ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Σχεδίαση Γλωσσών & Μεταγλωττιστές

Ενότητα 3: Λεξική Ανάλυση

Επ. Καθ. Π. Κατσαρός
Τμήμα Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

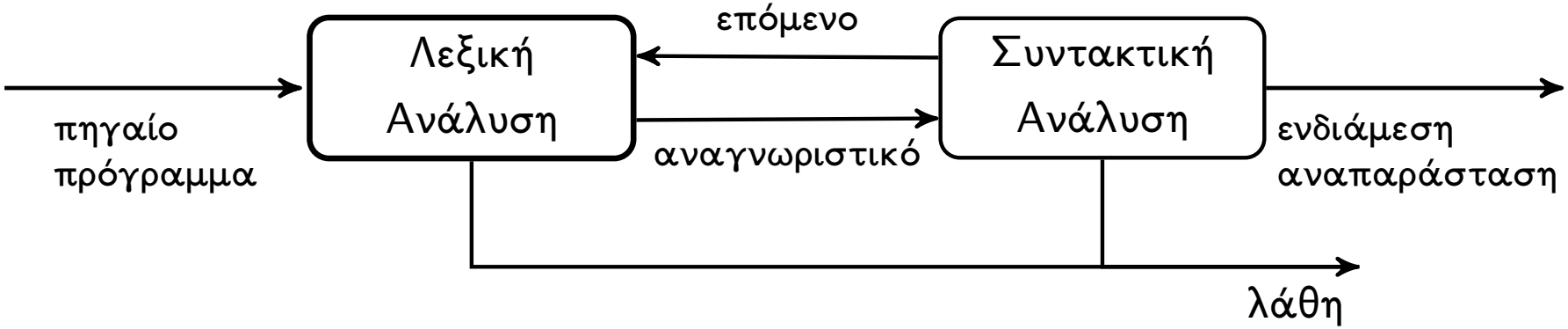
Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδεια χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο«Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Βασικές έννοιες



Η Λεξική Ανάλυση δέχεται μια συμβολοσειρά για είσοδο (πηγαίο πρόγραμμα), στην οποία καλείται να διακρίνει **λεξικές μονάδες** και να τροφοδοτήσει τη Συντακτική Ανάλυση με μία ροή **αναγνωριστικών**, ένα για την κάθε μονάδα:

- Λεξική μονάδα (lexeme): συμβολοσειρά που συμμορφώνεται στο πρότυπο κάποιου συντακτικού στοιχείου της ΓΠ.
- Αναγνωριστικό (token): συμβολική τιμή που αντιστοιχεί σε ένα από τα συντακτικά στοιχεία της ΓΠ, π.χ. όνομα (IDentifier), αριθμός (Number) κ.α.

Ο ρόλος της λεξικής ανάλυσης

- Η επεξεργασία της Λ.Α. είναι μια διαδικασία αναγνώρισης συμβολοσειρών με βάση προκαθορισμένα πρότυπα.
- Η λειτουργία της υλοποιείται από μία συνάρτηση που καλείται από το Συντακτικό Αναλυτή.
- Τα αναγνωριστικά μιας ΓΠ κατατάσσονται σε κατηγορίες όπως οι **δεσμευμένες λέξεις**, τα **ειδικά σύμβολα** (τελεστές, **διαχωριστές**), τα ονόματα, οι αριθμοί κ.α..
- Παραδείγματα λαθών κατά τη Λ.Α.:
  1. Όνομα που παραβιάζει περιορισμό μήκους της συμβολοσειράς.
  2. Απουσία χαρακτήρα που κλείνει ένα σχόλιο.
  3. Ορισμός συμβολοσειράς χωρίς τα εισαγωγικά που δηλώνουν το τέλος της.
  4. Μη έγκυροι χαρακτήρες στο κείμενο του προγράμματος.

Ο ρόλος της λεξικής ανάλυσης

Για ένα αναγωριστικό μπορεί να υπάρχουν περισσότερες από μία λεξικές μονάδες, που μπορεί να ανιχνεύσει η Λ.Α..

Λεξ. Μονάδα	**Πρότυπο συμβολοσειράς**	**Αναγνωριστικό**
+	χαρακτήρας ′+′	PLUS_OP
a1	αρχίζει από χαρακτήρα	ID
IF	οι πεζοί ή κεφαλαίοι χαρακτήρες ′I′ και ′F′	IF
1729	αποτελείται από ψηφία και δεν αρχίζει με 0	NUM

Ο ρόλος της λεξικής ανάλυσης

Για το πρόγραμμα:

```
if (i == j)
        z = 0;
else
        z = 1;
```

η Λ.Α. θα έπρεπε να δώσει τη ροή αναγνωριστικών
`IF, LPAREN, ID, EQEQ_OP, ID, RPAREN, ID, EQ_OP, NUM, SEMICOLON, ELSE, ID, EQ_OP, NUM, SEMICOLON`
πληροφορία που δεν επαρκεί για τη μεταγλώττιση.

Έτσι, διαβιβάζονται επιπλέον πληροφορίες για τις συμβολοσειρές, που λέγονται **ιδιότητες**. Η πιο σημαντική ιδιότητα είναι η λεξική μονάδα:
`IF, LPAREN, <ID,i>, EQEQ_OP, <ID,j>, RPAREN, <ID,z>, EQ_OP, <NUM,0>, SEMICOLON, ELSE, <ID,z>, EQ_OP, <NUM,1>, SEMICOLON`

Κανονικές εκφράσεις

Οι **κανονικές εκφράσεις** αποτελούνται από σταθερές και τελεστές που αναπαριστούν σύνολα συμβολοσειρών και πράξεις μεταξύ αυτών.

