Αλγόριθμοι και Πολυπλοκότητα

7ο εξάμηνο

Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

http://www.corelab.ece.ntua.gr/courses/

8η ενότητα: Αυτόματα, Μηχανές Turing

Διδάσκοντες: Στάθης Ζάχος – Άρης Παγουρτζής

Επιμέλεια: Παναγιώτης Χείλαρης

Αυτόματα και γλώσσες

- Ταξινόμηση υπολογιστικών συνόλων (γλώσσες) ανάλογα με είδος αυτομάτου (αφηρημένης υπολογιστικής συσκευής) που μπορεί να τα αναγνωρίσει

Κατάταξη αυτομάτων ανάλογα με:

- Αιτιότητα (ντετερμ., μη ντετερμ.)
- Μέγεθος (πεπερασμένα, άπειρα)
- Είσοδος/Έξοδος (διακριτή, συνεχής)
- Γενικό ρολόι (παρ/λία: συγχρονισμένα, ασύγχρ.)
- Αριθμός καταστάσεων (πεπερασμ., άπειρος)
- Λειτουργία (acceptor, generator, transducer)

Strings

- Αλφάβητο συμβολων Σ
- String: πεπερασμένη ακολουθία συμβόλων
- Πράξη παράθεσης
- Κενό string (ε)

Τυπικές γραμματικές

ένα αλφάβητο V από *μη τερματικά* σύμβολα (μεταβλητές),

ένα αλφάβητο T από *τερματικά* σύμβολα (σταθερές), τ.ω. $V \cap T = \emptyset$,

ένα πεπερασμένο σύνολο P από *κανόνες παραγωγής*, δηλαδή διατεταγμένα ζεύγη (α, β) όπου $\alpha, \beta \in (V \cup T)^*$ και $\alpha \neq \varepsilon$
(Σύμβαση: γράφουμε $\alpha \to \beta$ αντί για (α, β)),

ένα *αρχικό σύμβολο* (ή αξίωμα) $S \in V$.

Παραγωγές, ακολουθία παραγωγών, γλώσσα που παράγεται από γραμματική

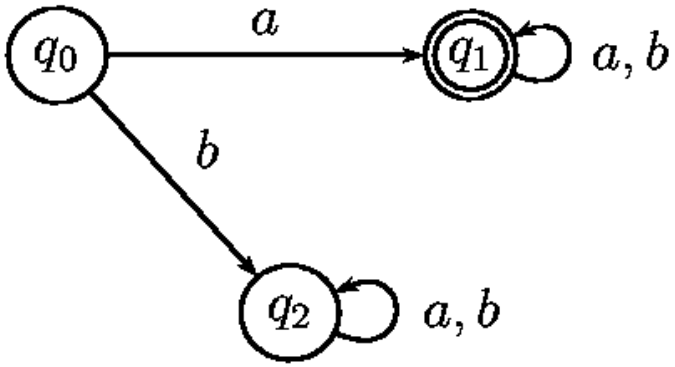
Ντετερμ. Πεπερασμ. Αυτόματα

- DFA τυπικά: $M = (Q, \Sigma, \delta, q_0, F)$.
  - Q: ένα πεπερασμένο σύνολο από καταστάσεις,
  - Σ: ένα αλφάβητο εισόδου $(\Sigma \cap Q = \emptyset)$,
  - $\delta : Q \times \Sigma \to Q$: η συνάρτηση μετάβασης,
  - $q_0 \in Q$: η αρχική κατάσταση,
  - $F \subseteq Q$: το σύνολο των τελικών καταστάσεων.

DFA: διάγραμμα ή πίνακας

DFA για γλώσσα $L_1 = \{w \in \Sigma^* \mid w \text{ αρχίζει από } a\}$

Διάγραμμα:

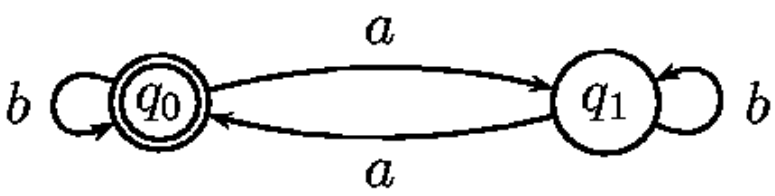


Πίνακας:

	a	b
q_0	q_1	q_2
(q_1)	q_1	q_1
q_2	q_2	q_2

DFA: διάγραμμα ή πίνακας (2)

$$L_2 = \{w \in \Sigma^* \mid w \text{ περιέχει άρτιο πλήθος από } a\}$$



	a	b
(q_0)	q_1	q_0
q_1	q_0	q_1

Αποδοχή σε DFA

Επέκταση συνάρτησης μετάβασης δ σε συμβολοσειρές:

$$\tilde{\delta} : Q \times \Sigma^* \to Q$$

$$\begin{cases} \tilde{\delta}(q, \varepsilon) = q \\ \tilde{\delta}(q, wa) = \delta(\tilde{\delta}(q, w), a) \end{cases}$$

(ορισμός σύμφωνα με σχήμα πρωταρχικής αναδρομής)

