ΚΕΦΑΛΑΙΟ 2 ΛΕΚΤΙΚΗ ΑΝΑΛΥΣΗ

Στόχος

Στόχος του κεφαλαίου αυτού είναι να δώσει μια σύντομη αλλά πλήρη παρουσίαση του πώς ακριβώς λειτουργεί ένας Λεκτικός Αναλυτής, πώς μπορείτε να κατασκευάσετε ένα Λεκτικό Αναλυτή με το χέρι και πώς να χρησιμοποιήσετε Κανονικές Εκφράσεις για να κατασκευάσετε αυτόματα έναν Λεκτικό Αναλυτή με τη βοήθεια κάποιου εργαλείου (LEX). Επίσης πώς λειτουργεί ένας τέτοιος γεννήτορας Λεκτικών Αναλυτών.

Προσδοκώμενα Αποτελέσματα

'Οταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να:

- εξηγήσετε τι δουλειά κάνει ένας Λεκτικός Αναλυτής,
- περιγράψετε πώς ακριβώς λειτουργεί ένας Λεκτικός Αναλυτής για να επιτελέσει την εργασία του,
- εξηγήσετε τι είναι οι Κανονικές Εκφράσεις και πώς περιγράφουν τα λεκτικά στοιχεία μιας γλώσσας προγραμματισμού,
- γράψετε Κανονικές Εκφράσεις που περιγράφουν τα tokens μιας γλώσσας,
- γράψετε Διαγράμματα Μετάβασης Καταστάσεων και Πίνακες Μετάβασης Καταστάσεων για ένα σύνολο Κανονικών Εκφράσεων,
- γράψετε τις ρουτίνες ενός απλού Λεκτικού Αναλυτή,
- εξηγήσετε τι είναι το εργαλείο LEX,
- περιγράψετε πώς γίνεται αυτόματα (από ένα εργαλείο όπως το LEX) η κατασκευή ενός Λεκτικού Αναλυτή,
- γράψετε ένα "πρόγραμμα" για το LEX.

ΕΝΝΟΙΕΣ-ΚΛΕΙΔΙΑ

Λεκτικός Αναλυτής (ΛΑ), Token, Κανονική 'Εκφραση (ΚΕ), Διάγραμμα Μετάβασης Καταστάσεων (ΔΜΚ), Πεπερασμένα Αυτόματα (ΠΑ), Συμβολοσειρά, Γλώσσα, Προσδιοριστικό Πεπερασμένο Αυτόματο (ΠΠΑ), Μη Προσδιοριστικό Πεπερασμένο Αυτόματο (ΜΠΠΑ), Πίνακας Μεταβάσεων (ΠΜ), LEX.

'Εχετε ήδη μελετήσει το Κεφάλαιο 1 και γνωρίζετε πλέον την βασική οργάνωση και τον τρόπο λειτουργίας ενός Μεταγλωττιστή, ενός Διερμηνευτή και

γενικότερα ενός Μεταφραστού. Στη συνέχεια θα εξετάσουμε τον τρόπο λειτουργίας και τρόπους κατασκευής ενός Λεκτικού Αναλυτή.

Για να κάνετε κτήμα σας το υλικό του κεφαλαίου αυτού δεν χρειάζεστε κάποιο επιπλέον υλικό. 'Ομως, αν θέλετε να αποκτήσετε πιο ολοκληρωμένη γνώση ή συμπληρωματικές πληροφορίες γύρω από την θεωρία που αποτελεί το υπόβαθρο για την κατασκευή Λεκτικών Αναλυτών σας συμβουλεύω να δείτε το υλικό της Θεματικής Υποενότητας Αυτόματα και Τυπικές Γλώσσες του προγράμματος Πληροφορικής.

ΕΝΟΤΗΤΑ 2.1 ΕΙΣΑΓΩΓΗ

'Εχει αναφερθεί ότι η λειτουργία του Λεκτικού Αναλυτή (ΛΑ) είναι να διαβάζει ένα-ένα τους χαρακτήρες του πηγαίου (Source) προγράμματος και να τους μεταφράζει σε tokens (π.χ. keywords, identifiers, constants). Το κεφάλαιο αυτό πραγματεύεται τον σχεδιασμό και την υλοποίηση των ΛΑ.

Επειδή χρειάζεται κάποιος τρόπος περιγραφής των διαφόρων tokens που εμφανίζονται σε κάποια γλώσσα, εισάγονται οι Κανονικές Εκφράσεις (Regular Expressions). Ακόμη χρειάζεται κάποιος μηχανισμός που να αναγνωρίζει τα tokens της γλώσσας. Τέτοιοι μηχανισμοί (token Recognizers) είναι τα Διαγράμματα Μετάβασης Καταστάσεων (Transition Diagrams) και Πεπερασμένα Αυτόματα (Finite Automata).

'Ενα πλεονέκτημα της χρήσης των Κανονικών Εκφράσεων για τον καθορισμό των tokens είναι το γεγονός ότι από μια Κανονική 'Εκφραση μπορούμε να κατασκευάσουμε αυτόματα ένα αναγνωριστή για τα tokens (token Recognizer) που παριστάνονται με Κανονικές Εκφράσεις.

Ο Λεκτικός Αναλυτής μπορεί να σχεδιαστεί σε χωριστό πέρασμα του Μεταγλωττιστή ή να συνεργάζεται με τον Συντακτικό Αναλυτή στο ίδιο πέρασμα (ο Λεκτικός Αναλυτής λειτουργεί σαν υπορουτίνα ή συνρουτίνα και καλείται από τον Συντακτικό Αναλυτή όταν αυτός χρειάζεται κάποιο token).

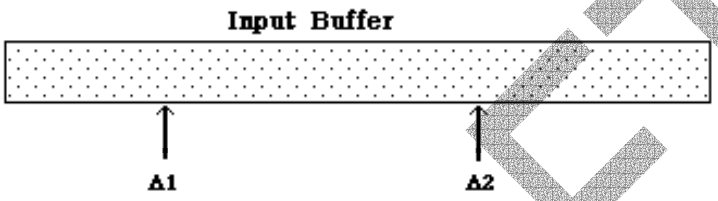
Ο σκοπός του διαχωρισμού της Λεκτικής και Συντακτικής Ανάλυσης είναι η απλοποίηση του όλου σχεδιασμού του Μεταγλωττιστή.

ΕΝΟΤΗΤΑ 2.2 ΣΧΕΔΙΑΣΜΟΣ ΛΕΚΤΙΚΟΥ ΑΝΑΛΥΤΗ

Επειδή ο Λεκτικός Αναλυτής για να αναγνωρίσει κάποιο token χρειάζεται πολλές φορές να διαβάσει αρκετούς χαρακτήρες μετά το τέλος του token, χρησιμοποιείται ένας βοηθητικός χώρος (buffer) από τον οποίο ο Λεκτικός Αναλυτής διαβάζει τους χαρακτήρες του. Χρησιμοποιούνται δύο δείκτες, Δ1 που δείχνει την αρχή του token και Δ2 (Lookahead δείκτης) που κινείται πέρα από τον Δ1 όσο χρειάζεται για να αναγνωριστεί το token.

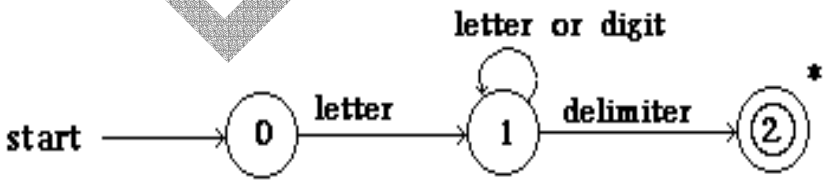
Η απόσταση που πρέπει να ταξιδέψει ο Δ2 πέρα από τον Δ1 για να αναγνωριστεί το token καλείται απόσταση Lookahead.

Μερικές φορές γίνεται μια προεπεξεργασία του Πηγαίου Προγράμματος για να απαλλαγεί από σχόλια ή κενά, ώστε να μειωθεί και η απόσταση Lookahead.