Ορισμός 1 (Κανονικές Εκφράσεις)

Δοθέντος ενός πεπερασμένου αλφάβητου Σ, κ.ε. είναι οι **σταθερές**:

- $\emptyset$ που αναπαριστά το κενό σύνολο
- ϵ που αναπαριστά το σύνολο με την «κενή» συμβολοσειρά χωρίς χαρακτήρες
- a που αναπαριστά το σύνολο με τη συμβολοσειρά a, όπου $a \in \Sigma$

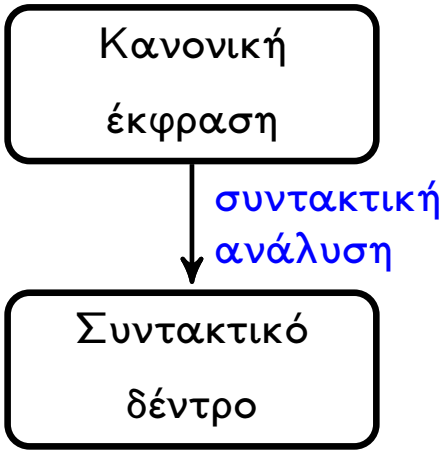
Δοθέντων κ.ε. R και S, ορίζουμε ότι οι ακόλουθες **πράξεις** είναι επίσης κ.ε.:

- *παράθεση* $R \cdot S$, το $\{\alpha\beta\colon\ \alpha$ στο σύνολο της R και β στο σύνολο της $S\}$
- *διάζευξη* $R|S$, την ένωση του συνόλου της R με το σύνολο της S
- *Kleene star* R^*, το ελάχιστο υπερσύνολο του συνόλου της R, που περιλαμβάνει την ϵ και είναι κλειστό ως προς την παράθεση

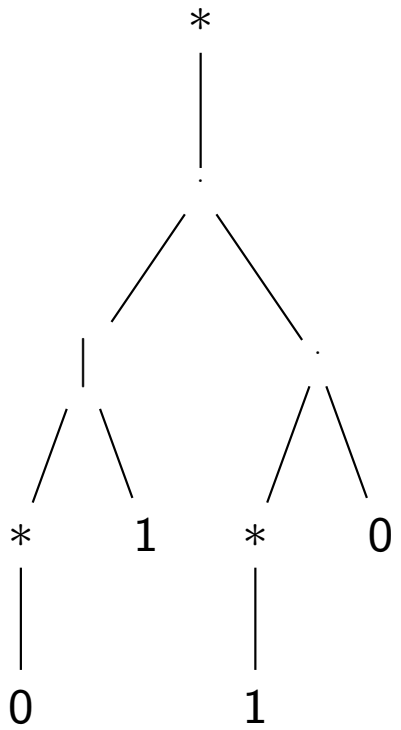
Κανονικές εκφράσεις

- Η Kleene star R^* εκφράζει το σύνολο όλων των συμβολοσειρών που σχηματίζονται με παράθεση πεπερασμένου αριθμού (ή μηδέν) συμβολοσειρών της R.
  Για παράδειγμα:
  $\{\text{“}ab\text{”}, \text{“}c\text{”}\} = \{\epsilon, \text{“}ab\text{”}, \text{“}c\text{”}, \text{“}abab\text{”}, \text{“}abc\text{”}, \text{“}cab\text{”}, \text{“}cc\text{”}, \text{“}ababab\text{”}, \ldots\}$
- **Προτεραιότητα τελεστών**: για την αποφυγή χρήσης παρενθέσεων υποθέτουμε ότι η Kleene star έχει την υψηλότερη προτεραιότητα και ακολουθούν κατά σειρά η παράθεση και η διάζευξη.
  Για παράδειγμα:
  $a \mid b \cdot c^* = a \mid (b \cdot (c^*))$
- Άλλα παραδείγματα:
  1. Η $a \mid b^*$ εκφράζει το $\{\epsilon, \text{“}a\text{”}, \text{“}b\text{”}, \text{“}bb\text{”}, \text{“}bbb\text{”}, \ldots\}$
  2. Η $(a \mid b)^*$ εκφράζει το $\{\epsilon, \text{“}a\text{”}, \text{“}b\text{”}, \text{“}aa\text{”}, \text{“}ab\text{”}, \text{“}ba\text{”}, \text{“}bb\text{”}, \text{“}aaa\text{”}, \ldots\}$
  3. Η $ab\,^*(c \mid \epsilon)$ εκφράζει τις συμβολοσειρές που αρχίζουν από “a”, περιλαμβάνουν 0 ή περισσότερα “b” και προαιρετικά καταλήγουν σε “c”, δηλ. το σύνολο $\{\text{“}a\text{”}, \text{“}ac\text{”}, \text{“}ab\text{”}, \text{“}abc\text{”}, \text{“}abb\text{”}, \text{“}abbc\text{”}, \ldots\}$

