Αυτόματα και Υπολογιστικά Μοντέλα
Automata and Models of Computation

Διδάσκων: Στάθης Ζάχος

Επιμέλεια Διαφανειών: Μάκης Αρσένης
CoReLab

ΣΗΜΜΥ - Ε.Μ.Π.

Απρίλιος 2019

Περιεχόμενα

1 Πεπερασμένα Αυτόματα και Κανονικά Σύνολα

- Παραλλαγές, επεκτάσεις και εφαρμογές FA/REGEXP
- Ιδιότητες κανονικών συνόλων
- Αλγεβρική περιγραφή κανονικών συνόλων. Ελαχιστοποίηση DFA

2 Τυπικές Γλώσσες

- Τυπικές Γραμματικές
- Απλοποίηση c.f. γραμματικών
- Αυτόματα στοίβας (pushdown automata)
- Ιδιότητες c.f. γλωσσών

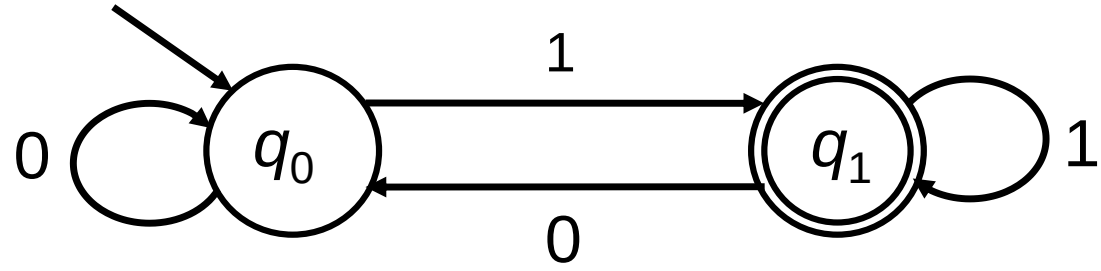
3 Μοντέλα Υπολογισμού

- Ιστορία - Εισαγωγή
- LOOP: Μια απλή γλώσσα προγραμματισμού
- LOOP-υπολογίσιμες και πρωταρχικές αναδρομικές συναρτήσεις

Αυτόματα και τυπικές γλώσσες

- Τυπικές γλώσσες: χρησιμοποιούνται για την κωδικοποίηση υπολογιστικών προβλημάτων αλλά και τον ορισμό γλωσσών προγραμματισμού.
  Π.χ. $L = \{x \in \{0,1\}^* \mid x$ δυαδική γραφή πρώτου αριθμού$\}$
- Αυτόματα: χρησιμεύουν για την αναγνώριση τυπικών γλωσσών και για την κατάταξη της δυσκολίας των αντίστοιχων προβλημάτων:
  - Κάθε αυτόματο αναγνωρίζει μια τυπική γλώσσα: *το σύνολο των συμβολοσειρών που το οδηγούν σε κατάσταση αποδοχής*.

Παράδειγμα: αναγνώριση περιττών



- q_0: τελευταίο ψηφίο διάφορο του 1
- q_1: τελευταίο ψηφίο ίσο με 1
- η q_0 λέγεται αρχική κατάσταση ενώ η q_1 λέγεται κατάσταση αποδοχής (ή και *τελική*)
- εκτέλεση με είσοδο 0110:
  $(q_0)0110 \rightarrow 0(q_0)110 \rightarrow 01(q_1)10 \rightarrow 011(q_1)0 \rightarrow 0110(q_0)$ ΑΠΟΡΡΙΨΗ
- εκτέλεση με είσοδο 101:
  $(q_0)101 \rightarrow 1(q_1)01 \rightarrow 10(q_0)1 \rightarrow 101(q_1)$ ΑΠΟΔΟΧΗ