Σχήμα 2.1 Ο βοηθητικός χώρος και οι δείκτες Δ1 και Δ2 του Λεκτικού Αναλυτή

Το Διάγραμμα Μετάβασης Καταστάσεων (Transition Diagram) είναι ένα χρήσιμο εργαλείο που χρησιμοποιείται στους Λεκτικούς Αναλυτές. Στο Σχήμα 2.2 απεικονίζεται το Διάγραμμα Μετάβασης Καταστάσεων για ένα "identifier". Το σχήμα αυτό στην πραγματικότητα περιγράφει ένα αναγνωριστή tokens, δηλαδή ένα πρόγραμμα (ρουτίνα) το οποίο αναγνωρίζει tokens του τύπου "identifier".



Σχήμα 2.2 Διάγραμμα Μετάβασης Καταστάσεων για ένα "identifier".

Οι κύκλοι καλούνται καταστάσεις (states) και συνδέονται με βέλη που καλούνται πλευρές (edges). Οι επιγραφές πάνω στις πλευρές σημειώνουν τους χαρακτήρες εισόδου που μπορούν να εμφανιστούν μετά από κάθε κατάσταση.

Για να μετατρέψουμε μια συλλογή από διαγράμματα μετάβασης καταστάσεων σε ένα πρόγραμμα, κατασκευάζουμε μια ρουτίνα για κάθε κατάσταση. Το πρώτο βήμα που γίνεται στη ρουτίνα για κάθε κατάσταση είναι η κλήση κάποιας function getchar, που επιστρέφει τον επόμενο χαρακτήρα από τον buffer και προχωράει τον δείκτη Δ2 (lookahead pointer) ώστε να δείχνει στον αμέσως επόμενο χαρακτήρα.

Το επόμενο βήμα είναι να προσδιοριστεί ποιά πλευρά της κατάστασης (που επιγράφεται από τον χαρακτήρα που επέστρεψε η getchar) πρέπει να ακολουθηθεί και να μεταφερθεί ο έλεγχος του προγράμματος στην κατάσταση που δείχνει η πλευρά. Αν δεν υπάρχει τέτοια πλευρά, ο δείκτης Δ2 πρέπει να γυρίσει πίσω στην αρχική του θέση (Δ1) και να γίνει προσπάθεια αναγνώρισης άλλου token, χρησιμοποιώντας κάποιο άλλο διάγραμμα μετάβασης καταστάσεων.

Αν όλα τα διαγράμματα δοκιμάστηκαν χωρίς επιτυχία, πρέπει να κληθεί κάποια ρουτίνα λάθους, μιας και πρόκειται για κάποιο λεκτικό σφάλμα στο πηγαίο πρόγραμμα.

Η ρουτίνα για την κατάσταση 0 του Σχήματος 2.2 μπορεί να είναι:

```
state 0: C:=getchar( );
         if letter(C) then goto state 1
         else fail( )
```

Η letter επιστρέφει **true** εφ' όσον το C είναι γράμμα. Η fail γυρίζει πίσω στην αρχική του θέση Δ1 τον Δ2 και ξεκινά το επόμενο Διάγραμμα Μετάβασης Καταστάσεων αν υπάρχει, ή καλεί την ρουτίνα λάθους. Για την κατάσταση 1 η αντίστοιχη ρουτίνα είναι:

```
state 1: C:=getchar( );
         if letter(C) or digit(C)
         then goto state 1
         else if delimeter(C)
              then goto state 2
              else fail( )
```

όπου digit και delimeter είναι παρόμοιες με την letter.

Η κατάσταση 2 υπονοεί ότι έχει ευρεθεί ένας identifier. Επειδή ο delimiter (στην πλευρά προς την κατάσταση 2) δεν είναι μέρος του identifier, πρέπει ο Δ2 να γυρίσει μια θέση πίσω. Αυτό γίνεται με μια ρουτίνα retract και οι καταστάσεις που χρειάζεται να την καλούν σημειώνονται με ένα αστερίσκο. Ακόμη χρειάζεται μια ρουτίνα που θα εισάγει τον ευρεθέντα identifier στον πίνακα συμβόλων, αν δεν υπάρχει ήδη εκεί. 'Ετσι ο κώδικας για την κατάσταση 2 θα είναι ως εξής:

```
state 2: retract( );
```

return (**id**, install( ))

Η κατάσταση 2 επιστρέφει στον Συντακτικό Αναλυτή ένα ζευγάρι που συνίσταται από ένα ακέραιο **(id)** κώδικα για identifier και ένα δείκτη στον πίνακα συμβόλων, που επιστρέφεται από την install.

Εάν τα κενά αγνοούνται στην πηγαία γλώσσα, η κατάσταση 2 θα πρέπει να περιλαμβάνει και εντολές που κινούν τον αρχικό δείκτη Δ1 στον επόμενο μη κενό χαρακτήρα του τμήματος του πηγαίου προγράμματος που βρίσκεται στον buffer εισόδου.

Καλύτερο πρόγραμμα φτιάχνεται από ένα γενικευμένο συλλογικό Διάγραμμα Μετάβασης Καταστάσεων παρά από μια συλλογή ξεχωριστών διαγραμμάτων μια και δεν χρειάζεται να γυρίσει πίσω και να ξαναψάξει για token από την αρχή χρησιμοποιώντας κάποιο άλλο διάγραμμα μετάβασης καταστάσεων.

Δραστηριότητα 1 / Κεφάλαιο 2

Πάρετε τα tokens και τις τιμές τους από τον πίνακα του Σχήματος 2.3 και κατασκευάστε γενικευμένα Διαγράμματα Μετάβασης Καταστάσεων τα οποία αναγνωρίζουν τα tokens του πίνακα.

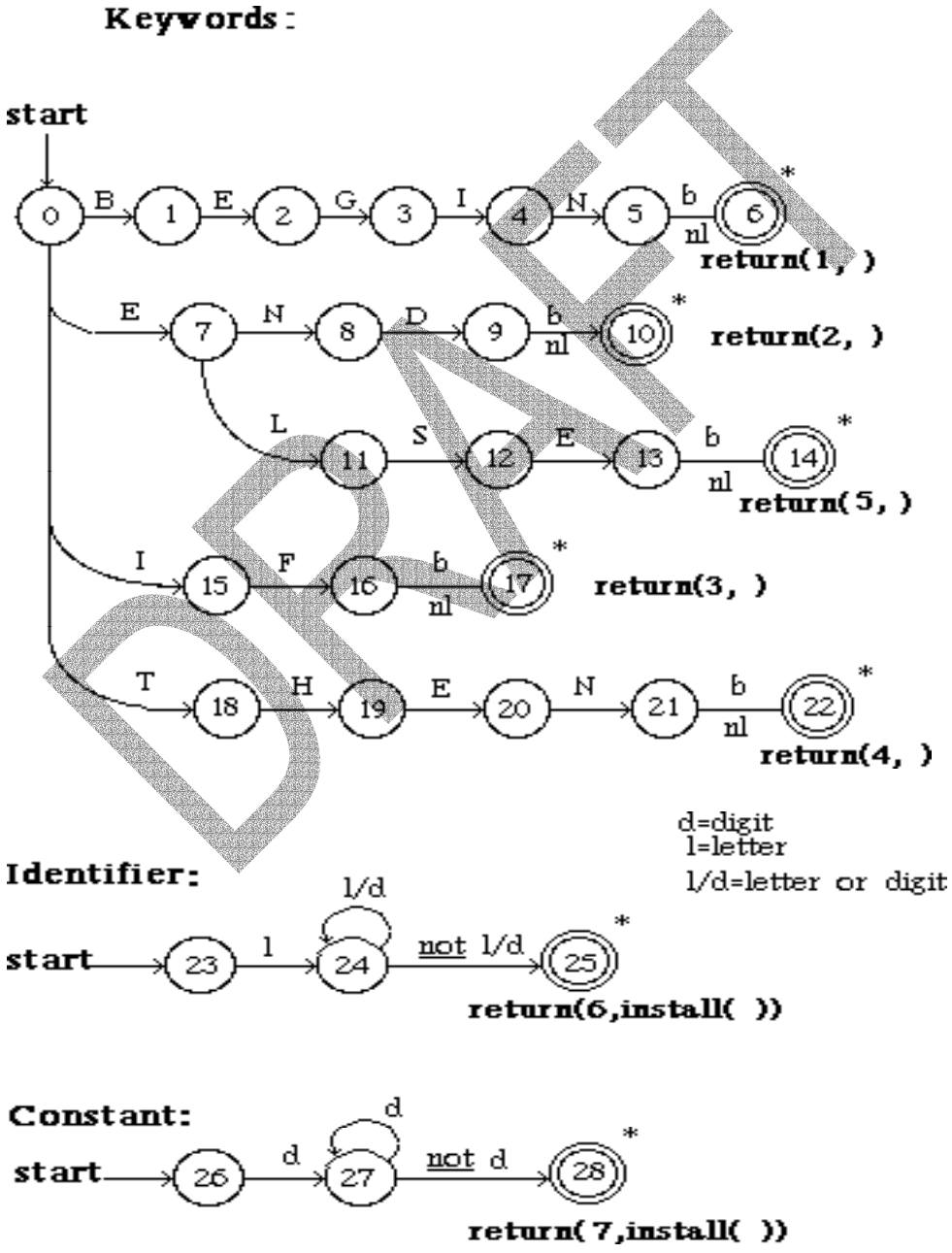
Παρατηρούμε κατ'αρχήν ότι το διάγραμμα για τα keywords δεν συνδυάζεται με εκείνο των identifiers διότι δημιουργούνται προβλήματα σύγκρουσης καταστάσεων π.χ. βλέποντας τα γράμματα "els" δεν μπορούμε να διακρίνουμε αν είμαστε στην κατάσταση 12 ή την 24.

