

Ενότητα 2: Λεκτική Ανάλυση

(Εισαγωγή και ad-hoc scanning)

Μ.Στεφανιδάκης

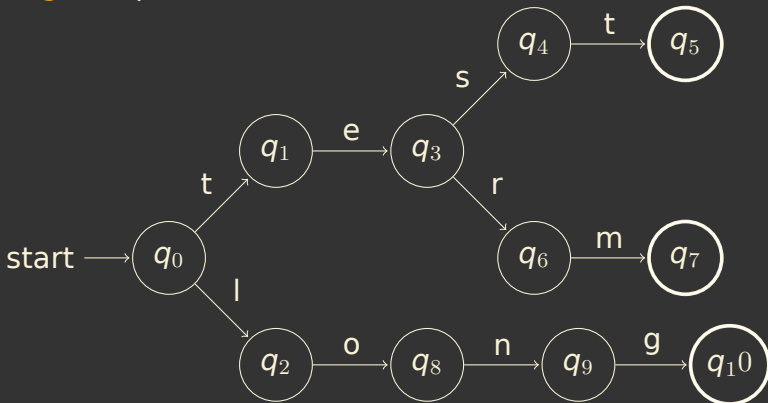
21-2-2018

Λεκτική Ανάλυση (Scanning)

- ▶ Ομαδοποίηση χαρακτήρων εισόδου σε σύμβολα (**tokens**) με συλλογική έννοια
- ▶ Σε κάθε σύμβολο αντιστοιχεί το κείμενο που έχει αναγνωριστεί (**lexeme**)
 - ▶ η ανάλυση δηλώσεων όπως **int i;** θα δώσει π.χ. τα tokens **K_INT** **IDENTIFIER** **K_SEMIC**
 - ▶ τα **K_INT** και **K_SEMIC** θα αντιπροσωπεύουν πάντα το κείμενο **int** και **;**
 - ▶ το **IDENTIFIER** αντιπροσωπεύει εδώ το **i** (σε άλλες δηλώσεις μεταβλητών θα αντιστοιχεί σε διαφορετικό κείμενο)
- ▶ Μείωση της πολυπλοκότητας του επόμενου σταδίου (συντακτική ανάλυση)

Λεκτική Ανάλυση: υλοποίηση

Η έννοια του **αυτόματου πεπερασμένων καταστάσεων**
Παράδειγμα: αναγνώριση των λέξεων **test**, **term** και **long** (και μόνον!)



Πεπερασμένα Αυτόματα (Finite Automata –FA)

Από τη “Θεωρία Υπολογισμού”: FA είναι μια πεντάδα $(Q, \Sigma, \delta, q_0, F)$, όπου

- ▶ Q ένα **πεπερασμένο** σύνολο καταστάσεων
 - ▶ συν μια κατάσταση σφάλματος q_e
- ▶ Σ ένα **πεπερασμένο** αλφάβητο
 - ▶ σύνολο χαρακτήρων εισόδου
- ▶ $\delta: Q \times \Sigma \rightarrow Q$ η συνάρτηση μετάβασης
 - ▶ από την τρέχουσα στην επόμενη κατάσταση, με την εμφάνιση ενός νέου χαρακτήρα εισόδου
 - ▶ αν δεν υπάρχει μετάβαση, τότε σφάλμα
- ▶ $q_0 \in Q$ η αρχική κατάσταση
- ▶ $F \subseteq Q$ το σύνολο των καταστάσεων αποδοχής
 - ▶ εάν βρισκόμαστε εδώ όταν τελειώσει η ανάλυση, τότε αποδεχόμαστε το κείμενο εισόδου

Υλοποίηση με πίνακα μεταβάσεων

επόμενος
χαρακτήρας
εισόδου



τρέχουσα
κατάσταση



	l	o	n	g	t	e	s	...
q0	q2				q1			
q1						q3		
q2		q8						
q3							q4	
q4					q5			
...								
...								

καταστάσεις αποδοχής: q5, q7, ...

Παράδειγμα υλοποίησης

Ο πίνακας μεταβάσεων για τα **test**, **term** και **long**
(π.χ. στο state **s0**, αν εμφανιστεί ο χαρακτήρας **t**, μετάβαση στο state **s1**)

```
td = { 's0':{ 't':'s1', 'l':'s2' },  
       's1':{ 'e':'s3' },  
       's2':{ 'o':'s4' },  
       's3':{ 's':'s5', 'r':'s6' },  
       's4':{ 'n':'s7' },  
       's5':{ 't':'s8f' },  
       's6':{ 'm':'s9f' },  
       's7':{ 'g':'s10f' },  
 }
```

Παράδειγμα υλοποίησης (2)

Ο πίνακας καταστάσεων αποδοχής (accepting states) και το αντίστοιχο σύμβολο (token)

```
ad = { 's8f': 'TEST_TOKEN',  
       's9f': 'TERM_TOKEN',  
       's10f': 'LONG_TOKEN'  
}
```

Παράδειγμα υλοποίησης (3)

Η συνάρτηση scan()

```
def scan(text,transition_table,accept_states):
```

```
    # initial state
```

```
    pos = 0
```

```
    state = 's0'
```

```
    while True:
```

```
        c = getchar(text,pos)    # get next char
```

```
        if c in transition_table[state]:
```

```
            state = transition_table[state][c]
```

```
            # set new state
```

```
            pos += 1    # advance to next char
```

```
            # check if new state is accepting
```

```
            if state in accept_states:
```

```
                return accept_states[state],pos
```

```
        else:    # no transition found
```

```
            return 'ERROR',pos
```


Επεκτάσεις προηγούμενου κώδικα

- ▶ Η σειρά σας: αναγνωρίστε τις λέξεις **today** και **tomorrow**
- ▶ Αν επιθυμούμε το **μέγιστο** δυνατό ταίριασμα;
 - ▶ Δεν γίνεται με τον προηγούμενο κώδικα...
 - ▶ Υπόδειξη: επιστρέφουμε μόνο όταν δεν υπάρχουν άλλες μεταβάσεις
 - ▶ Μόνο τότε ελέγχουμε αν βρισκόμαστε σε κατάσταση αποδοχής
 - ▶ Η σειρά σας: κατασκευάστε λεκτικό αναλυτή που αναγνωρίζει
 - ▶ ακεραίους (σειρά από ψηφία 0-9, **INT_TOKEN**)

Επεκτάσεις προηγούμενου κώδικα (2)

- ▶ Αν απαιτούνται **προαιρετικά** (optional) ταιριάσματα;
 - ▶ Υπόδειξη: Θυμόμαστε την τελευταία κατάσταση αποδοχής από την οποία έχουμε περάσει
 - ▶ Και αν φτάσουμε σε αδιέξοδο, επιστρέφουμε εκείνη
 - ▶ Η σειρά σας: κατασκευάστε λεκτικό αναλυτή που αναγνωρίζει
 - ▶ ακεραίους (σειρά από ψηφία 0-9, **INT_TOKEN**)
 - ▶ κλασματικούς (σειρά από ψηφία 0-9, μία τελεία και σειρά από ψηφία 0-9, **FLOAT_TOKEN**)