Άλλα αυτόματα

- Μηχανισμοί: χωρίς είσοδο – έξοδο: $\delta(q_i) = q_j$
  εκτέλεση: q_0 -> q_j -> q_k -> q_m . . .
- Αυτόματα στοίβας (PDA, pushdown automata): έχουν πολύ περισσότερες δυνατότητες καθώς μπορούν να χρησιμοποιήσουν μνήμη (σε μορφή στοίβας).
- Μηχανές Turing (TM): έχουν ακόμη περισσότερες δυνατότητες καθώς έχουν απεριόριστη μνήμη (σε μορφή ταινίας, με δυνατότητα επιστροφής).
- Γραμμικά περιορισμένα αυτόματα (LBA): είναι TM με μνήμη γραμμικά περιορισμένη (ως προς το μήκος της εισόδου).

Άλλες τυπικές γλώσσες

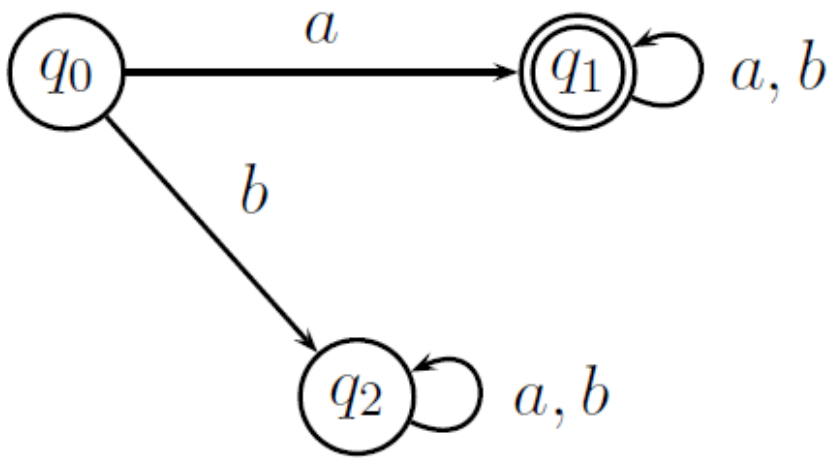
$L_1 := \{w \in \{a,b\}^* \mid w \text{ αρχίζει με } 'a'\}$

$L_2 := \{w \in \{1,3\}^* \mid w \text{ περιέχει άρτιο πλήθος } '1'\}$

$L_3 := \{w \in \{a,b\}^* \mid w \text{ είναι παλινδρομική }\}$

Παράδειγμα: DFA για L_1

$L_1 := \{w \in \{a,b\}^* \mid w \text{ αρχίζει με } 'a'\}$



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

εκτέλεση με είσοδο *abba* :

$(q_0)abba \rightarrow a(q_1)bba \rightarrow ab(q_1)ba \rightarrow abb(q_1)a \rightarrow abba(q_1)$

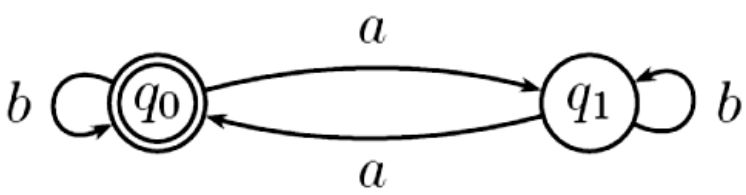
ΑΠΟΔΟΧΗ

Τυπικός ορισμός DFA

- Ντετερμινιστικό πεπερασμένο αυτόματο (Deterministic Finite Automaton, DFA): πεντάδα $M = (Q,\Sigma,\delta,q_0,F)$
  - Q : το σύνολο των καταστάσεων του M (πεπερασμένο), π.χ. $Q = \{q_0, q_1, q_2\}$
  - Σ : πεπερασμένο αλφάβητο εισόδου ($\Sigma \cap Q = \varnothing$), π.χ. $\Sigma = \{a,b\}$
  - $\delta : Q \times \Sigma \rightarrow Q$: συνάρτηση μετάβασης, π.χ. $\delta(q_0,a) = q_1$
  - $q_0 \in Q$: αρχική κατάσταση
  - $F \subseteq Q$: σύνολο τελικών καταστάσεων (αποδοχής), π.χ. $F = \{q_1\}$