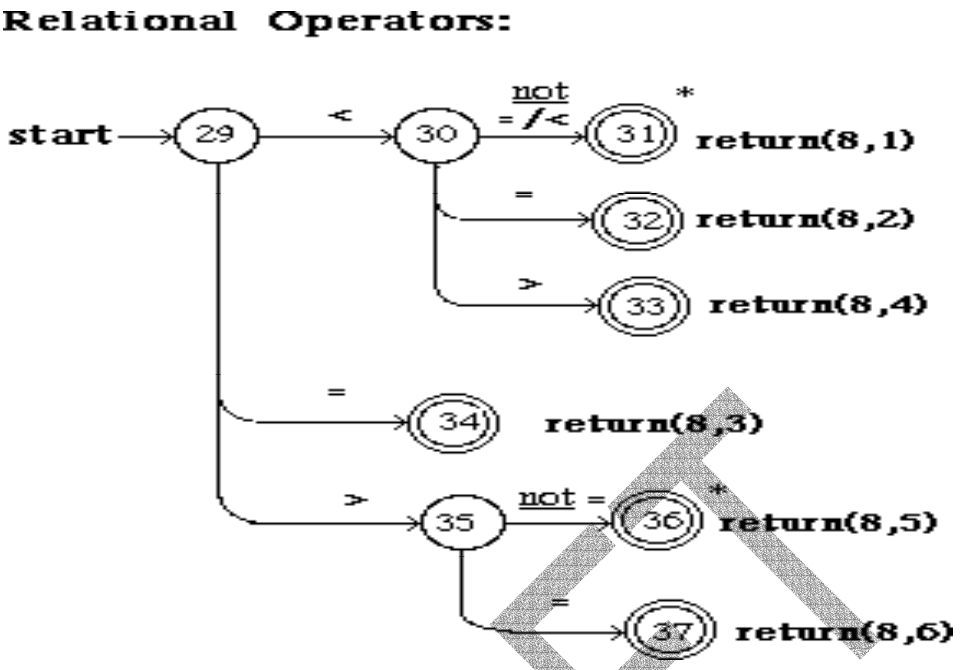
Στα διαγράμματα που δείχνονται παρακάτω δεν υπάρχουν έξοδοι λάθους. Επίσης στις καταστάσεις 5, 9, 13, 16 και 21 δεν φαίνεται ότι στην περίπτωση που ο επόμενος χαρακτήρας είναι γράμμα ή ψηφίο τότε μεταβαίνουμε στην κατάσταση 24 (identifier).

Token	Κώδικας	Τιμή
begin	1	-
end	2	-
if	3	-
then	4	-
else	5	-
indentifier	6	Δείκτης στον Πίνακα Συμβόλων
Constant	7	Δείκτης στον Πίνακα Συμβόλων

<	8	1
<=	8	2
=	8	3
<>	8	4
>	8	5
>=	8	6

Σχήμα 2.3 Πίνακας των tokens και των τιμών τους.





Σχήμα 2.4 Διαγράμματα Μετάβασης Καταστάσεων για τα tokens του Σχήμ. 2.3.

ΕΝΟΤΗΤΑ 2.3 ΚΑΝΟΝΙΚΕΣ ΕΚΦΡΑΣΕΙΣ (REGULAR EXPRESSIONS)

Σ'αυτήν την ενότητα εισάγεται ο συμβολισμός των Κανονικών Εκφράσεων που είναι κατάλληλος για να περιγράφει tokens. Κατόπιν δείχνουμε πώς οι Κανονικές Εκφράσεις μετασχηματίζονται αυτόματα σε Πεπερασμένα Αυτόματα (finite Automata), τα οποία δεν είναι παρά τυπικές προδιαγραφές (formal specifications) για τα Διαγράμματα Μετάβασης Καταστάσεων. Τέλος θα συζητηθεί η υλοποίηση των πεπερασμένων αυτόματων με προγράμματα.

2.3.1 ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΚΑΙ ΓΛΩΣΣΕΣ: ΟΡΙΣΜΟΙ

Ο όρος Αλφάβητο ή κλάση χαρακτήρων (Alphabet ή Character Class) αναφέρεται σε ένα πεπερασμένο σύνολο συμβόλων. Σύμβολο και χαρακτήρας υπονοούν το ίδιο πράγμα.

Μια συμβολοσειρά (string ή sentence ή word) είναι μια ακολουθία από πλήθος συμβόλων. Το μήκος (length) της συμβολοσειράς x συμβολίζεται με |x| και είναι το πλήθος των συμβόλων του x. (π.χ. x=01101 => |x| = 5).

Η κενή συμβολοσειρά συμβολίζεται με ε και ισχύει |ε|=0. Η σύζευξη των συμβολοσειρών x και y συμβολίζεται με x.y ή και xy.

π.χ. x = abc, y =de => xy =abcde

Ακόμη μπορούμε να γράψουμε $x^1 = x$, $x^2 = x\,x$, $x^3 = x\,x\,x$ κ.λ.π.

Prefix της συμβολοσειράς x είναι μια υπο-συμβολοσειρά (substring) του x από την οποία έχουν αφαιρεθεί οι 0 ή περισσότεροι τελευταίοι χαρακτήρες του x.

'Ενα suffix του x σχηματίζεται αφαιρώντας μηδέν ή περισσότερους από τους πρώτους χαρακτήρες του x.

Αφαιρώντας ένα prefix και ένα suffix από το χ έχουμε ένα Substring. Τα μη κενά, Prefix, Suffix και Substring του x καλούνται proper Prefix, Suffix και Substring.

Με τον όρο Γλώσσα (Language) εννοούμε οποιοδήποτε σύνολο συμβολοσειρών που σχηματίζεται από κάποιο αλφάβητο.

Ο ορισμός αυτός της γλώσσας είναι φοβερά ευρύς. Σύνολα όπως το ∅ (κενό σύνολο) ή το {ε} (το σύνολο που αποτελείται από τη κενή συμβολοσειρά) είναι γλώσσες κάτω από αυτόν τον ορισμό. Γλώσσα είναι επίσης το σύνολο των σωστών προγραμμάτων FORTRAN και το σύνολο όλων των Ελληνικών προτάσεων.

Σημειώνουμε ότι ο ορισμός της γλώσσας που δώσαμε δεν επισυνάπτει οποιοδήποτε νόημα (ή έννοια) στις συμβολοσειρές της γλώσσας. Αργότερα θα μιλήσουμε για συντακτικά - κατευθυνόμενη - μετάφραση σαν μια μέθοδο που επισυνάπτει νόημα στη γλώσσα.

