιστά το X στην κορυφή της στοίβας με το WVU (το U στην κορυφή). Σαν έξοδος γίνεται η σημασιολογική λειτουργία που καθορίζει η γραμματική γι' αυτόν τον κανόνα. Προς το παρόν μπορούμε να υποθέσουμε ότι απλώς τυπώνεται ο κανόνας. Αν M[X,a] = error, το πρόγραμμα καλεί μια ρουτίνα λάθους.

Μια διαγραμματική απεικόνιση του τρόπου λειτουργίας ενός LL(1) συντακτικού αναλυτή δείχνεται στο σχήμα 4.6.

Δραστηριοτητα 2 / Κεφ. 4

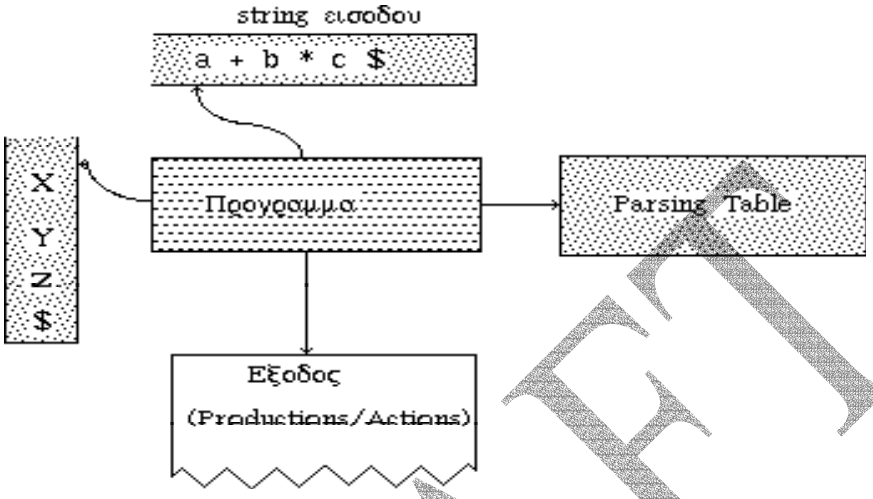
Σας δίνεται η παρακάτω γραμματική (9) :

$E \rightarrow TE', E' \rightarrow +TE'|\varepsilon, T \rightarrow FT', T' \rightarrow *FT'|\varepsilon, F \rightarrow (E)|$ **id** (9)

Ένας πίνακας συντακτικής ανάλυσης για την (9) είναι αυτός που δίνεται στο Σχήμα 4.7, όπου οι κενές θέσεις είναι θέσεις για ρουτίνες σφαλμάτων οι δε θέσεις με την ένδειξη "Synch" χρησιμοποιούνται για ανάνηψη από συντακτικά λάθη. [Δεν θα αναφερθούμε στους αλγόριθμους κατασκευής του πίνακα αυτού. Αν σας ενδιαφέρουν μπορείτε να τους βρείτε στην βιβλιογραφία. Όσον αφορά την ανάνηψη από λάθη συμβουλευθείτε το κεφάλαιο 3 της θεματικής υποενότητας 'Εργαστήρια Μεταγλωττιστών' αλλά και τη βιβλιογραφία].

Δοθέντος λοιπόν του πίνακα και του τρόπου λειτουργίας ενός LL(1) αναλυτή προσπαθήστε να καταγράψετε τα βήματα του αναλυτή όταν είσοδός του είναι η συμβολοσειρά id+id*id.

Προσπαθήστε μόνοι σας και κατόπιν συγκρίνετε την απάντησή σας με αυτή του Σχήματος 4.8.



Σχήμα 4.6 Διάγραμμα λειτουργίας ενός Predictive Αναλυτή

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

Σχήμα 4.7 Ο πίνακας συντακτικής ανάλυσης της γραμματικής (9).

Οι είσοδοι Synch αναφέρονται σε σύνολα χαρακτήρων συγχρονισμού για ανάνηψη του Αναλυτού από συντακτικά λάθη.

στοίβα	είσοδος	έξοδος
$ E	id+id*id$	
$ E' T	id+id*id$	E -> TE '
$ E ' T ' F	id+id*id$	T -> FT '
$ E' T ' id	id+id*id$	F -> id
$ E ' T '	+id*id$	
$ E '	+id*id$	T ' -> ε
$ E ' T +	+id*id$	E ' -> +TE '

$ E ' T	id*id$	
$ E ' T ' F	id*id$	T -> FT '
$ E ' T ' id	id*id$	F -> id
$ E ' T '	*id$	
$ E ' T ' F*	*id$	T ' -> *FT '
$ E ' T ' F	id$	
$ E ' T ' id	id$	F -> id
$ E ' T '	$	
$ E '	$	T ' ->ε
$	$	E ' ->ε