Γλώσσα με DFA και γλώσσα χωρίς DFA

$$L_2 := \{w \in \{a,b\}^* \mid w \text{ περιέχει άρτιο πλήθος } 'a'\}$$



	a	b
(q_0)	q_1	q_0
q_1	q_0	q_1

$$L_3 := \{w \in \{a,b\}^* \mid w \text{ είναι παλινδρομική } \}$$

Αποδεικνύεται ότι δεν υπάρχει DFA που να αναγνωρίζει την L_3!
(χρειάζεται μνήμη με μέγεθος που εξαρτάται από την είσοδο)

Αποδοχή DFA: τυπικοί ορισμοί

- Επέκταση συνάρτησης δ: $Q \times \Sigma^* \to Q$
  Η επεκτεταμένη δ δέχεται ως ορίσματα μια κατάσταση q και μια συμβολοσειρά u και δίνει την κατάσταση όπου θα βρεθεί το αυτόματο αν ξεκινήσει από την q και διαβάσει την u.
- Ορισμός επεκτεταμένης δ (σχήμα *πρωταρχικής αναδρομής*):

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

όπου w είναι συμβολοσειρά οποιουδήποτε μήκους, ενώ a απλό σύμβολο του αλφαβήτου

Αποδοχή DFA: τυπικοί ορισμοί

- Ένα DFA αποδέχεται μία συμβολοσειρά u ανν $\delta(q_0,u) \in F$
- Ένα DFA M αποδέχεται τη γλώσσα

$$L(M) = \{w \mid \delta(q_0,w) \in F\}$$

- Οι γλώσσες που γίνονται αποδεκτές από DFA λέγονται κανονικές

Μη ντετερμινιστικά αυτόματα

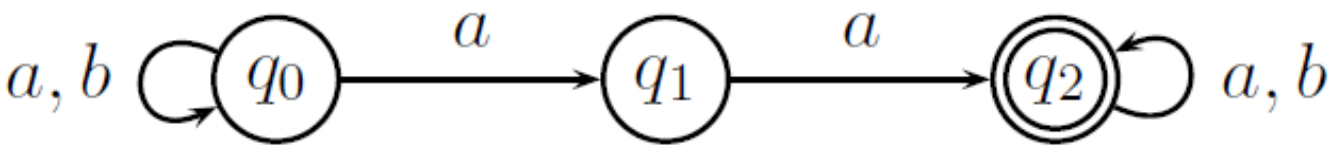
- Ντετερμινιστικά αυτόματα: για κάθε συνδυασμό κατάστασης / συμβόλου εισόδου υπάρχει μοναδική επόμενη κατάσταση
- Μη-ντετερμινιστικά αυτόματα:
  - για κάθε συνδυασμό κατάστασης / συμβόλου εισόδου υπάρχει *επιλογή* από σύνολο δυνατών επόμενων κατάστασεων
  - αποδοχή αν *κάποια* σειρά επιλογών οδηγεί σε αποδοχή

Μη ντετερμινιστικά πεπερασμένα αυτόματα

- NFA (Non-deterministic Finite Automaton): για κάθε κατάσταση και σύμβολο εισόδου επιλέγεται μία από ένα σύνολο δυνατών επόμενων καταστάσεων.
- NFAε (NFA με ε-κινήσεις): μπορεί να αλλάζει κατάσταση χωρίς ανάγνωση επόμενου συμβόλου.