Ο συμβολισμός σύζευξης ισχύει και στις γλώσσες. Αν L και M είναι γλώσσες τότε ορίζεται και η γλώσσα $LM = \{xy | x \in L \text{ και } y \in M\}$

π.χ. αν $L = \{0,01,110\}$ και $M = \{10,110\}$

τότε $LM = \{010,0110,0110,01110,11010,110110\}$

Σε αναλογία με τις συμβολοσειρές ορίζουμε:

$L^i = LLL$ (i-φορές), $L^0 = \{\varepsilon\}$, $\{\varepsilon\}L = L\{\varepsilon\} = L$

Η ένωση των γλωσσών L και M:

$LUM = \{x | x \in L \text{ ή } x \in M\}$

και $\varnothing UL = LU\varnothing = L$, $\varnothing L = L\varnothing = \varnothing$.

Ο τελεστής closure (ή "any number of") L* δηλώνει την σύζευξη της γλώσσας L με τον εαυτό της οσεσδήποτε φορές.

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Ο μοναδιαίος τελεστής + (Unary Postfix operator) χρησιμοποιείται να δηλώσει το Positive closure

$L^+ = L(L^*)$ που αποκλείει το $L = \{\varnothing\}$

(μια ή περισσότερες παρουσίες της L).

Για παράδειγμα, αν L = {αα} τότε L* είναι όλες οι συμβολοσειρές με άρτιο πλήθος από α, μια και $L^0 = \{\varepsilon\}$, $L^1 = \{\alpha\alpha\}$, $L^2 = \{\alpha\alpha\alpha\alpha\}$ κ.λ.π. Επίσης αν D είναι μία γλώσσα που αποτελείται από τις συμβολοσειρές 0,1,...9, ήτοι, κάθε συμβολοσειρά είναι ένα απλό δεκαδικό ψηφίο, τότε το D* είναι όλες οι συμβολοσειρές των ψηφίων συμπεριλαμβανομένης και της κενής συμβολοσειράς.

2.3.2 ΟΡΙΣΜΟΣ ΚΑΝΟΝΙΚΩΝ ΕΚΦΡΑΣΕΩΝ (ΚΕ)

Κανονικές Εκφράσεις είναι ένας συμβολισμός που χρησιμοποιούμε για να καθορίσουμε μία κλάση γλωσσών που είναι γνωστές σαν Κανονικά Σύνολα (Regular sets), ή Κανονικές Γραμματικές (Regular grammars). 'Ενα token είναι είτε μια μοναδική συμβολοσειρά (π.χ. σημείο στίξης) ή μία συλλογή από συμβολοσειρές ορισμένου τύπου (π.χ. identifier).

Αν θεωρήσουμε το σύνολο των συμβολοσειρών σε κάθε κλάση από tokens σαν μια γλώσσα, μπορούμε να χρησιμοποιήσουμε τον συμβολισμό των Κανονικών Εκφράσεων για να περιγράψουμε τα tokens.

Παραδείγματος χάρη, ένας identifier σε κάποια γλώσσα προγραμματισμού ορίζεται σαν ένα γράμμα ακολουθούμενο από μηδέν έως επτά γράμματα ή ψηφία. Σε (γενικευμένο) συμβολισμό Κανονικών Εκφράσεων θα μπορούσαμε να γράψουμε: identifier = letter $(\text{letter}|\text{digit})^7$

Στις ενότητες 2.5 και 2.6 θα δούμε ότι από τέτοιες περιγραφές μπορούμε αυτόματα να κατασκευάσουμε πρόγραμμα που αναγνωρίζει identifiers.

Αυτά που ονομάζουμε **Κανονικές Εκφράσεις στο Αλφάβητο Σ** είναι εκείνες ακριβώς οι εκφράσεις που κατασκευάζονται από τους παρακάτω κανόνες. Κάθε Κανονική 'Εκφραση υποδηλώνει μία Γλώσσα και δίνουμε τους κανόνες για την κατασκευή αυτών των Γλωσσών μαζί με τους κανόνες κατασκευής των Κανονικών Εκφράσεων:

1. ε είναι μία Κανονική 'Εκφραση που υποδηλώνει το {ε} δηλαδή την γλώσσα που περιέχει μόνο την κενή συμβολοσειρά.
2. για κάθε α στο Σ, α είναι μία Κανονική 'Εκφραση που υποδηλώνει το {α},

ήτοι την γλώσσα με μια μόνο συμβολοσειρά η οποία αποτελείται από το γράμμα α.

3. Αν R και S είναι Κανονικές Εκφράσεις που υποδηλώνουν τις γλώσσες L_R και L_S τότε:

α) (R)|(S) είναι μία Κανονική 'Εκφραση που υποδηλώνει την γλώσσα L_R UL_S

β) (R).(S)[1] είναι μία Κανονική 'Εκφραση που υποδηλώνει την γλώσσα L_R . L_S

γ) (R)* είναι μία Κανονική 'Εκφραση που υποδηλώνει την γλώσσα L_R*

Υπάρχει και η δυνατότητα να απαλείψουμε τις παρενθέσεις στις Κανονικές Εκφράσεις, όπου χρειάζεται, κάνοντας χρήση των κανόνων προτεραιότητας που λένε ότι το * έχει την υψηλότερη προτεραιότητα, κατόπιν το **.** και τελευταίο το |.

Στα παρακάτω παραδείγματα υποθέτουμε το Αλφάβητο Σ={α,b}. Η Κανονική 'Εκφραση, α υποδηλώνει την γλώσσα {α}, που είναι διαφορετικό πράγμα από τη συμβολοσειρά α.

Παράδειγμα 1 / Κεφ.2 Η Κανονική 'Εκφραση α* δηλώνει το closure της γλώσσας {α} ήτοι,

$$\alpha^* = \bigcup_{i=0}^{\infty} \{\alpha^i\}$$

δηλαδή το σύνολο των συμβολοσειρών με μηδέν ή περισσότερα α, ενώ η Κανονική 'Εκφραση αα*, που σύμφωνα με τους κανόνες προτεραιότητας είναι ισοδύναμη με την α(α)*, υποδηλώνει τις συμβολοσειρές που αποτελούνται από ένα ή περισσότερα α.

Παράδειγμα 2 / Κεφ. 2 Εφ' όσον η Κανονική 'Εκφραση α|b δηλώνει την γλώσσα {α,b}, η (α|b)* δηλώνει την

$$\bigcup_{i=0}^{\infty} \{a, b\}^i$$

[1] ?s?d??αμα ???feta? ?a? sa? (R)(S) και υποδηλώνει την γλώσσα $L_R L_S$

που είναι το σύνολο όλων των συμβολοσειρών που αποτελούνται από α και b συμπεριλαμβανομένης και της κενής συμβολοσειράς.

Παράδειγμα 3 / Κεφ. 2 Η Κανονική 'Εκφραση α|bα* που ομαδοποιείται και σαν α|(b(α)*) δηλώνει το σύνολο των συμβολοσειρών που συνίστανται είτε από μόνο το α είτε το b ακολουθούμενο από κανένα ή περισσότερα α.

Παράδειγμα 4 / Κεφ. 2 Η Κανονική 'Εκφραση αα|αb|ba|bb δηλώνει όλες τις συμβολοσειρές μήκους 2 με τα γράμματα α και b.

Παράδειγμα 5 / Κεφ. 2 Τα tokens στο Σχήμα 2.3 μπορούν να περιγραφούν από τις Κανονικές Εκφράσεις:

keyword = begin | end | if | then | else

identifier = letter (letter|digit)*

constant = digit$^+$

relop = <|<=|=|< >|>|>=

όπου letter σημαίνει A|B|...|Z και digit 0|1|...|9.

'Ασκηση Αυτοαξιολόγησης 1 / Κεφ. 2

Μια Κανονική 'Εκφραση σε κάποιο αλφάβητο Σ ορίζει:

(1) μια συμβολοσειρά η οποία περιγράφει μια λεκτική δομή (π.χ. μια σταθερά) σε μια γλώσσα προγραμματισμού.

(2) ένα σύνολο συμβολοσειρών το οποίο ορίζει μια "γλώσσα"

(3) ένα token σε κάποια γλώσσα προγραμματισμού.