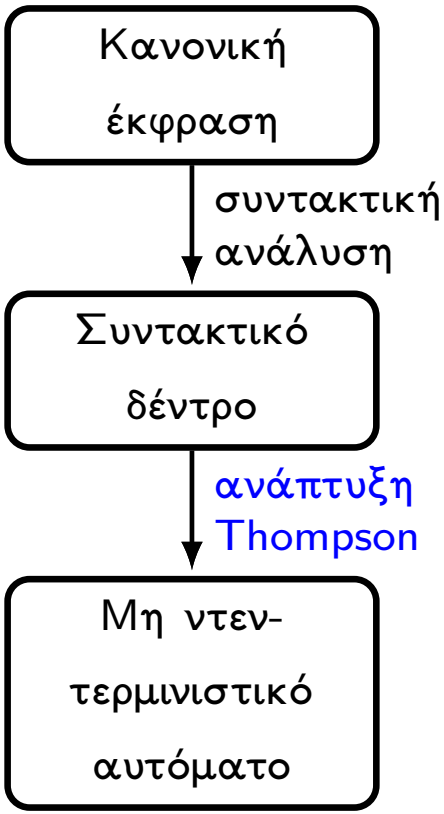
Λεξική Ανάλυση και Αυτόματα



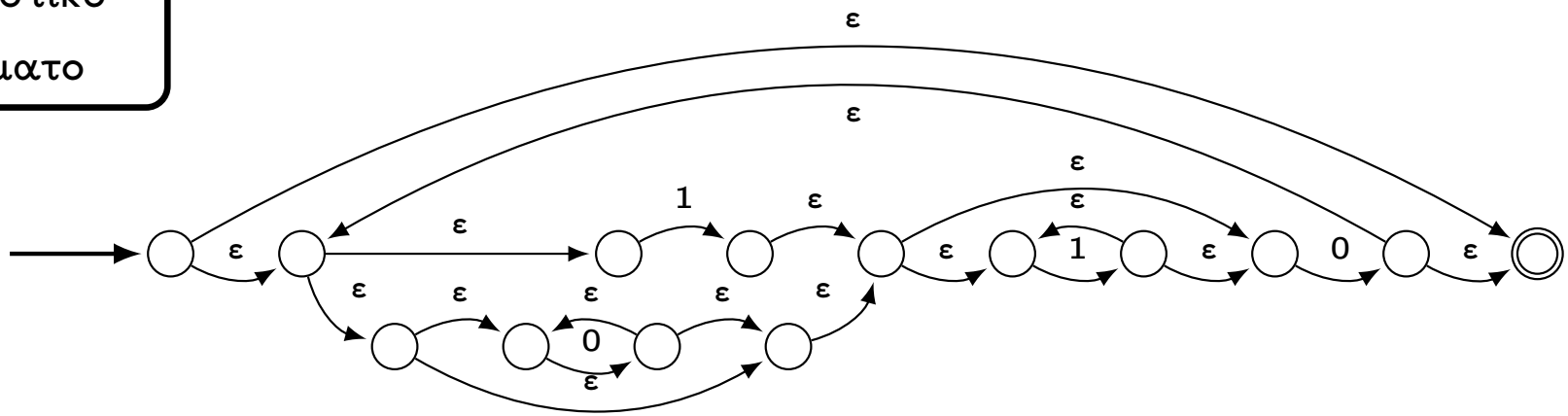
$((0^*|1)1^*0)^*$



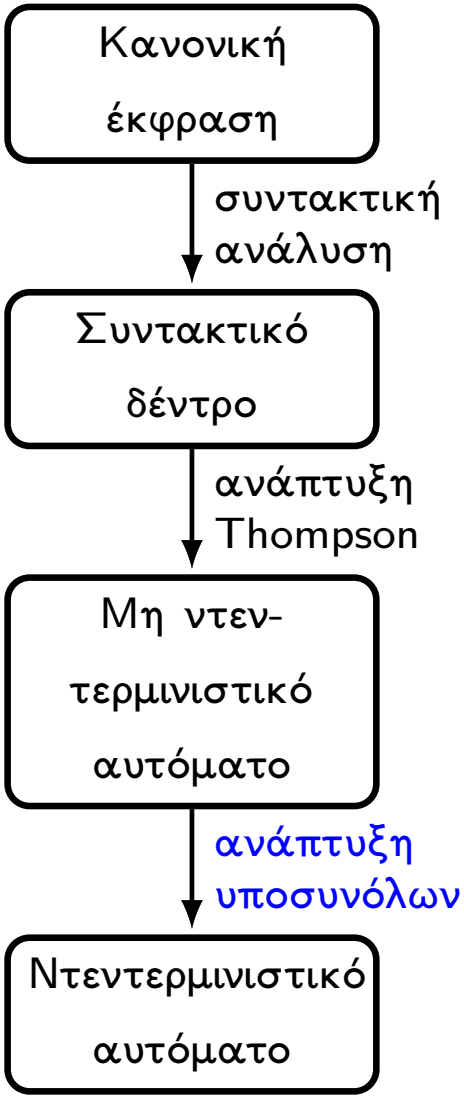
Λεξική Ανάλυση και Αυτόματα



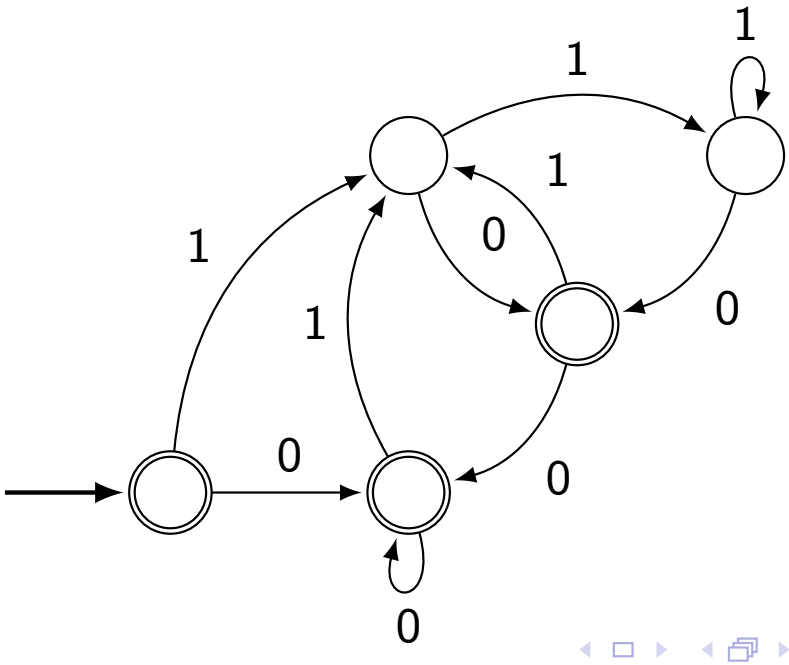
$$((0^*|1)1^*0)^*$$



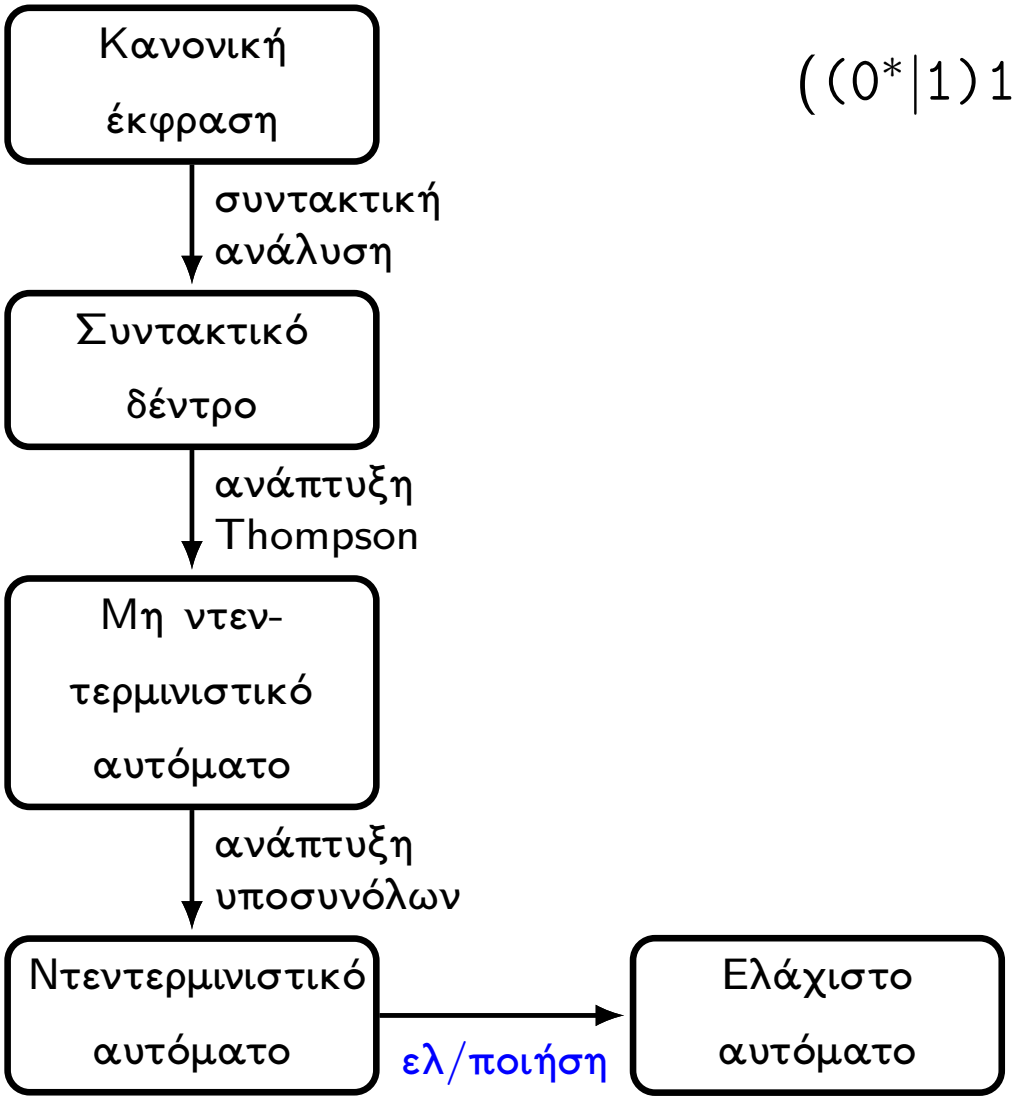
Λεξική Ανάλυση και Αυτόματα



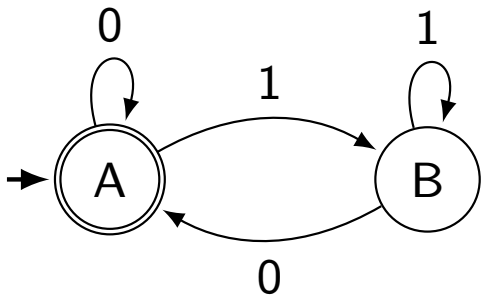
$$((0^*|1)1^*0)^*$$



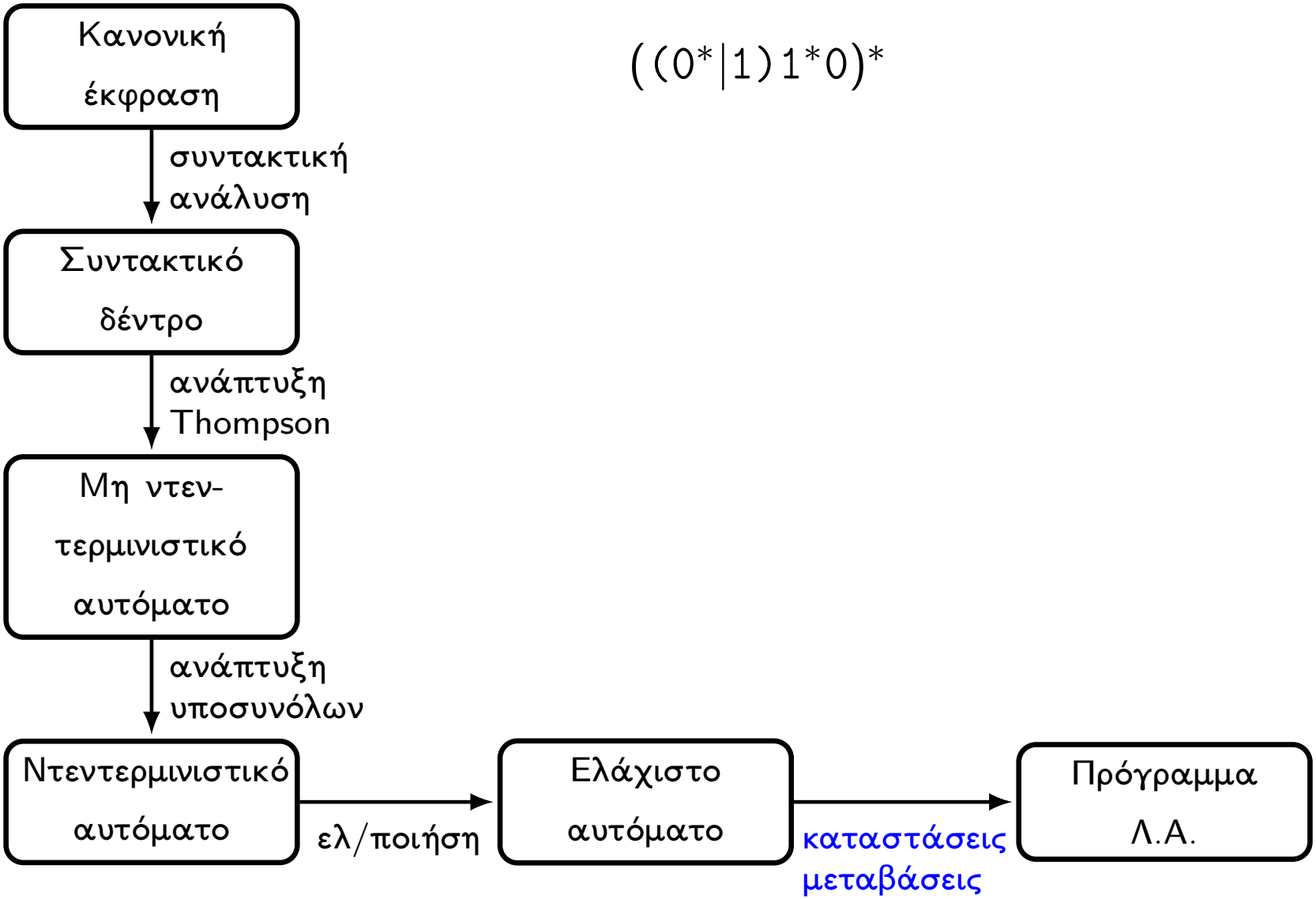
Λεξική Ανάλυση και Αυτόματα



$$((0^*|1)1^*0)^*$$

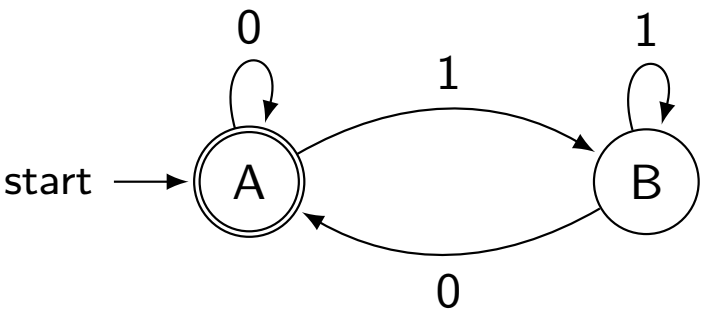


Λεξική Ανάλυση και Αυτόματα



$$((0^*|1)1^*0)^*$$

Λεξική Ανάλυση και Αυτόματα




```
typedef enum {A,B} STATES;
STATES state;
char c;
state = A;
while ((c = getc()) != EOF) {
        switch (state) {
        case A:
                if (c == '0') {
                        state = A; }
                else if (c == '1') {
                        state = B; }
                else {
                        printf ("ERROR\n");
                        exit (1); }
        break;
        case D:
                if (c == '0') {
                        state = A; }
                else if (c == '1') {
                        state = B; }
                else {
                        printf ("ERROR\n");
                        exit(1); }
        break; }
        if (c == EOF && state == A) {