- Ένα DFA αποδέχεται το string $w \in \Sigma^*$ ανν $\tilde{\delta}(q_0, w) \in F$
- Ένα DFA M αποδέχεται τη γλώσσα $L(M) = \{w \mid \tilde{\delta}(q_0, w) \in F\}$
- Η γλώσσα L λέγεται *κανονική* (*regular*) ανν $\exists$ FA $M : L = L(M)$

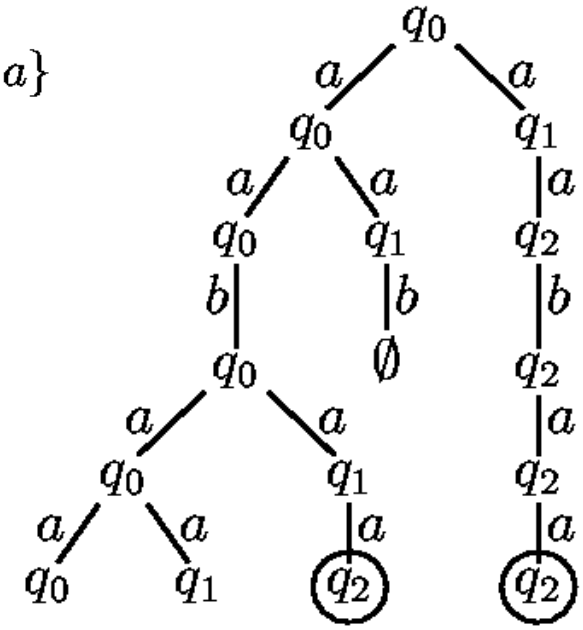
Μη ντετερμινιστικά FA (NFA)

- Πιθανόν περισσότερες από μία (ή καμία) επόμενες καταστάσεις για το ίδιο σύμβολο εισόδου:

$$L_4 := \{w \in \Sigma^* \mid w \text{ περιέχει δύο συνεχόμενα } a\}$$

	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

- Δένδρο υπολογισμού για είσοδο *aabaa:*



Τυπικός ορισμός NFA

Ίδιος με του DFA εκτός από την συνάρτηση μετάβασης που πλέον έχει πεδίο τιμών το δυναμοσύνολο των καταστάσεων:

- $\delta : Q \times \Sigma \to \mathrm{Pow}(Q)$: η συνάρτηση μετάβασης

Παρομοίως όπως στα DFA επεκτείνουμε την παραπάνω συνάρτηση ούτως ώστε να δέχεται ως είσοδο συμβολοσειρές: $\tilde{\delta} : Q \times \Sigma^* \to \mathrm{Pow}(Q)$

Αποδοχή σε NFA

- Ένα NFA αποδέχεται το string $w \in \Sigma^*$ ανν $\tilde{\delta}(q_0, w) \cap F \neq \emptyset$
- Ένα NFA M αποδέχεται τη γλώσσα $L(M) = \{w | \tilde{\delta}(q_0, w) \cap F \neq \emptyset\}$

- Ή ισοδύναμα, έχουμε αποδοχή αν υπάρχει φύλλο στο δένδρο υπολογισμού επιγεγραμμένο με τελική κατάσταση

Ισοδυναμία NFA, DFA

- Τα DFA είναι προφανώς υποπερίπτωση των NFA, αλλά ενώ τα NFA φαίνονται ισχυρότερα ως προς τις δυνατότητες αναγνώρισης, στην πραγματικότητα δεν είναι:

Θεώρημα Rabin-Scott:
Έστω *M* NFA, υπάρχει DFA *M'* με $L(M) = L(M')$.

Απόδειξη θ. Rabin-Scott

Ορίζουμε ένα DFA με σύνολο καταστάσεων το δυναμοσύνολο καταστάσεων του NFA. Τυπικά:

Απόδειξη. Ορίζουμε το DFA $M' = (Q', \Sigma', q_0', F', \delta')$ όπου $Q' = \mathrm{Pow}(Q)$, $\Sigma' = \Sigma$, $q_0' = \{q_0\}$, $F' = \{R \in Q' | R \cap F \neq \emptyset\} = \bigcup_{R \cap F \neq \emptyset} (R \in Q')$ και τέλος $\delta'(R, a) = \delta^{\times}(R, a)$, όπου $R \in Q'$.

Μένει να αποδειχθεί ότι $L(M) = L(M')$ με την βοήθεια του ισχυρισμού: $\tilde{\delta}'(q_0', w) = \tilde{\delta}(q_0, w)$. (Αφήνεται ως άσκηση· χρησιμοποιήστε επαγωγή.) □

NFA με ε-κινήσεις (NFA_ε)

- Μπορούμε να επεκτείνουμε το NFA, ώστε να συμπεριλάβουμε στο πεδίο ορισμού της δ και το ε (κενή συμβολοσειρά). Αυτό πρακτικά σημαίνει ότι το αυτόματο μπορεί να αλλάζει κατάσταση χωρίς να διαβάζει μέρος της εισόδου, με μία ε-κίνηση, όπως λέγεται.