Ποιά από τις παραπάνω τρεις πιθανές απαντήσεις είναι η σωστή;

'Ασκηση Αυτοαξιολόγησης 2 / Κεφ. 2

'Ενας συρμός τρένου αποτελείται από μια ή δύο μηχανές στην αρχή που ακολουθούνται από ένα ή περισσότερα βαγόνια που ακολουθούνται από ένα βαγόνι ελεγκτών.

Αν συμβολίσουμε την μηχανή με Μ, κάθε βαγόνι του τρένου με Β και το βαγόνι ελεγκτών με Ε, προσπάθησε να γράψεις μια Κανονική 'Εκφραση η οποία να περιγράφει διάφορα τρένα (συρμούς).

ΕΝΟΤΗΤΑ 2.4 ΠΕΠΕΡΑΣΜΕΝΑ ΑΥΤΟΜΑΤΑ (FINITE AUTOMATA)

Το μέρος του Λεκτικού Αναλυτή που αναγνωρίζει την παρουσία ενός token στη συμβολοσειρά εισόδου (πηγαίο πρόγραμμα) είναι ένας αναγνωριστής (recognizer) της γλώσσας που καθορίζει το token.

'Ενας τρόπος να μετατραπεί μια Κανονική 'Εκφραση σε ένα αναγνωριστή, είναι να δημιουργήσουμε ένα γενικευμένο Διάγραμμα Μετάβασης Καταστάσεων (ΔΜΚ) από την Κανονική 'Εκφραση.

Το διάγραμμα αυτό καλείται Μη Προσδιοριστικό Πεπερασμένο Αυτόματο, (Non Deterministic Finite Automaton) και δεν είναι εν γένει εύκολο να εξομοιωθεί από ένα απλό πρόγραμμα. Μια παραλλαγή όμως του Μη Προσδιοριστικού Πεπερασμένου Αυτόματου, το Προσδιοριστικό Πεπερασμένο Αυτόματο (Deterministic Finite Automaton) που κατασκευάζεται από το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο, μπορεί να εξομοιωθεί εύκολα από ένα πρόγραμμα.

Ορισμός

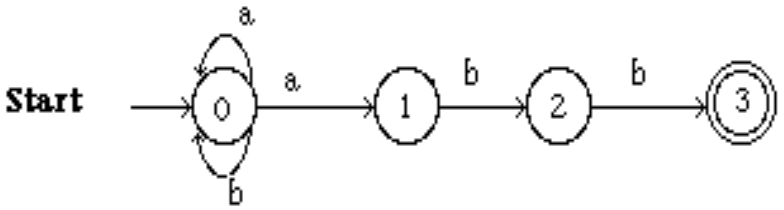
'Ενα Μη Προσδιοριστικό Πεπερασμένο Αυτόματο είναι μία πεντάδα (K, VT, M, S, Z) όπου:

1. K είναι ένα αλφάβητο στοιχείων που καλούνται καταστάσεις.
2. VT είναι ένα αλφάβητο στοιχείων που καλείται αλφάβητο εισόδου (το αλφάβητο των χαρακτήρων εισόδου),
3. M είναι μία απεικόνιση (συνάρτηση) από το KxVT μέσα στο K. (Αν M(Q,T)=R, τότε όταν το Προσδιοριστικό Πεπερασμένο Αυτόματο είναι στην κατάσταση Q με επόμενο χαρακτήρα εισόδου T, μεταβαίνει στην κατάσταση R),
4. S είναι η αρχική κατάσταση (και φυσικά $S \in K$),
5. Z είναι ένα μη κενό σύνολο τελικών καταστάσεων ($Z \subset K$).

'Ενα Μη Προσδιοριστικό Πεπερασμένο Αυτόματο που αναγνωρίζει την γλώσσα:

(a|b)*abb

είναι το παρακάτω:



Σχήμα 2.5. Το ΜΚΠΑ της γλώσσας **(a|b)*abb**

Πρόκειται δηλαδή για ένα κατευθυνόμενο γράφημα (directed graph). 'Εχει καταστάσεις (states) και πλευρές (edges) που καλούνται μεταβάσεις (transitions).

Το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο μοιάζει πολύ με ένα Διάγραμμα Μετάβασης Καταστάσεων εκτός του ότι κάποια πλευρά μπορεί να επιγράφεται με το ε και με χαρακτήρες και ότι ο ίδιος χαρακτήρας μπορεί να επιγράφει περισσότερες μεταβάσεις μιας κατάστασης.

Υπάρχει μια αρχική κατάσταση και μία ή περισσότερες τελικές καταστάσεις. Το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο μπορεί να παρασταθεί και από ένα Πίνακα Μετάβασης Καταστάσεων (Transition Table). Π.χ. Ο Πίνακας Μετάβασης Καταστάσεων (ΠΜΚ) του Σχήματος 2.5 είναι:

Κατάσταση	Σύμβολο στην Είσοδο	
	α	b
0	{0,1}	{0}
1	-	{2}
2	-	{3}

Το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο δέχεται (accepts) μια συμβολοσειρά x, εάν και μόνο εάν υπάρχει κάποιος δρόμος (path) από την αρχική κατάσταση σε μια από τις τελικές καταστάσεις. Το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο του Σχήματος 2.5 θα δεχθεί τις συμβολοσειρές εισόδου αbb, ααbb, bαbb, αααbb,... Π.χ. το ααbb γίνεται δεκτό από τον δρόμο (path) που φαίνεται παρακάτω:

Κατάσταση	Υπολειπόμενη Είσοδος
0	ααbb
0	αbb
1	bb

2	b
3	ε

Αρχικά θεωρείται ότι το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο είναι στην κατάσταση 0 διαβάζοντας το πρώτο χαρακτήρα εισόδου α.

Η Γλώσσα που ορίζεται από ένα Μη Προσδιοριστικό Πεπερασμένο Αυτόματο είναι το σύνολο των συμβολοσειρών που δέχεται (αναγνωρίζει). Π.χ. Το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο του Σχήματος 2.5 δέχεται (α|b)*αbb.

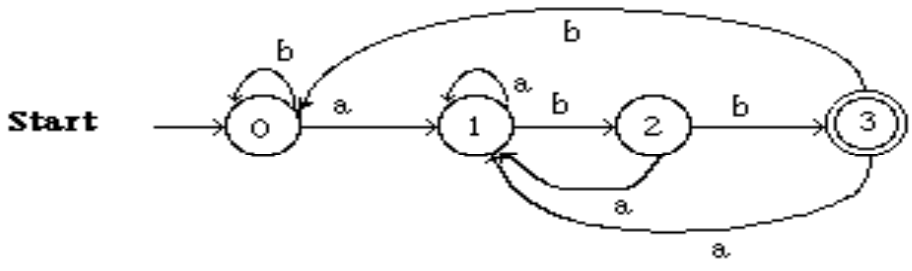
Παρατηρούμε ότι το Μη Προσδιοριστικό Πεπερασμένο Αυτόματο του Σχήματος 2.5 έχει περισσότερες από μια μεταβάσεις από την κατάσταση 0 όταν έχει είσοδο το α. Αυτός είναι ο λόγος που είναι δύσκολη η εξομοίωση του Μη Προσδιοριστικού Πεπερασμένου Αυτόματου από ένα πρόγραμμα. 'Ενα Πεπερασμένο Αυτόματο είναι Προσδιοριστικό (deterministic), αν:

1. Δεν έχει μεταβάσεις (transitions) για περίπτωση που η είσοδος είναι ε.

2. Για κάθε κατάσταση S και κάθε σύμβολο εισόδου α, υπάρχει το πολύ μια πλευρά επιγραφόμενη α, που βγαίνει από την S.

Παράδειγμα 6 / Κεφ. 2 'Ενα Προσδιοριστικό Πεπερασμένο Αυτόματο για την γλώσσα: **(a|b)*abb**
(αντίστοιχο του Μη Προσδιοριστικού Πεπερασμένου Αυτόματου του Σχήματος 2.5) είναι αυτό του Σχήματος 2.6.