                printf ("ACCEPTED\n");
        } else {
                printf ("NOT_ACCEPTED\n");
        }
}
```

Λεξική Ανάλυση και Αυτόματα

Οι αλγόριθμοι διδάχθηκαν στη Θεωρία Γλωσσών & Αυτομάτων.
Αναλυτική περιγραφή και παραδείγματα υπάρχουν στις εξής πηγές:

Αλγόριθμος	**Πηγή**
Συντακτική ανάλυση κανονικών εκφράσεων	*Μεταγλωττιστές Γλ. Προγραμματισμού* των Κ.Λάζου, Π.Κατσαρού, Ζ.Καραϊσκου: Αλγόριθμος ενότητας 2.3.4 και Παράδειγμα 2.3.3
Ανάπτυξη Thompson	*Μεταγλωττιστές Γλ. Προγραμματισμού* των Κ.Λάζου, Π.Κατσαρού, Ζ.Καραϊσκου: Αλγόριθμος ενότητας 2.3.5 και Παράδειγμα 2.3.4 (προτείνεται άσκηση με το JFLAP)
Ανάπτυξη υποσυνόλων	*Μεταγλωττιστές Γλ. Προγραμματισμού* των Κ.Λάζου, Π.Κατσαρού, Ζ.Καραϊσκου: Αλγόριθμος ενότητας 2.3.8 και παραδείγματα 2.3.6, 2.3.7, 2.3.8 (προτείνεται άσκηση με το JFLAP)
Ελαχιστοποιήση αυτόματου	Σημειώσεις που βρίσκονται στον ΠΗΛΕΑ

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Οι **δεσμευμένες** λέξεις προορίζονται για ειδική χρήση και δε μπορούν να χρησιμοποιηθούν από τους προγραμματιστές.
- Οι **λέξεις κλειδιά** προορίζονται για ειδική χρήση σε συγκεκριμένο πλαίσιο (πρόταση) και δεν είναι απαραίτητα δεσμευμένες.
- Ο αριθμός των δεσμευμένων λέξεων κυμαίνεται από 0 για τη βασική έκδοση της `Prolog` μέχρι και 400 για την `COBOL`, αλλά δε σχετίζεται με την εκφραστική ισχύ της ΓΠ.