Παράδειγμα NFA$_\varepsilon$

$$L_5 := \{a^*b^*\} = \{a^n b^m \mid n, m \in \mathbb{N}\}$$

	a	b	ε
q_0	$\{q_0\}$	$\{q_1\}$	$\{q_1\}$
((q_1))	$\emptyset$	$\{q_1\}$	$\emptyset$

Τυπικός ορισμός NFA$_\varepsilon$

- Μόνη αλλαγή σε σχέση με NFA:
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathrm{Pow}(Q)$: η συνάρτηση μετάβασης

Ως ε-κλείσιμο: $Q \to \mathrm{Pow}(Q)$ ορίζουμε το

ε-κλείσιμο$(q) = \{p |$ τα p προσβάσιμα από το q μόνο με ε-κινήσεις$\}$

Κατά σύμβαση πάντα το ε-κλείσιμο(q) περιέχει το q

Αποδοχή σε NFA_ε

$$\tilde{\delta} : Q \times \Sigma^* \to \mathrm{Pow}(Q)$$

$$\begin{cases} \tilde{\delta}(q, \varepsilon) = \varepsilon\text{-κλείσιμο}(q) \\ \tilde{\delta}(q, wa) = \varepsilon\text{-κλείσιμο}(\{p \in \delta(r, a) \mid r \in \tilde{\delta}(q, w)\}) \end{cases}$$

- Ένα NFA_ε αποδέχεται το string $w \in \Sigma^*$ ανν $\tilde{\delta}(q_0, w) \cap F \neq \emptyset$
- Ένα NFA_ε M αποδέχεται τη γλώσσα $L(M) = \{w | \tilde{\delta}(q_0, w) \cap F \neq \emptyset\}$

Ισοδυναμία NFA_ε, NFA (DFA)

- Τελικά και τα NFA_ε αν και φαίνονται ισχυρότερα είναι ισοδύναμα με τα NFA (άρα και τα DFA)

$$\textit{Έστω } M \textit{ ένα } NFA_\varepsilon \textit{ τότε } \exists \text{ NFA } M' : L(M) = L(M')$$

$$\textit{Απόδειξη.} \quad \text{NFA } M' = (Q, \Sigma, q_0, F', \delta') \text{ όπου}$$

$$F' = \begin{cases} F \cup \{q_0\}, & \text{αν } \varepsilon\text{-κλείσιμο}(q_0) \cap F \neq \emptyset \\ F, & \text{ειδάλλως} \end{cases}$$

$$\delta'(q, a) = \tilde{\delta}(q, a)$$

Κανονικές παραστάσεις

- Πράξεις επί γλωσσών επί αλφαβήτου Σ
  - $L_1 L_2 := \{uv \mid u \in L_1 \wedge v \in L_2\}$: παράθεση
  - $L_1 \cup L_2 := \{w \mid w \in L_1 \vee w \in L_2\}$: ένωση
  - $L_1 \cap L_2 := \{w \mid w \in L_1 \wedge w \in L_2\}$: τομή
  - $L^0 := \{\varepsilon\}$, $L^{n+1} := LL^n$
  - $L^* := \bigcup_{n=0}^{\infty} L^n$: άστρο του Kleene

Επαγωγικός ορισμός

- Κανονική παράσταση (regular expression) είναι:

$\emptyset$: παριστάνει το κενό σύνολο·

a: παριστάνει το $\{a\}$, όπου $a \in \Sigma$·

ε: παριστάνει το $\{\varepsilon\}$·

$(r + s)$: παριστάνει το $R \cup S$,
όπου r, s κανονικές παραστάσεις

(rs): παριστάνει το RS, όπου r, s κανονικές παραστάσεις

r^*: παριστάνει το R^*, όπου r κανονική παράσταση

(r παριστάνει γλώσσα R, s παριστάνει γλώσσα S)

Παραδείγματα Regexp

$L_1 = \{w \in \Sigma^* \mid w \text{ αρχίζει με } a\}$

$L_2 = \{w \in \Sigma^* \mid w \text{ περιέχει ζυγό αριθμό από } a\}$

$L_3 = \{w \in \Sigma^* \mid w \text{ είναι παλινδρομική }\}$

$L_5 := \{a^*b^*\} = \{a^n b^m \mid n, m \in \mathbb{N}\}$

$L_1 = a(a+b)^*$

$L_2 = (b^*ab^*a)^*b^*$

L_3 δεν είναι δυνατόν να παρασταθεί με κανονική παράσταση

$L_4 = (a+b)^*aa(a+b)^*$ (τουλάχιστον δύο συνεχόμενα a)

$\overline{L_4} = (a+\varepsilon)(ba+b)^*$ (όχι συνεχόμενα a)

$L_5 = a^*b^*$

Ισοδυναμία Regexp, FA

- Θεώρημα: Μία γλώσσα *L* μπορεί να παρασταθεί με regexp ανν *L=L(M)* για κάποιο FA *M*.
- => επαγωγή στην δομή της regexp.
- <= R_{ij}^k σύνολο strings που οδηγούν από την κατάσταση q_i στην q_j χωρίς να περνούν από κατάσταση με δείκτη μεγαλύτερο του *k*

$$R_{ij}^0 = \begin{cases} \{a|\delta(q_i,a)=q_j\}, & \text{αν } i \neq j \\ \{a|\delta(q_i,a)=q_j\} \cup \{\varepsilon\}, & \text{αν } i = j \end{cases}$$

$$R_{ij}^k = R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \cup R_{ij}^{k-1}$$

$$L(M) = \bigcup_{q_j \in F} R_{1j}^n$$

Άλλα FA

- 2-way FA (κεφαλή κινείται δεξιά-αριστερά) ισοδύναμο με DFA (δύσκολη απόδειξη)
- Μηχανές με έξοδο:

- Μηχανή Moore, $\lambda: Q \to \Delta$.
- Μηχανή Mealey, $\lambda: Q \times \Sigma \to \Delta$.
- Πεπερασμένοι υπολογιστές (transducers), $\lambda: Q \times \Sigma \to \Delta^*$.