Για κάθε Μη Προσδιοριστικό Πεπερασμένο Αυτόματο μπορούμε να βρούμε ένα Προσδιοριστικό Πεπερασμένο Αυτόματο που δέχεται την ίδια γλώσσα. Ο αριθμός των καταστάσεων του Προσδιοριστικού Πεπερασμένου Αυτόματου μπορεί να αυξάνει εκθετικά με τον αριθμό των καταστάσεων του Μη Προσδιοριστικού Πεπερασμένου Αυτόματου, αν και αυτό είναι σπάνιο στην πράξη.



Σχήμα 2.6 'Ενα ΠΠΑ αντίστοιχο του ΜΠΠΑ του Σχήματος 2.5

Υπάρχουν αλγόριθμοι κατασκευής ενός Προσδιοριστικού Πεπερασμένου Αυτόματου από ένα Μη Προσδιοριστικό Πεπερασμένο Αυτόματο, όπως και αλγόριθμοι κατασκευής ενός Μη Προσδιοριστικού Πεπερασμένου Αυτόματου ή και Προσδιοριστικού Πεπερασμένου Αυτόματου (πιο δύσκολοι) από μια Κανονική 'Εκφραση.

'Ασκηση Αυτοαξιολόγησης 3 / Κεφ. 2

Γράψε την Κανονική 'Εκφραση η οποία περιγράφει όλες τις συμβολοσειρές από χαρακτήρες "0" και "1", οι οποίες περιλαμβάνουν τρία "1" και απροσδιόριστο αριθμό από "0", (π.χ. 111, 0111, 010011, 10101000 κ.λ.π.).

'Ασκηση Αυτοαξιολόγησης 4 / Κεφ. 2

Δίνονται οι αποκλειστικές λέξεις:

Step, switch, string,

από τη λεκτική περιγραφή κάποιας γλώσσας προγραμματισμού. Ζητείται να κατασκευάσετε:

(1) την Κανονική 'Εκφραση που τις περιγράφει

(2) ένα Γενικευμένο Διάγραμμα Μετάβασης Καταστάσεων και

(3) τον Πίνακα Μετάβασης.

ΕΝΟΤΗΤΑ 2.5 ΑΥΤΟΜΑΤΗ ΚΑΤΑΣΚΕΥΗ ΕΝΟΣ ΛΕΚΤΙΚΟΥ ΑΝΑΛΥΤΗ

Εξετάζουμε τώρα το πώς μπορεί κάποιος να κατασκευάσει ένα Λεκτικό Αναλυτή από μια προδιαγραφή των tokens υπό μορφή μιας λίστας από Κανονικές Εκφράσεις.

Είναι αλήθεια ότι η όλη δομή των συμβόλων στις περισσότερες γλώσσες προγραμματισμού είναι αρκετά όμοια, τόσο ώστε να μπορούμε να γράψουμε ένα γενικό Λεκτικό Αναλυτή για όλες τις γλώσσες.

Ο Λεκτικός Αναλυτής αυτός θα χρησιμοποιεί διάφορους πίνακες (ή πινακοποιημένες πληροφορίες), και είναι αυτοί ακριβώς οι πίνακες οι οποίοι διαφέρουν από γλώσσα σε γλώσσα. Μπορούμε να προγραμματίσουμε ένα Κατασκευαστή (πρόγραμμα) ο οποίος δέχεται σαν είσοδο την περιγραφή των tokens (σαν Κανονική 'Εκφραση) και δημιουργεί τους απαιτούμενους πίνακες για

τη γλώσσα. Από την στιγμή που δημιουργήθηκαν από τον Κατασκευαστή οι πίνακες, ο Λεκτικός Αναλυτής είναι έτοιμος. Στο Σχήμα 2.7 φαίνεται διαγραμματικά αυτή η διαδικασία αυτόματης κατασκευής ενός Λεκτικού Αναλυτή. Διάφορα τέτοια συστήματα για την αυτόματη δημιουργία Λεκτικού Αναλυτή έχουν προταθεί και χρησιμοποιούνται (AED RWORD, LEX, FLEX κλπ.).

Το LEX [Lesk 1975] για παράδειγμα δέχεται στην είσοδο ένα σύνολο από Κανονικές Εκφράσεις μαζί με κάποια δράση (action) για κάθε Κανονική 'Εκφραση. Η δράση είναι ορισμένες εντολές οι οποίες πρέπει να εκτελούνται όταν αναγνωρίζεται ένα token (όπως π.χ. στην περίπτωση που επισυνάπτουμε σημασιολογικές λειτουργίες σε διαγράμματα μετάβασης καταστάσεων για tokens π.χ. εισαγωγή ενός identifier στο symbol table κ.λ.π.).

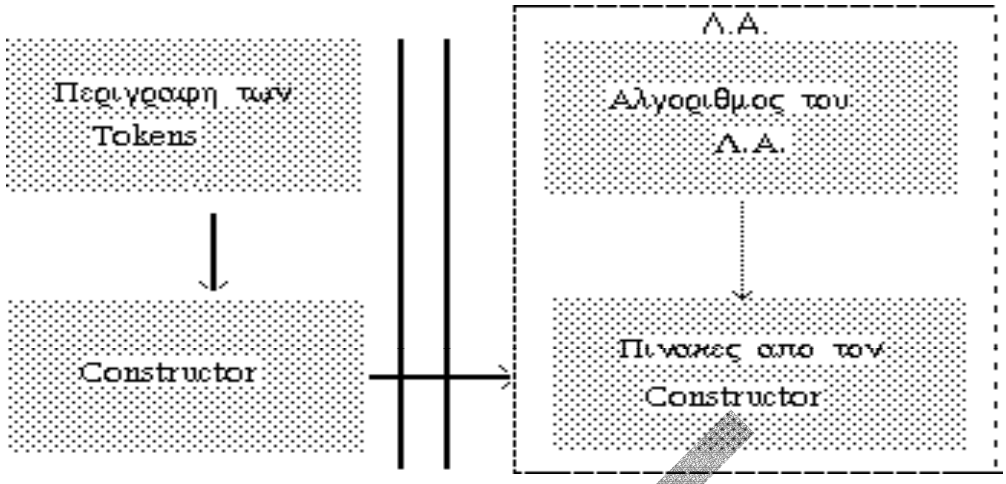
Η έξοδος του LEX είναι ένας Πίνακας Μετάβασης Καταστάσεων (ΠΜΚ) που εκφράζει ένα Προσδιοριστικό Πεπερασμένο Αυτόματο που προκύπτει από τις Κανονικές Εκφράσεις.

Ακόμη ο LEX δίνει και ένα πρόγραμμα το οποίο εξομοιώνει ένα Προσδιοριστικό Πεπερασμένο Αυτόματο το οποίο περιγράφεται από ένα Πίνακα Μετάβασης Καταστάσεων**.** Ο Λεκτικός Αναλυτής αποτελείται από πρόγραμμα εξομοίωσης και τον Πίνακα Μετάβασης Καταστάσεων.

ΕΝΟΤΗΤΑ 2.6 ΠΕΡΙΓΡΑΦΗ ΤΟΥ LEX

Ο LEX είναι ένα εργαλείο για την αυτόματη δημιουργία Λεκτικών Αναλυτών.
Το πηγαίο πρόγραμμα του LEX είναι οι προδιαγραφές ενός Λεκτικού Αναλυτή δηλαδή ένα σύνολο από Κανονικές Εκφράσεις (ΚΕ) μαζί με κάποια δράση (action) για κάθε Κανονική 'Εκφραση. Μια δράση εκτελείται όταν αναγνωρίζεται το token στου οποίου την Κανονική 'Εκφραση αντιστοιχεί η δράση.
Μια τυπική δράση, για παράδειγμα, μπορεί να επιστρέφει στον Συντακτικό Αναλυτή μια ένδειξη για το token που ευρέθηκε και να εισάγει και το token στον πίνακα συμβόλων.

Μπορούμε να θεωρήσουμε τον LEX σαν ένα Μεταγλωττιστή για μια εξειδικευμένη γλώσσα η οποία είναι κατάλληλη για να γράφουμε Λεκτικούς Αναλυτές και ορισμένα άλλα προγράμματα επεξεργασίας κειμένων.



Σχήμα 2.7 Διαγραμματική απεικόνιση της διαδικασίας αυτόματης κατασκευής Λεκτικού Αναλυτή.