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Κάποιες ΓΠ υιοθετούν τη **διάκριση πεζών - κεφαλαίων**:
  1. Για να επιβάλλουν συμβάσεις συγγραφής κώδικα (π.χ. στη Java τα ονόματα κλάσεων αρχίζουν με κεφαλαίο, σε αντίθεση με τα ονόματα μεταβλητών και συναρτήσεων).
  2. Για να απαλείψουν την ασάφεια και πιθανή σύγχυση, όταν γίνεται μη συνεπής χρήση πεζών - κεφαλαίων σε ονόματα.
- Στις περισσότερες ΓΠ χρησιμοποιείται **ένα κενό** για το διαχωρισμό των λεξικών μονάδων. Περισσότερα από ένα κενά δεν επηρεάζουν τη σύνταξη (με εξαίρεση τις `Haskell`, `Python` και τις πρώτς εκδόσεις της `FORTRAN`), αλλά βοηθάνε στην αναγνωσιμότητα του προγράμματος.

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Όλες οι ΓΠ διαθέτουν χαρακτήρες με σημασία:
  1. Διαχωρισμού εντολών, που υποδεικνύουν τα όρια μεταξύ δύο διαφορετικών εντολών
  2. Τερματισμού εντολών, που υποδεικνύουν το τέλος μιας εντολής
  3. Συνέχισης σε νέα γραμμή, που διαφέρει από το χαρακτήρα αλλαγής γραμμής
- Στις περισσότερες ΓΠ για το διαχωρισμό/τερματισμό εντολών χρησιμοποιείται το “;” ή ο χαρακτήρας αλλαγής γραμμής ή η “.”
- Κάποιες ΓΠ διαθέτουν επιπλέον χαρακτήρα (συνήθως το “;”) για διαχωρισμό εντολών που βρίσκονται στην ίδια γραμμή.
- Διαχωριστές λεξικών μονάδων: π.χ. στη C το “,” ως διαχωριστής σε εκφράσεις, αλλά και οι “(”, “)”, “[”, “]”, “:” καθώς και 15 απλοί τελεστές και 21 τελεστές με 2 χαρακτήρες.