και οι τρεις παραπάνω τύποι ισοδύναμοι (εύκολη απόδειξη)

Ελαχιστοποίηση DFA (i)

- Συνέπεια θεωρήματος Myhill-Nerode: υπάρχει μοναδικό (εκτός μετονομασίας καταστάσεων) ελάχιστο DFA. Μέθοδος κατασκευής:

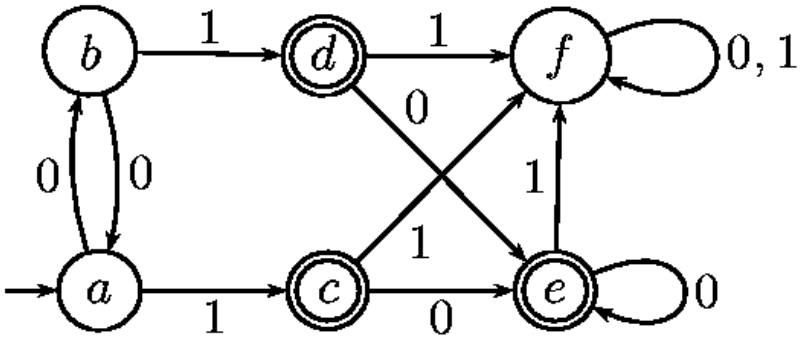
1. Εξαλείφουμε απρόσιτες καταστάσεις.
2. Συγχωνεύουμε ισοδύναμες καταστάσεις που δεν διακρίνονται με κανένα επόμενο string

Ελαχιστοποίηση DFA (ii)

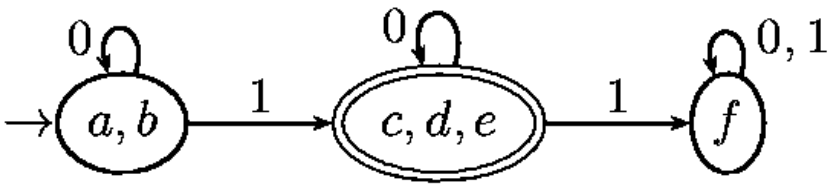
Το p είναι διακρίσιμο από το q εάν υπάρχει ένα x τέτοιο ώστε $\delta(p, x)$ είναι στο F και $\delta(q, x)$ δεν είναι ή αντίστροφα.

Φτιάχνουμε ένα πίνακα για να συγκρίνουμε κάθε ζεύγος καταστάσεων. Βάζουμε ένα X σε κάθε θέση του πίνακα κάθε φορά που ανακαλύπτουμε ότι δύο καταστάσεις δεν είναι ισοδύναμες. Αρχικά εγγράφουμε X σε όλα τα ζεύγη που προφανώς διακρίνονται γιατί η μία είναι τελική και η άλλη δεν είναι. Μετά προσπαθούμε να δούμε αν διακρίνονται δύο καταστάσεις, διότι από αυτές με ένα σύμβολο a οδηγούμαστε σε διακρίσιμες καταστάσεις. Επαναλαμβάνουμε την πιο πάνω προσπάθεια ώσπου να μην προστίθεται κανένα X πια στον πίνακα. Τα υπόλοιπα ζευγάρια είναι μη διακρίσιμα και συνεπώς συγχωνεύσιμα.

Παράδειγμα ελαχιστοποίησης



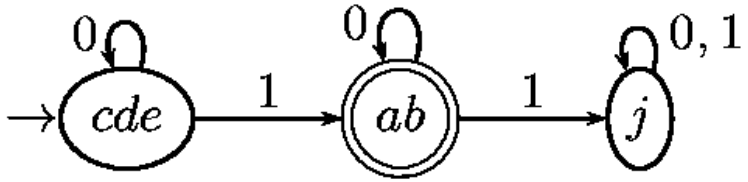
b					
ⓒ	X_1	X_1			
ⓓ	X_1	X_1			
ⓔ	X_1	X_1			
f	X_2	X_2	X_1	X_1	X_1
	a	b	ⓒ	ⓓ	ⓔ



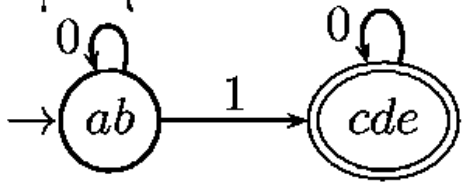
Μέθοδος Beckmann

1. Κατασκεύασε το καθρέφτισμα του FA: δηλαδή ανέστρεψε τη φορά των τόξων. Αντάλλαξε τελικές με αρχικές καταστάσεις.

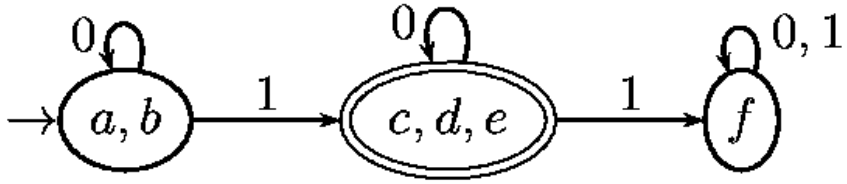
2. Κατασκεύασε DFA ισοδύναμο με το προκύπτον.