Ο LEX παράγει σαν έξοδο δύο πράγματα, ένα Πίνακα Μεταβάσεων Καταστάσεων και ένα πρόγραμμα εξομοίωσης το οποίο εξομοιώνει ένα Πεπερασμένο Αυτόματο το οποίο περιγράφεται από ένα σύνολο Κανονικών Εκφράσεων.

Το πρόγραμμα εξομοίωσης του Πεπερασμένου Αυτόματου και ο πίνακας μεταβάσεων που περιγράφει το Πεπερασμένο Αυτόματο αποτελούν το Λεκτικό Αναλυτή που φτιάχνει ο LEX.

'Ενα "πηγαίο" πρόγραμμα για τον LEX αποτελείται από δύο πράγματα :

1. **βοηθητικούς ορισμούς** (auxiliary definitions) και
2. **μεταφραστικούς κανόνες** (translation Rules).

Οι βοηθητικοί ορισμοί είναι "εντολές" της μορφής:

$D_1 = R_1$

$D_2 = R_2$

...

$D_n = R_n$

που κάθε D_i είναι μοναδικό όνομα και κάθε R_i είναι μια Κανονική 'Εκφραση της οποίας τα σύμβολα διαλέγονται από τα $\Sigma U\{D_1,D_2,...,D_{i-1}\}$ δηλαδή χαρακτήρες ή προηγουμένως ορισθέντα ονόματα. Σ είναι το αλφάβητο των συμβόλων εισόδου π.χ. το ASCII ή EBCDIC σύνολο των χαρακτήρων.

Οι μεταφραστικοί κανόνες είναι "εντολές" της μορφής:

P_1 $\{A_1\}$

P_2 $\{A_2\}$

...

P_m $\{A_m\}$

'Οπου κάθε P_i είναι μια Κανονική 'Εκφραση που καλείται - πρότυπο πάνω στο αλφάβητο Σ και τα ονόματα των βοηθητικών ορισμών.

Τα πρότυπα περιγράφουν την μορφή των tokens. Κάθε A_i είναι ένα κομμάτι προγράμματος που περιγράφει τι δράση - δουλειά - πρέπει να κάνει ο Λεκτικός Αναλυτής όταν ευρεθεί το token P_i. Τα A_i's γράφονται σε κάποια από τις γλώσσες προγραμματισμού (συνήθως C).

Για να δημιουργηθεί ο Λεκτικός Αναλυτής L, κάθε A_i πρέπει να μεταφραστεί (compiled) σε κώδικα μηχανής, όπως και κάθε άλλο πρόγραμμα που είναι γραμμένο στην ίδια γλώσσα με τα A_i's.

Ο Λεκτικός Αναλυτής L που φτιάχνεται από τον LEX συμπεριφέρεται ως εξής: Ο L διαβάζει στην είσοδό του, ένα-ένα χαρακτήρα, μέχρις ότου βρεί το μεγαλύτερο prefix της εισόδου (θεωρείται σαν μια συμβολοσειρά) το οποίο ταιριάζει με μια από τις Κανονικές Εκφράσεις P_i.

Μόλις ο L βρεί αυτό το prefix, το αφαιρεί από το string εισόδου και το βάζει σε μια βοηθητική περιοχή (buffer) που καλείται TOKEN (το TOKEN μπορεί στην πράξη να είναι δύο δείκτες - αρχή και τέλος της συμβολοσειράς που ταιριάζει στην Κανονική 'Εκφραση - μέσα στον buffer εισόδου).

Κατόπιν ο L εκτελεί την δράση A_i. Μόλις τελειώσει την εκτέλεση της A_i, ο L επιστρέφει στον Συντακτικό Αναλυτή. Στην περίπτωση που κανένα από τα P_i δεν ταιριάζει με κανένα prefix της εισόδου έχει συμβεί κάποιο λάθος και πιθανότατα ο L ή παραδίδει τον έλεγχο σε κάποια ρουτίνα λάθους ή δίνει κάποια ένδειξη λάθους στον Συντακτικό Αναλυτή. Στην περίπτωση που δύο ή περισσότερα πρότυπα ταιριάζουν στο ίδιο μέγιστο prefix της εισόδου, ο L θα ευνοήσει το token που βρέθηκε πρώτο στον πίνακα των μεταφραστικών κανόνων.

Δραστηριότητα 2 / Κεφάλαιο 2

Λάβετε υπόψιν σας τα tokens που δίνονται στον πίνακα του Σχήματος 2.3 και προσπαθήστε να γράψετε τους "βοηθητικούς ορισμούς" και τους "μεταφραστικούς κανόνες", δηλαδή το "πρόγραμμα" το οποίο θα είναι είσοδος στο LEX ώστε να σας κατασκευάσει αυτόματα τον Λεκτικό Αναλυτή ο οποίος αντιστοιχεί στα tokens του Σχήματος 2.3

Ο Λεκτικός Αναλυτής (L) που φτιάχνει ο LEX επιστρέφει πάντοτε στον Συντακτικό Αναλυτή μια ποσότητα, τον τύπο του token. Για να επιστρέψει και κάποια τιμή, η τιμή αυτή τοποθετείται στην καθολική μεταβλητή LEXVAL.

Το παρακάτω "πρόγραμμα" καθορίζει τον Λεκτικό Αναλυτή που επιθυμούμε:

AUXILIARY DEFINITIONS
letter=A|B|.....|Z
digit =0|1|.....|9
TRANSLATION RULES

begin	{return 1}
end	{return 2}
if	{return 3}
then	{return 4}
else	{return 5}
letter (letter|digit)*	{LEXVAL:=INSTALL(); return 6}
digit*	{LEXVAL:=INSTALL(); return 7}
<	{LEXVAL:=1; return 8}
< =	{LEXVAL:=2; return 8}
=	{LEXVAL:=3; return 8}
<>	{LEXVAL:=4; return 8}
>	{LEXVAL:=5; return 8}
> =	{LEXVAL:=6; return 8}

Σχήμα 2.8 Το "Πρόγραμμα" του LEX για τα tokens του Σχήματος 2.3

Ας υποθέσουμε ότι ο L που θα προκύψει από τους παραπάνω κανόνες έχει είσοδο **begin** ακολουθούμενο από ένα κενό. Το πρώτο και το έκτο πρότυπο ταιριάζουν στο **begin** και κανένα πρότυπο δεν ταιριάζει σε ολόκληρη τη συμβολοσειρά εισόδου. Μια και το πρότυπο **"begin"** προηγείται στους κανόνες από εκείνο των identifiers (έκτο), ο L αναγνωρίζει το keyword **begin.** Ανάλογη κατάσταση εμφανίζεται στην αναγνώριση των < = και <.

Σύνοψη Κεφαλαίου 2

Στο κεφάλαιο αυτό μάθατε πως μπορείτε μόνοι σας (γράφοντας δηλαδή κώδικα εσείς οι ίδιοι) ή με τη βοήθεια ενός γεννήτορα λεκτικών αναλυτών όπως είναι το LEX να κατασκευάσετε ένα Λεκτικό Αναλυτή.

Μάθατε ότι για να περιγραφούν τα λεκτικά στοιχεία μιας γλώσσας προγραμματισμού, αυτά δηλαδή που αναφέρονται σαν tokens χρειαζόμαστε τον συμβολισμό των 'Κανονικών Εκφράσεων'. Όταν έχετε περιγράψει τα tokens μιας γλώσσας με τη βοήθεια των Κανονικών Εκφράσεων μπορείτε στη συνέχεια να κατασκευάσετε τα Διαγράμματα Μετάβασης Καταστάσεων τα οποία απεικονίζουν γραφικά τα tokens.

Εξηγήσαμε (στην ενότητα 2.2) πως μπορείτε να γράψετε κώδικα, δηλαδή να προγραμματίσετε ένα Λεκτικό Αναλυτή μόνοι σας γράφοντας ένα μικρό κομμάτι κώδικα για κάθε μία κατάσταση του Διαγράμματος Μετάβασης Καταστάσεων, και επίσης δείξαμε πως μπορείτε να ενοποιήσετε περισσότερα από ένα Διαγράμματα Μετάβασης Καταστάσεων και να κατασκευάσετε ένα γενικευμένο τέτοιο διάγραμμα. Θυμίζουμε εδώ ότι ένα Διάγραμμα Μετάβασης Καταστάσεων (γενικευμένο ή όχι) είναι ένα Πεπερασμένο Αυτόματο το οποίο μπορεί να είναι Προσδιοριστικό ή μη Προσδιοριστικό.