Σχήμα 4.8 Τα βήματα του Αναλυτή για είσοδο id+id*id

Δίνουμε τώρα στο Σχήμα 4.9, σε μορφή ψευδοκώδικα, το πρόγραμμα οδηγό που αντιστοιχεί στη λειτουργία του LL(1) αναλυτή.

```
/*έστω X το κορυφαίο στοιχείο της στοίβας και
            a  το επόμενο σύμβολο στην είσοδο */
if X=a=$
then /* να τερματίσεις. Άδεια στοίβα και άδεια είσοδος */
else
    repeat
    begin
     if X είναι τερματικό or  $
        then
          if  X = a
            then
             begin  βγάλε το X από τη στοίβα και το a  από την είσοδο end
            else error( )
        else    /* X  είναι μη τερματικό σύμβολο */
          if   M[X,a] = X -> Y1 Y2 ...Yκ
          then
            begin
             βγάλε το X από τη στοίβα  και
            βάλε στη στοίβα το  Y1 ,Y2 ,...,Yκ
            end
          else  error( )
    end
    until  X = $ */ η στοίβα είναι άδεια */.
```

Σχήμα 4.9 Πρόγραμμα οδηγός για Predictive Αναλυτή

Άσκηση αυτοαξιολόγησης 4 / Κεφ. 4

Κάθε μια από τις παρακάτω δηλώσεις μπορεί να είναι αληθής ή ψευδής. Προσπαθήστε να αναγνωρίσετε ποια είναι αληθής και ποια ψευδής και φυσικά να εξηγήσετε γιατί.

α. Οι γραμματικές που είναι κατάλληλες για αναλυτές τύπου Αναδρομικής Κατάβασης είναι επίσης κατάλληλες και για αναλυτές LL(1).

Β. Μερικές γραμματικές οι οποίες είναι κατάλληλες για συντακτικούς αναλυτές τύπου bottom up είναι επίσης κατάλληλες και για αναλυτές τύπου LL(1). Τυπικό παράδειγμα αποτελεί η γραμματική

1. $E \rightarrow E + T \mid T$
2. $T \rightarrow T * F \mid F$ (6)
3. $F \rightarrow (\ E\ ) \mid \mathbf{id}$

για την οποία κατασκευάσαμε τον πίνακα προτεραιοτήτων του σχήματος 4.3.

ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Στο κεφάλαιο αυτό μελετήσαμε τις βασικότερες μεθοδολογίες συντακτικής ανάλυσης (ανίχνευσης). Είδαμε πώς μπορούμε να κατασκευάσουμε συντακτικούς αναλυτές του τύπου Operator Precedence, Αναδρομικής Κατάβασης και Predictive– LL(1).

Ακόμη μελετήσαμε τι είδους μετασχηματισμούς πρέπει να κάνουμε σε μια γραμματική ώστε να γίνει κατάλληλη για συντακτικούς αναλυτές της κατηγορίας top down χωρίς οπισθοανίχνευση, δηλαδή για τους αναλυτές Αναδρομικής Κατάβασης και τους predictive – LL(1). Σκόπιμα δεν μελετήσαμε καθόλου τους αναλυτές τύπου LR διότι αυτοί αποτελούν ιδιαίτερο αντικείμενο μελέτης στην θεματική υποενότητα ΘΕ 9.1.

ΑΠΑΝΤΗΣΕΙΣ ΑΣΚΗΣΕΩΝ ΑΥΤΟΑΞΙΟΛΟΓΗΣΗΣ

Απάντηση άσκησης 1 / Κεφ 4

Για διευκόλυνση σας επαναλαμβάνουμε εδώ τον πίνακα προτεραιοτήτων του Σχήματος 4.3.

	+	*	(	)	id	$
+	·>	<·	<·	·>	<·	·>
*	·>	·>	<·	·>	<·	·>
(	<·	<·	<·	=	<·	e_4
)	·>	·>	e_3	·>	e_3	·>
id	·>	·>	e_3	·>	e_3	·>
$	<·	<·	<·	e_2	<·	e_1

Να θυμίσουμε ότι τα τερματικά σύμβολα της αριστερής στήλης του πίνακα αντιστοιχούν στα σύμβολα στην κορυφή της στοίβας, ενώ τα τερματικά σύμβολα

της οριζόντιας γραμμής του πίνακα αντιστοιχούν στα σύμβολα της εισόδου.

Παρατηρώντας τον πίνακα βλέπουμε ότι συντακτικό λάθος έχουμε στις εξής περιπτώσεις.