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Τα σχόλια στις ΓΠ διακρίνονται ανάλογα με:
  1. το στυλ σε inline ή σε block σχόλια
  2. το πώς αναλύονται, σε αυτά που αγνοούνται, σε όσα αποθηκεύονται στη μνήμη και σε όσα παρεμβάλονται στο πρόγραμμα
  3. το αν υποστηρίζεται ένθεση (nesting) ή όχι
- Τα inline σχόλια είναι αυτά που στις περισσότερες περιπτώσεις τερματίζουν με χαρακτήρα αλλαγής γραμμής, ενώ αρχίζουν με συγκεκριμένο χαρακτήρα ή ακολουθία χαρακτήρων.
- Τα block σχόλια χρησιμοποιούνε συγκεκριμένο χαρακτήρα ή ακολουθία χαρακτήρων για άνοιγμα/κλείσιμο, ενώ οι χαρακτήρες αλλαγής γραμμής και κενό δε λογίζονται ως διαχωριστές.

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Μία **τιμή (literal)** αναπαριστά ένα δεδομένο. Οι ΓΠ διαθέτουν ατομικές τιμές για την αναπαράσταση ακεραίων, αριθμών κινητής υποδιαστολής, συμβολοσειρών, τιμών αληθείας (booleans), ενώ κάποιες ΓΠ υποστηρίζουν τιμές τύπων απαρίθμησης και σύνθετες τιμές όπως πίνακες, εγγραφές και αντικείμενα.
- Οι μεταβλητές/σταθερές είναι σύμβολα στα οποία εκχωρείται μία τιμή από ένα σύνολο πιθανών τιμών. Οι σταθερές δε μπορούν να αλλάξουν τιμή. Οι τιμές χρησιμοποιούνται για την αρχικοποιήση μεταβλητών.
- Σε κάποιες αντικειμενοστρεφείς ΓΠ (π.χ. JavaScript) τα αντικείμενα εκφράζονται από τιμές και οι μέθοδοι από τιμές συνάρτησης, π.χ.:

```
{"cat", "dog"}
{ name:"cat", length: 57}
```

Χαρακτηριστικά λεξικής δομής Γ.Π.

- Τα **ονόματα (identifiers)** χρησιμοποιούνται για την ταυτοποιήση διαφόρων οντοτήτων, περιλαμβανομένων των μεταβλητών, των τύπων, των ετικετών (labels), των υποπρογραμμάτων και των πακέτων.
- Συνήθως επιβάλονται περιορισμοί στο ποιοι χαρακτήρες εμφανίζονται σε ονόματα. Π.χ. στις πρώτες εκδόσεις της C και της C++ τα ονόματα μπορεί να είναι ακολουθίες από έναν ή περισσότερους ASCII χαρακτήρες, ψηφία (εκτός της πρώτης θέσης) και underscores.
- Μεταγενέστερες εκδόσεις αυτών των ΓΠ και οι περισσότερες σύγχρονες ΓΠ επιτρέπουν χαρακτήρες Unicode, εκτός από τους χαρακτήρες κενό και αυτούς που χρησιμοποιούνται για τους τελεστές.

Σύγκριση λεξικής δομής Γ.Π. 1

	Σύνολο χαρακτήρων	Διάκριση πεζών – κεφαλαίων	Κενά & αλλαγή γραμμής
FORTRAN		Όχι	Fortran 77: οι 6 πρώτες στήλες κενά (εξαιρούνται labels, σχόλια) Fortran 90: υποστηρίζεται και η ελεύθερη σύνταξη
C	91 single byte, 7 multiple byte	Ναι	
Pascal		Όχι	
Smalltalk		Ναι	
Java	Unicode	Ναι	
JavaScript	Unicode	Ναι, με κάποιες εξαιρέσεις για κώδικα client-side.	Αγνοούνται, εκτός αν περιέχονται σε συμβ/ρές ή σε κανονικές εκφράσεις.

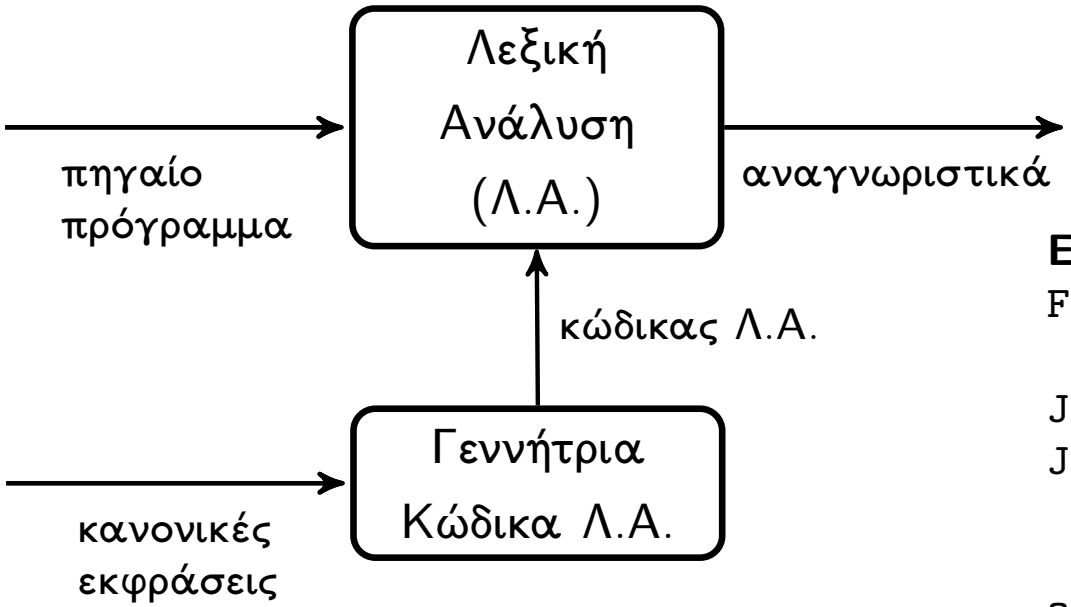
Σύγκριση λεξικής δομής Γ.Π. 2

	Γραμμές εντολών	Σχόλια	Τιμές (literals)
FORTRAN			
C			
Pascal			
Smalltalk			
Java			
JavaScript	Διαχωρίζονται με ';' (προαιρετικό αν υπάρχει αλλαγή γραμμής).	`/* Comment... multiple lines */` `//Comment... one line`	numbers, strings, booleans, functions, objects, arrays, null, undefined, regular expressions