3. Ανακατασκεύασε το καθρέφτισμα.

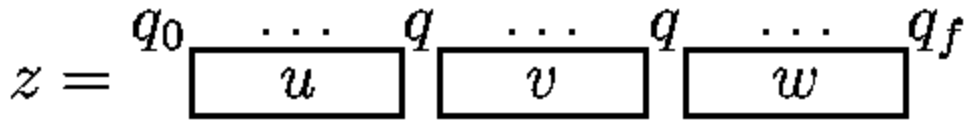


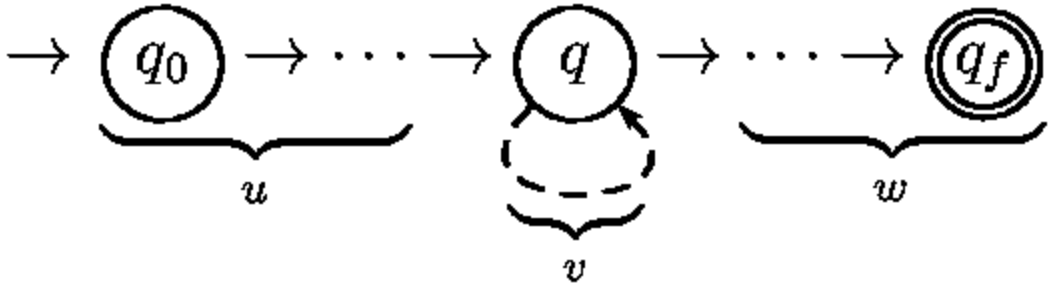
4. Ανακατασκεύασε ισοδύναμο DFA. Αυτό τώρα είναι το ελάχιστο DFA που βρήκαμε και παραπάνω, δηλ.



Pumping Lemma

Λήμμα 2.3.29 (Pumping Lemma). *Εάν L είναι regular τότε:*
$\exists n \in \mathbb{N}, \forall z \in L$ *με* $|z| \geq n, \exists u, v, w \in \Sigma^*$:
$[z = uvw \ \wedge |uv| \leq n \ \wedge |u| \geq 1 \wedge \forall i \in \mathbb{N} \ (uv^i w \in L)]$





Χρήση Pumping Lemma

- Για να δείξουμε ότι γλώσσα δεν είναι regular, π.χ.:

$$L = \{a^k b^k \mid k \in \mathbb{N}\}$$

1. Υποθέτουμε ότι L είναι regular
2. PL: $\exists n \in N$
3. Διαλέγουμε $z = a^n b^n$. Εντάξει επιλογή, διότι $z \in L$, $|z| = 2n \geq n$.
4. PL: z μπορεί να γραφεί: $z = uvw$ με $|uv| \leq n \ \wedge |u| \geq 1$, ώστε $u = a^l$ με $l \geq 1$.
5. Διαλέγουμε $i = 2$: $uvvw = a^{n+l} b^n \in L$.

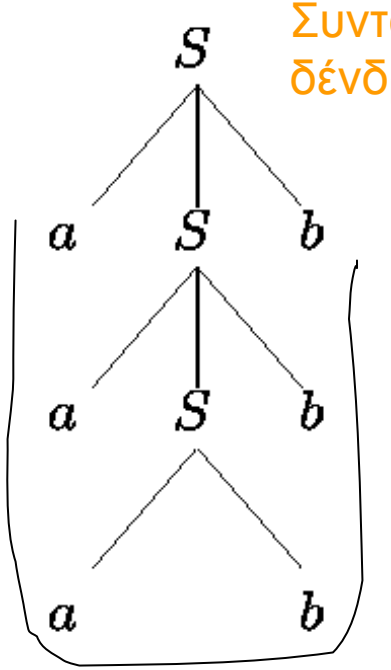
Άτοπο

Context free γραμματικές

Κανόνες της μορφής: A -> β όπου $\beta \in (V \cup T)^*$ και $A \in V$

$G_1: \quad V = \{S\},\ T = \{a, b\},\ P = \{S \to \varepsilon, S \to aSb\}$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$



Συντακτικό δένδρο

Φύλλωμα

Διφορούμενες γραμματικές

- *Διφορούμενη γραμματική* G: αν υπάρχουν δυο διαφορετικά δένδρα με το ίδιο φύλλωμα w στο L(G).
- *Εγγενώς διφορούμενη γλώσσα:* αν όλες οι γραμματικές που την παράγουν είναι διφορούμενες, π.χ.:

$$\{a^i b^j c^k \mid i = j \text{ ή } j = k\}$$

Απλοποίηση, κανονικές μορφές

- Κρατάμε μόνον χρήσιμα σύμβολα (προσβάσιμα, παραγωγικά) και εξαλείφουμε κανόνες A->B
- Κανονικές μορφές (απλή μορφή κανόνων):

Θεώρημα 2.4.9 (Κανονικής μορφής Chomsky, CNF). *Κάθε c.f. γλώσσα χωρίς το ε παράγεται από μια γραμματική στην οποία όλες οι παραγωγές είναι της μορφής* $A \to BC$ *ή* $A \to a$. *όπου* A, B, C *μεταβλητές και* a *τερματικό.*