(, $ e4
), (e3
), id e3
id, (e3
id, id e3
$,) e2
$, $ e1

Μπορούμε τώρα να διακρίνουμε ότι η περίπτωση λάθους που αντιστοιχεί στην ρουτίνα e3 ουσιαστικά είναι η περίπτωση κατά την οποία λείπει ένας τελεστής ανάμεσα στα ζευγάρια ")(", ")id", "id(", και "idid". Στην περίπτωση αυτή η ρουτίνα λάθους θα πρέπει: (1) να τυπώσει ένα διαγνωστικό μήνυμα συντακτικού λάθους, (2) να επιχειρήσει να ανανήψει από το λάθος «εισάγοντας» ένα τελεστή στην στοίβα και (3) να ενημερώσει τον προγραμματιστή για την πράξη του αυτή. Έτσι ο ψευδοκώδικας για την ρουτίνα e3 θα μπορούσε να είναι όπως ο παρακάτω:

```
e3:     print "missing operator"
        print "assume "+" is the missing operator "
        push_on_stack("+")
        exit
```

Για την ρουτίνα e2 ο ψευδοκώδικας θα μπορούσε να είναι

```
e2:     print "missing ("
        print "insert "(""
        push_on_stack("(")
        exit
```

Για την ρουτίνα e4 ο ψευδοκώδικας θα μπορούσε να είναι

```
e4:     print "missing )"
        print "insert ")""
        push_on_stack(")")
        exit
```

Για την ρουτίνα e1 ο ψευδοκώδικας θα μπορούσε να είναι

```
e1:     print "missing sentence for translation"
```

print “terminate compilation”
exit

Απάντηση άσκησης 2 / Κεφ 4

Κατασκευάζουμε τα σύνολα των συμβολοσειρών που παράγονται από τον αρχικό κανόνα του Α και από τον μετασχηματισμένο Α (δηλαδή κατασκευάζουμε τις παραγωγές του Α και στις δυο περιπτώσεις).

A	A
|	|
Aα	βA'
|	|
Aα α	βαA'
|	|
βα α (α)	βααA' (β)
	|
	βααε

Βλέπουμε δηλαδή ότι και στις δύο περιπτώσεις παράγονται οι ίδιες συμβολοσειρές : βαα και βααε (=βαα).

Απάντηση άσκησης 3 / Κεφ 4

Οι ζητούμενες ρουτίνες Αναδρομικής Κατάβασης της γραμματικής (9) είναι οι παρακάτω.

```
procedure T( );                                  /* T → FT'  */
  begin F( ); TTONOS( ) end;
```

```
  procedure TTONOS( ) ;                          /* T' → *FT' | ε  */
  if input-symbol '*'
    then begin advance( );
                F( );
                TTONOS( )
         end
  else /* ε */;
```

Απάντηση άσκησης 4 / Κεφ 4

α. Αν απαντήσατε ότι η δήλωση αυτή είναι **αληθής** τότε μπράβο σας γιατί φαίνεται ότι έχετε κατανοήσει καλά την ενότητα αυτή.

Αν απαντήσατε ότι η δήλωση αυτή είναι **ψευδής** τότε δεν έχετε καταλάβει καλά ότι για τους top down αναλυτές χωρίς οπισθοανίχνευση όπως είναι οι Αναδρομικής Κατάβασης και οι LL(1) οι απαιτήσεις από τις γραμματικές είναι ίδιες. Απλά ένας LL(1) αναλυτής είναι ένας διαφορετικός τρόπος υλοποίησης της Αναδρομικής Κατάβασης.

β. Αν απαντήσατε ότι η δήλωση αυτή είναι **ψευδής** πάλι μπράβο σας και καλά θα κάνετε να συνεχίσετε με το επόμενο κεφάλαιο.

Αν απαντήσατε ότι η δήλωση αυτή είναι **αληθής** έχετε κάνει λάθος αλλά μην απογοητεύεστε γιατί ήτανε μια πολύ παραπλανητική ερώτηση. Αυτό διότι η δήλωση: "Μερικές γραμματικές οι οποίες είναι κατάλληλες για συντακτικούς αναλυτές τύπου bottom up είναι επίσης κατάλληλες και για αναλυτές τύπου LL(1)" θα μπορούσε να ήταν και αληθής σε κάποιες περιπτώσεις. Όμως το παράδειγμα που σας δίνεται, δηλαδή η γραμματική (6)

1. $E \rightarrow E + T \mid T$
2. $T \rightarrow T * F \mid F$ (6)
4. $F \rightarrow (E) \mid id$

Έχει μεν χρησιμοποιηθεί σε Operator Precedence αναλυτή ο οποίος είναι πράγματι bottom up αναλυτής, αλλά η (6) έχει τους κανόνες 1 και 2 οι οποίοι είναι αριστερά αναδρομικοί κανόνες πράγμα που σε ένα Operator Precedence αναλυτή δεν δημιουργεί κανένα πρόβλημα, αλλά δεν επιτρέπεται σε αναλυτές τύπου LL(1). Έτσι η συνολική δήλωση είναι ψευδής.