Σύγκριση λεξικής δομής Γ.Π. 3

	Ονόματα (identifiers)	Δεσμευμένες λέξεις
FORTRAN		Οι λέξεις κλειδιά δεν είναι δεσμευμένες.
C		Όλες οι λέξεις κλειδιά (32).
Pascal		
Smalltalk		Οι λέξεις κλειδιά (6 ψευδομεταβλητές).
Java		Όλες οι λέξεις κλειδιά (50) και 2 που πλέον δε χρησιμοποιούνται.
JavaScript		Λέξεις κλειδιά (59) και προκαθορισμένες καθολικές μεταβλητές και συναρτήσεις (30).

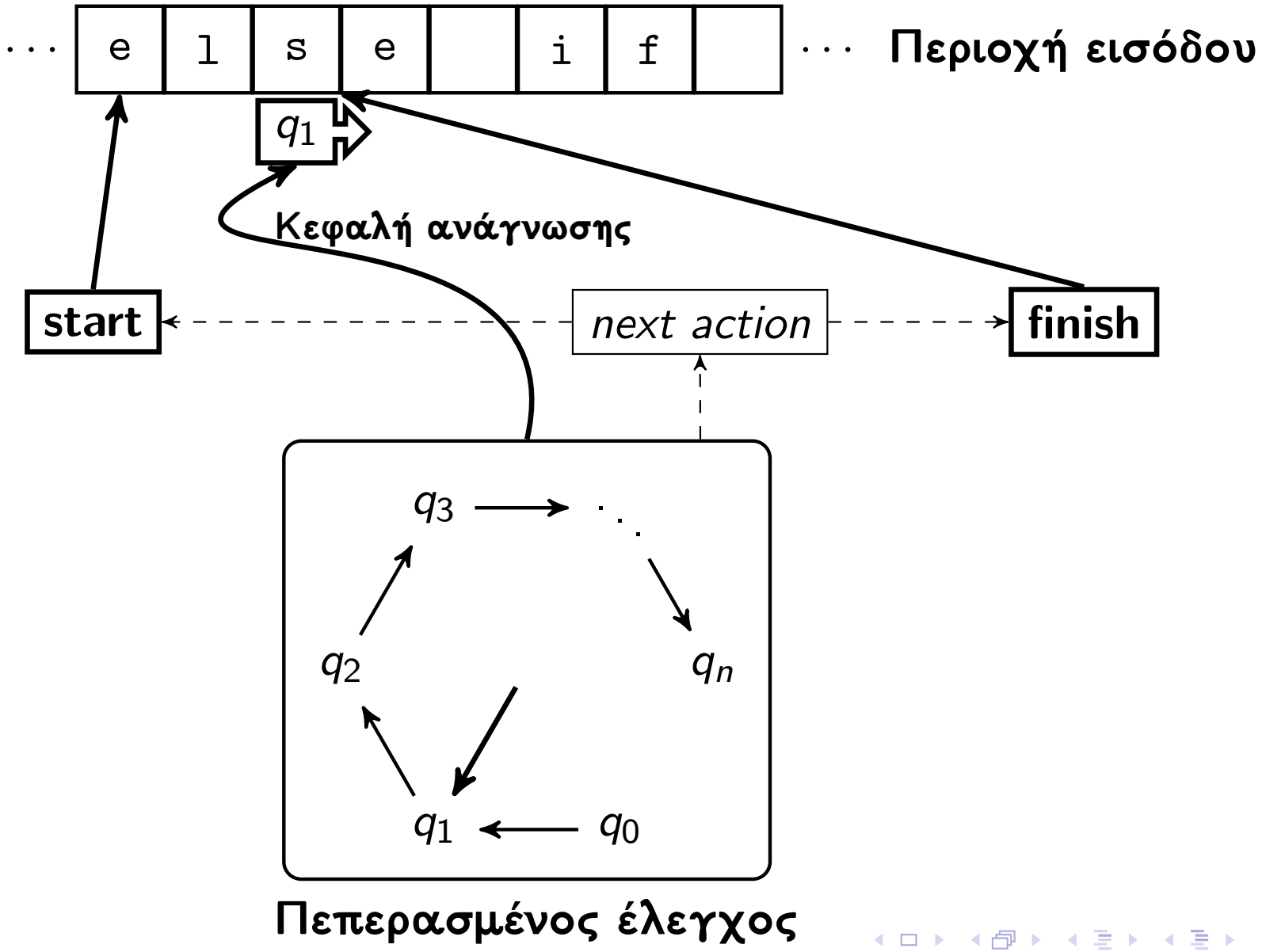
Γεννήτριες λεξικής ανάλυσης (scanner generators)



Εργαλείο Γλ. Προγραμματισμού

Flex	C/C++ (flex.sourceforge.net)
JFlex	Java (jflex.de)
JLex	Java, Unicode (www.cs.princeton.edu /~appel/modern/java/JLex/)
Quex	C/C++, Unicode (quex.sourceforge.net)
JavaCC	Java, Unicode, είναι και parser generator (javacc.java.net)