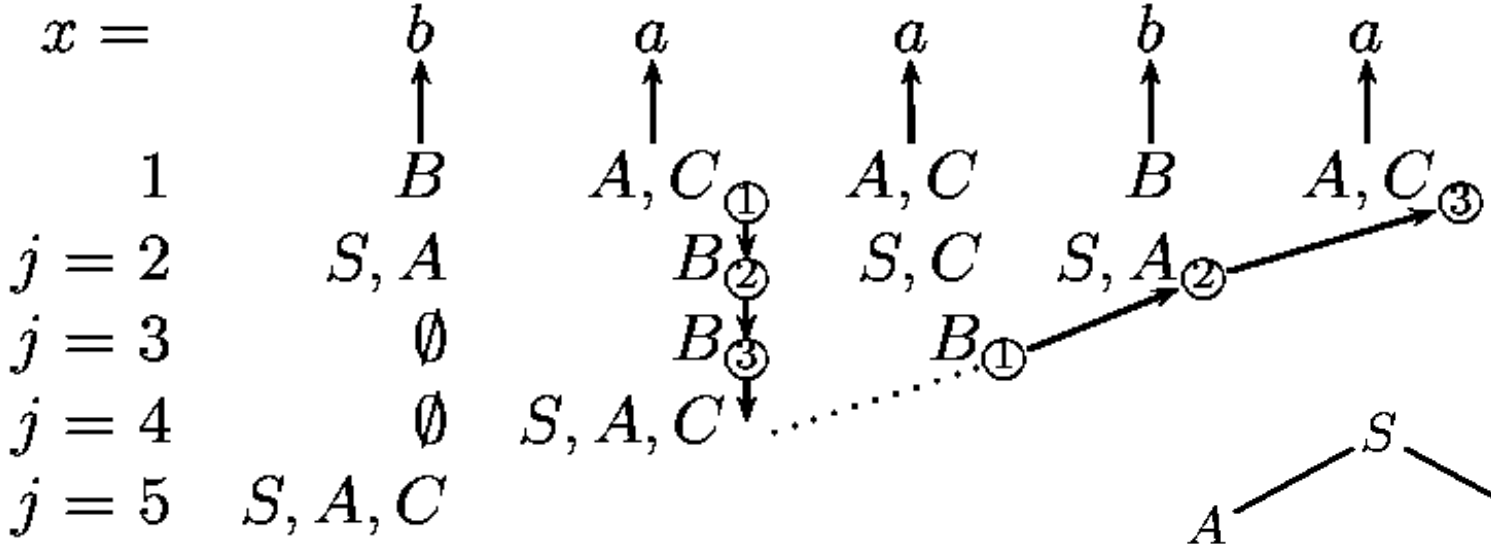
Θεώρημα 2.4.10 (Κανονικής μορφής Greibach, GNF). *Κάθε c.f. γλώσσα χωρίς το ε παράγεται από μια γραμματική στην οποία όλες οι παραγωγές είναι της μορφής* $A \to a\alpha$, *όπου* $\alpha \in V^*$, $a \in T$.

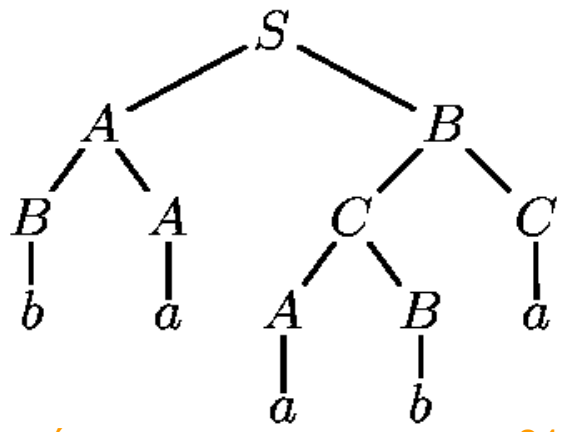
Αλγόριθμος CYK

function CYK(x: string): **boolean** *(* assumes Chomsky n.f. *)*
begin $n := |x|$
  for $i := 1$ **to** n **do**
    $V_i^1 := \{A \mid (A \to a) \in P \wedge (x)_i = a\}$;
  for $j := 2$ **to** n **do**
    for $i := 1$ **to** $n - j + 1$ **do**
    begin $V_i^j := \emptyset$;
      for $k := 1$ **to** $j - 1$ **do**
        $V_i^j := V_i^j \cup \{A \mid (A \to BC) \in P \wedge B \in V_i^k \wedge C \in V_{i+k}^{j-k}\}$
    end ;
  CYK $:= S \in V_1^n$
end

CYK, παράδειγμα

$$S \to AB \mid BC, \quad A \to BA \mid a, \quad B \to CC \mid b, \quad C \to AB \mid a$$





Αυτόματο στοίβας (PDA)

- Όπως το FA, αλλά επιπλέον έχει και στοίβα (μη φραγμένη μνήμη στην οποία έχουμε πρόσβαση με τις λειτουργίες push, pop)
- Μη ντετερμινιστικά PDA ισχυρότερα από τα ντετερμινιστικά
- Αποδοχή είτε σε τελική κατάσταση, είτε με κενή στοίβα

Ισοδυναμία PDA, CFG

Θεώρημα: L αποδεκτή από PDA ανν L είναι CFG.