Στη συνέχεια, επειδή για τις λεκτικές δομές μιας γλώσσας προγραμματισμού (tokens) μας ενδιαφέρουν μόνο τα Προσδιοριστικά Πεπερασμένα Αυτόματα, είδατε πως αυτά αναπαρίστανται από ένα Πίνακα Μετάβασης Καταστάσεων. Έτσι λοιπόν για το σύνολο των tokens μιας γλώσσας είδατε ότι μπορείτε να κατασκευάσετε ένα γενικευμένο Πίνακα Μετάβασης Καταστάσεων ο οποίος περιλαμβάνει μια αρχική κατάσταση, ένα σύνολο τερματικών καταστάσεων (οι οποίες αντιστοιχούν στην αναγνώριση των διαφόρων τύπων tokens) και ένα αλφάβητο χαρακτήρων. Ο Πίνακας Μετάβασης Καταστάσεων είναι εύκολο να προγραμματισθεί. Μπορείτε δηλαδή να γράψετε ένα πρόγραμμα το οποίο διαβάζει ένα οποιονδήποτε Πίνακα Μετάβασης Καταστάσεων και εξομοιώνει την λειτουργία του πίνακα. Τώρα πλέον αυτό το πρόγραμμα εξομοίωσης και ο πίνακας (μαζί) αποτελούν ένα Λεκτικό Αναλυτή. Έτσι λειτουργούν στην πραγματικότητα όλοι οι γεννήτορες Λεκτικών Αναλυτών όπως το LEX το οποίο περιγράφεται συνοπτικά στην ενότητα 2.6.

Στα πλαίσια μιας ολοκληρωμένης παρουσίασης της θεωρίας πίσω από τους Λεκτικούς Αναλυτές χρειάσθηκε να αναφερθούμε, πολύ συνοπτικά, σε έννοιες

όπως η Γλώσσα, οι Κανονικές Εκφράσεις (κάθε Κανονική Έκφραση περιγράφει και μία γλώσσα), και τα Πεπερασμένα Αυτόματα (Προσδιοριστικά και μη Προσδιοριστικά). Εάν θέλετε να μελετήσετε βαθύτερα την θεωρία πίσω από αυτά τα πράγματα τότε σας συνιστώ να μελετήσετε την Θεματική Υποενότητα ‘Αυτόματα και Τυπικές Γλώσσες’ της Θεματικής Ενότητας ‘Θεμελιώσεις της Επιστήμης των Υπολογιστών’.

Απαντήσεις Ασκήσεων Αυτααξιολόγησης του Κεφαλαίου 2

Απάντηση 'Ασκησης 1

1) Η απάντηση αυτή δεν είναι σωστή, διότι μια Κανονική Έκφραση στη γενική της μορφή δεν παράγει μια μόνο συμβολοσειρά αλλά ένα σύνολο από συμβολοσειρές οι οποίες "ορίζουν" μια γλώσσα. Επομένως μια Κανονική Έκφραση δεν ορίζει "μια σταθερά" (για παράδειγμα), αλλά ένα σύνολο από σταθερές ενός συγκεκριμένου τύπου (π.χ. ακέραιες σταθερές). Επίσης δεν ορίζει ένα identifier αλλά όλους τους identifiers ενός συγκεκριμένου τύπου. Στην περίπτωση βέβαια των αποκλειστικών λέξεων όπως και των τελεστών, η Κανονική Έκφραση ορίζει μια μόνο συμβολοσειρά η οποία αντιστοιχεί στην αποκλειστική λέξη ή τον τελεστή.

(2) Θαυμάσια, αυτή είναι η σωστή απάντηση.

(3) Αυτή η απάντηση δεν είναι σωστή διότι ένα token περιγράφει μια συγκεκριμένη συμβολοσειρά ενός συγκεκριμένου τύπου. Για παράδειγμα τα tokens ALFA, BETA, και NEWLINE ορίζονται όλα από μια Κανονική Έκφραση η οποία περιγράφει τα tokens τύπου identifier.

Απάντηση 'Ασκησης 2

Η Κανονική Έκφραση η οποία περιγράφει συρμούς με την δομή που περιγράφεται στην άσκηση είναι η:

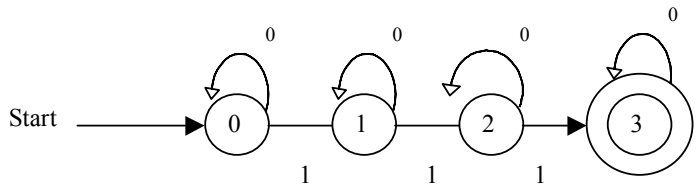
$$\mathbf{M\ (M|\varepsilon)\ B^{+}\ E}$$

Απάντηση 'Ασκησης 3

Η Κανονική Έκφραση που περιγράφει τις παραπάνω συμβολοσειρές είναι η:

0*10*10*10*

Από αυτή τη Κανονική 'Εκφραση θα μπορούσες να κατασκευάσεις και το Διάγραμμα Μετάβασης Καταστάσεων το οποίο θα είναι ως εξής



Φυσικά θα μπορούσες να κατασκευάσεις και τον Πίνακα Μεταβάσεων ο οποίος θα πρέπει να είναι ως εξής:

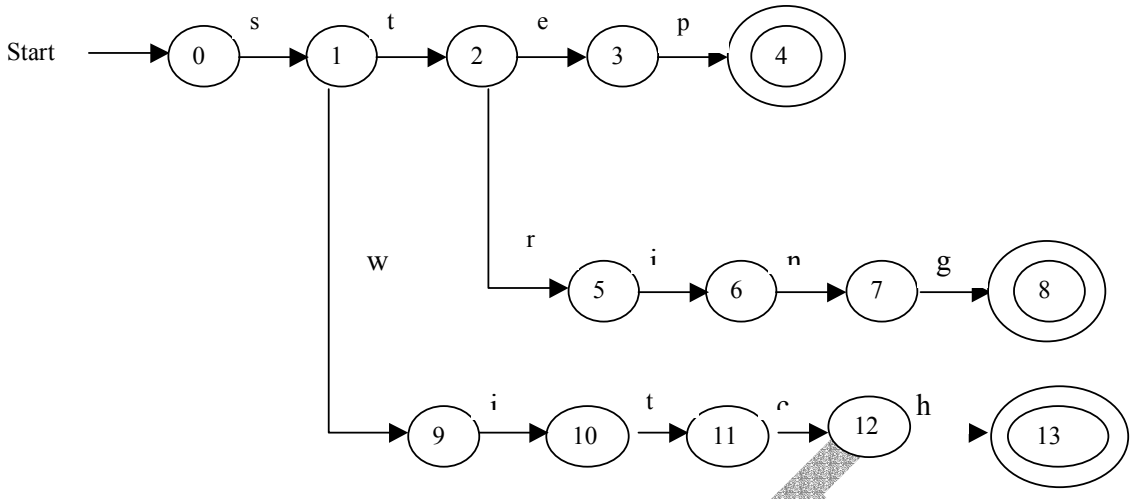
Κ/Σ	0	1	Τ
0	0	1	
1	1	2	
2	2	3	
3	3	-	Τ

Απάντηση 'Ασκησης 4

(1) Η ζητούμενη Κανονική 'Εκφραση είναι η:

s (t (ep|ring) | witch)

(2) Το δε Γενικευμένο Διάγραμμα Μετάβασης Καταστάσεων που αναγνωρίζει τις λέξεις αυτές είναι το:



(3) Τέλος ο Πίνακας Μεταβάσεων είναι ο εξής:

K/Σ	s	t	e	p	r	i	n	g	w	c	h	T
0	1											
1		2							9			
2			3		5							
3				4								
4												T
5						6						
6							7					
7								8				
8												T
9						10						
10		11										
11										12		
12											13	
13												T

Με “T” σημειώνονται οι τερματικές καταστάσεις.