Λειτουργία Λεξικού Αναλυτή



Κανονικές εκφράσεις στο Flex

Καν. έκφραση	Περιγραφή	Παράδειγμα
"..."	Αναγνωρίζεται η ακολουθία χαρακτήρων ανάμεσα στα "και "	"a+b"
[...]	Χαρακτήρας που αναφέρεται/περιλαμβάνεται στα [και]	[A-Za-z0-9]
[^p]	Χαρακτήρας που δεν αναφέρεται/περιλαμβάνεται στα [και]	[^ab]
.	Οποιοσδήποτε χαρακτήρας εκτός από τη νέα γραμμή	
p*	Μηδέν ή περισσότερες φορές συμβ/σειρά που εκφράζει η κ.ε. p	abc*
p|q	Συμβ/σειρά που εκφράζει η κ.ε. p ή η κ.ε. q	a|b
p+	Μία ή περισσότερες φορές συμβ/σειρά που εκφράζει η κ.ε. p	a(bc)+
p?	Προαιρετική εμφάνιση συμβ/σειράς που εκφράζει η κ.ε. p	a(bc)?
p{n,m}	Συμβ/σειρά που η κ.ε. p επαναλαμβάνεται από *n* μέχρι *m* φορές	a{1,5}
^p	Συμβ/σειρά της κ.ε. p μόνο μετά από αλλαγή γραμμής	
p$	Συμβ/σειρά της p ακολουθούμενη από αλλαγή γραμμής	
p/q	Συμβ/σειρά της p ανν ακολουθείται από συμβ/σειρά της q	ab/cd
/c	Ο χαρακτήρας c χωρίς ειδική σημασία	

Περιγραφή Λ.Α. του Flex: παράδειγμα

```
%{
        # include <stdio.h>
%}
        ends_with_a     .*a\n
        begins_with_a   a.*\n
%%
        {ends_with_a}           ECHO;
        {begins_with_a}         ECHO;
        .*\n;
%%
void main()
        {
          yylex();
        }
```

Σύνταξη περιγραφής Λ.Α. για το Flex

```
%{
/* Κώδικας που πρέπει να περιληφθεί στη Λ.Α. */
%}
/* Ορισμοί κανονικών εκφράσεων */
%%
/* Κανονικές εκφράσεις και ενέργειες */
```

p_1	{ ενέργεια 1 }
p_2	{ ενέργεια 2 }
...	...
p_n	{ ενέργεια n }

```
%%
/* Βοηθητικές συναρτήσεις */
```

- Αν κάποια συμβολοσειρά παράγεται από περισσότερες της μιας κ.ε., τότε η Λ.Α. του Flex την ταυτίζει με την κ.ε. που αναφέρεται πρώτη.
- Αν κάποια συμβολοσειρά παράγεται από μία κ.ε. και ταυτόχρονα είναι μέρος μεγαλύτερης λεξικής μονάδας, τότε η Λ.Α. του Flex την αναγνωρίζει ως μέρος της μεγαλύτερης λεξικής μονάδας.

Ψευδομεταβλητές και συναρτήσεις του Flex

	Περιγραφή
`YYTYPE yylval`	...
`char* yytext`	...
`int yyleng`	...
`FILE* yyin`	...
`FILE* yyout`	...
`int yylex()`	...
`register int input()`	...
`void yymore()`	...
`void yyless(int n)`	...
`int yywrap()`	...
`ECHO`	...

Μελέτη πηγών για Λ.Α. - Πρακτική άσκηση με το Flex

- Λεξική Ανάλυση: *Μεταγλωττιστές Γλ. Προγραμματισμού* των Κ.Λάζου, Π.Κατσαρού, Ζ.Καραϊσκου - Κεφάλαιο 2
- Οδηγίες χρήσης του `Flex`: *Μεταγλωττιστές Γλ. Προγραμματισμού* των Κ.Λάζου, Π.Κατσαρού, Ζ.Καραϊσκου - Ενότητα 2.4
- Εργαστηριακό υλικό για το `Flex` στον ΠΗΛΕΑ: περιλαμβάνει το εργαλείο και δύο παραδείγματα Λ.Α.

Οδηγίες χρήσης εργαστηριακού υλικού για το Flex

1. Αποθηκεύστε τοπικά το συμπιεσμένο αρχείο που θα βρείτε στον ΠΗΛΕΑ και αποσυμπιέστε το σε ένα νέο φάκελο.
2. Αναγνώστε το αρχείο περιγραφής Λ.Α. `lex1` με τον editor της προτίμησής σας.
3. Εκτελέστε στην κονσόλα την ακόλουθη εντολή για να παραχθεί η Λ.Α.

   ```
   flex lex1
   ```

4. Το πρόγραμμα Λ.Α. βρίσκεται στο `lexyy.c`. Μεταγλωττίστε το με έναν C compiler και συνδέστε το μεταγλωττισμένο με το `zyywrap.obj` που υπάρχει στο υλικό. Παράδειγμα μεταγλώττισης με τον gcc:

   ```
   gcc lexyy.c zyywrap.obj
   ```

5. Χρησιμοποιήστε το εκτελέσιμο που δημιουργείται - έστω `a.exe` - για την επεξεργασία του παραδείγματος `testlex1` ή άλλου κειμένου:

   ```
   a < testlex1
   ```

6. Επαναλάβετε τα βήματα 2-5 για να κατασκευάσετε την πιο προηγμένη Λ.Α. του αρχείου `lex2`. Χρησιμοποιήστε το παράδειγμα `testlex2`.

Οδηγίες χρήσης εργαστηριακού υλικού για το Flex

Παρατηρήσεις:

- Η διαδικασία έχει δοκιμαστεί με τους C compilers gcc και cc του Microsoft Visual Studio.
- Έχουν παρατηρηθεί ασυμβατότητες του κώδικα που παράγει το Flex με τις πιο πρόσφατες εκδόσεις του Microsoft Visual Studio. Αν εμφανιστούν λάθη κατά τη μεταγλώττιση, δοκιμάστε μια παλαιότερη έκδοση του Microsoft Visual Studio ή μια νεώτερη έκδοση του Flex.
- Το αρχείο `zyywrap.obj` έχει δημιουργηθεί σε Η/Υ 32 bit Intel συμβατό. Αν χρησιμοποιήτε διαφορετική πλατφόρμα ή διαφορετικό compiler θα χρειαστεί πιθανότητα να ξαναδημιουργηθεί με μεταγλώττιση του `zyywrap.c` (συμπεριλαμβάνεται στο υλικό).
- Προτείνεται να αυτοματοποιήσετε την εκτέλεση των εντολών με ένα `Makefile` ή με ένα batch file.
- Ενθαρύνεται η χρήση σύγχρονων εργαλείων αντίστοιχων του Flex για όσους π.χ. θέλουν να γράψουν Λ.Α. σε Java.

Τέλος ενότητας

Επεξεργασία: Εμμανουέλα Στάχτιαρη
Θεσσαλονίκη, 23/07/2014