Γενικές γραμματικές (τύπου 0)

Παραγωγές: $\alpha \to \beta$, με $\alpha \neq \varepsilon$

$$L = \{a^{2^n} \mid n \in N\}$$

$$S \to AaCB$$
$$CB \to E \mid DB$$
$$aE \to Ea$$
$$AE \to \varepsilon$$
$$aD \to Da$$
$$AD \to AC$$
$$Ca \to aaC$$

L γίνεται αποδεκτή από μηχανή Turing ανν παράγεται από γενική γραμματική

(τότε η L λέγεται και αναδρομικά αριθμήσιμη, recursively enumerable)

Γραμματικές με συμφραζόμενα

Παραγωγές: $\alpha \rightarrow \beta$, με $\alpha \neq \varepsilon$, $|\alpha| \leq |\beta|$

$1^n 0^n 1^n$

$S \rightarrow 1Z1$

$S \rightarrow 0 \mid 1Z0A$

$A0 \rightarrow 0A$

$A1 \rightarrow 11$

Linear Bounded Automaton: μη ντετ. μηχανή Turing της οποίας η κεφαλή είναι περιορισμένη να κινείται μόνον στο τμήμα της ταινίας που περιέχει την είσοδο.

L γίνεται αποδεκτή από LBA ανν παράγεται από γραμματική χωρίς συμφραζόμενα

Θεώρημα Ιεραρχίας

$$regular \underset{\neq}{\subset} context\ free \underset{\neq}{\subset} context\ sensitive \underset{\neq}{\subset} recursively\ enumerable$$

Κανονικές γραμματικές

Στις *κανονικές γραμματικές* όλοι οι κανόνες παραγωγής είναι της μορφής:

1. δεξιογραμμικοί (rightlinear): $A \to wB$, $A \to w$, όπου $w \in T^*$, ή
2. αριστερογραμμικοί (leftlinear): $A \to Bw$, $A \to w$, όπου $w \in T^*$.

Θεώρημα: Τα κανονικά σύνολα παράγονται από κανονικές (αριστερο- ή δεξιογραμμικές γραμματικές).

Μηχανή Turing (TM)

- Απλός ιδεατός υπολογιστής
- Διαθέτει *ταινία* άπειρου (δυνητικά) μήκους με *κύτταρα* (θέσεις) που περιέχουν 0 ή 1
- Βασικές λειτουργίες:
  - Διάβασε περιεχόμενο τρέχοντος κυττάρου
  - Γράψε 1 ή 0 στο τρέχον κύτταρο
  - Κάνε τρέχον το αμέσως αριστερά ή το αμέσως δεξιά κύτταρο

Λειτουργία Μηχανής Turing

- Πεπερασμένος αριθμός εσωτερικών καταστάσεων:

$$Q=\{q_0,q_1,\ldots q_k\}$$

- Πρόγραμμα: *συνάρτηση μετάβασης*

$$Q \times \Sigma \rightarrow Q \times \Sigma \cup \{L,R\}$$

Υπολογισμός του 2x με ΤΜ (i)

- Ο αριθμός x αναπαριστάται με x+1 ψηφία '1'
- Περιγραφή υψηλού επιπέδου:

```
αρχικοποίηση; (διαγραφή του πρώτου 1) while είσοδος<>0 do
begin
  διάγραψε ένα 1 από είσοδο;
  μετακίνησε κεφαλή δεξιά πέρα από είσοδο και έξοδο;
  πρόσθεσε δύο 1 στην έξοδο;
  μετακίνησε κεφαλή αριστερά πέρα από έξοδο και είσοδο
 end
```

Υπολογισμός του 2x με ΤΜ (ii)

$< q_0 \quad 1 \quad 0 \quad q_1 >$

$< q_1 \quad 0 \quad \mathrm{R} \quad q_2 >$

$< q_2 \quad 1 \quad 0 \quad q_3 >$ | halt για $< q_2 \quad 0 >$

$< q_3 \quad 0 \quad \mathrm{R} \quad q_4 >$

$< q_4 \quad 1 \quad \mathrm{R} \quad q_4 >$

$< q_4 \quad 0 \quad \mathrm{R} \quad q_5 >$

$< q_5 \quad 1 \quad \mathrm{R} \quad q_5 >$

$< q_5 \quad 0 \quad 1 \quad q_6 >$

$< q_6 \quad 1 \quad \mathrm{R} \quad q_6 >$

$< q_6 \quad 0 \quad 1 \quad q_7 >$

$< q_7 \quad 1 \quad \mathrm{L} \quad q_7 >$

$< q_7 \quad 0 \quad \mathrm{L} \quad q_8 >$

$< q_8 \quad 1 \quad \mathrm{L} \quad q_8 >$

$< q_8 \quad 0 \quad \mathrm{R} \quad q_2 >$

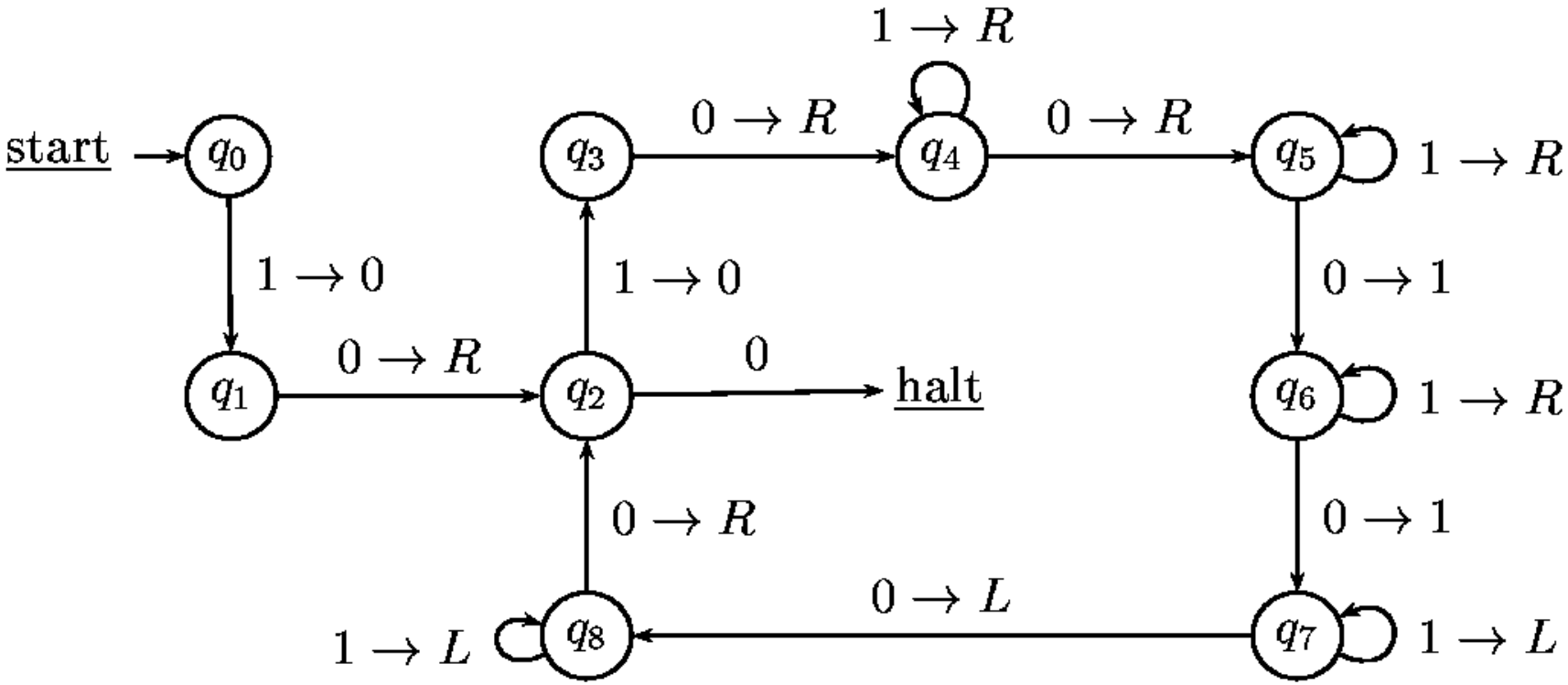
Υπολογισμός του 2x με ΤΜ (iii)

Σε μορφή πίνακα:

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
0		R/q_2	halt	R/q_4	R/q_5	$1/q_6$	$1/q_7$	L/q_8	R/q_2
1	$0/q_1$		$0/q_3$		R/q_4	R/q_5	R/q_6	L/q_7	L/q_8

Υπολογισμός του 2x με ΤΜ (iv)

Διάγραμμα Καταστάσεων:



Μηχανή Τυχαίας Προσπέλασης (Random Access Machine – RAM)

- Sheperdson & Sturgis, 1963, Elgott & Maclane (θεωρητική προσαρμογή)
- Στοιχεία μιας RAM:
  - Ταινίες εισόδου και εξόδου μιας κατεύθυνσης
  - Καταχωρητές
  - Συσσωρευτής (accumulator)
  - Σύνολο εντολών

Τυπικό σύνολο εντολών RAM

Κώδικας εντολής	Διεύθυνση
1. LOAD	operand
2. STORE	operand
3. ADD	operand
4. SUB	operand
5. MULT	operand
6. DIV	operand
7. READ	operand
8. WRITE	operand
9. JUMP	operand
10. JGTZ	label
11. JZERO	label
12. HALT	label

Τυπικό πρόγραμμα RAM:

Πρόγραμμα RAM		Αντίστοιχη ψευδογλώσσα
	READ 1	read r1
	LOAD 1	
	JGTZ pos	if $r1 <= 0$ then write 0
	WRITE =0	
	JUMP endelse	
pos:	LOAD 1	else begin
	STORE 2	
	LOAD 1	$r2 \leftarrow r1$
	SUB =1	
	STORE 3	$r3 \leftarrow r1 - 1$
while:	LOAD 3	
	JGTZ continue	
	JUMP endwhile	while $r3 > 0$ do begin
continue:	LOAD 2	
	MULT 1	
	STORE 2	
	LOAD 3	$r2 \leftarrow r1 * r1$
	SUB =1	
	STORE 3	$r3 \leftarrow r3 - 1$
	JUMP while	
endwhile:	WRITE 2	write r2
endelse:	HALT	