Παράδειγμα NFA

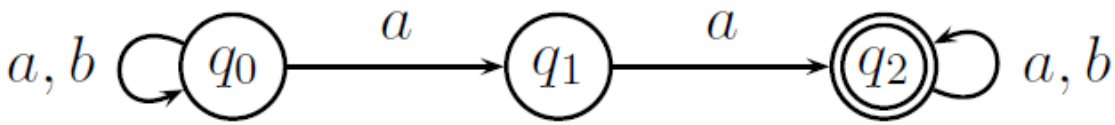
$$L_4 := \{w \in \Sigma^* \mid w \text{ περιέχει δύο συνεχόμενα } a\}$$



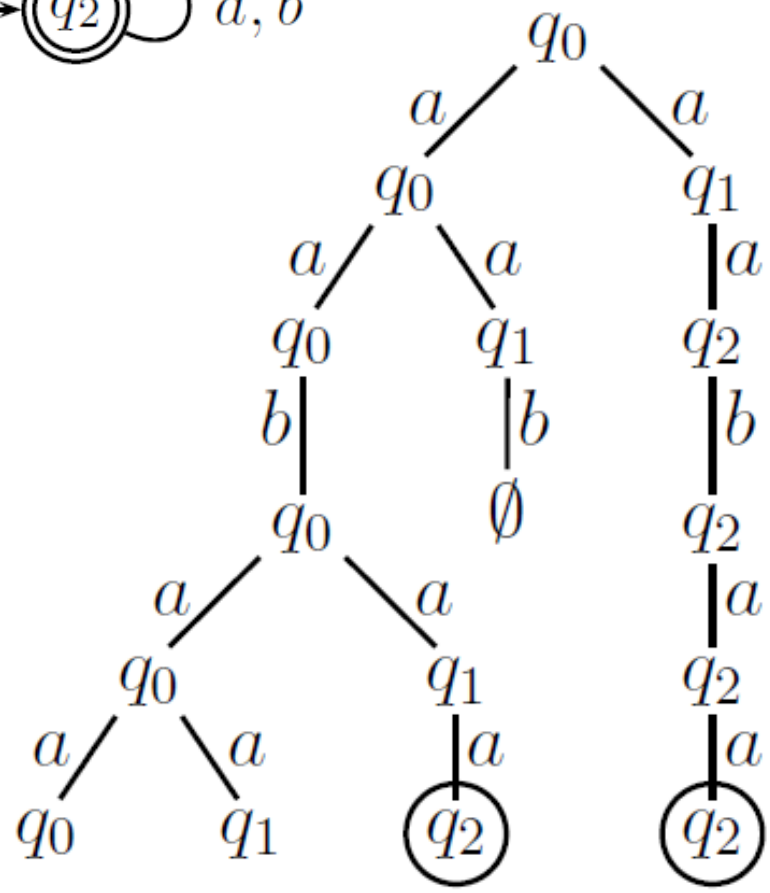
	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

Στη συνάρτηση μετάβασης, κενό σύνολο σημαίνει ότι η τρέχουσα εκτέλεση απορρίπτει (*προσοχή*: *μπορεί κάποια άλλη να αποδέχεται*).

Παράδειγμα NFA



Δένδρο υπολογισμού
για είσοδο *aabaa*



ΑΠΟΔΟΧΗ

Τυπικός ορισμός NFA

πεντάδα $M = (Q, \Sigma, \delta, q_0, F)$

- Q : το σύνολο των καταστάσεων του M (πεπερασμένο)
- Σ : πεπερασμένο αλφάβητο εισόδου ($\Sigma \cap Q = \varnothing$)
- $\delta : Q \times \Sigma \rightarrow \text{Pow}(Q)$: συνάρτηση μετάβασης,
  π.χ. $\delta(q_i, a) = \{ q_j, q_k, q_m \}$
- $q_0 \in Q$: αρχική κατάσταση
- $F \subseteq Q$: σύνολο τελικών καταστάσεων (αποδοχής)

Υπενθύμιση: στη συνάρτηση μετάβασης, κενό σύνολο σημαίνει ότι η συγκεκριμένη εκτέλεση απορρίπτει (*προσοχή*: *μπορεί κάποια άλλη να αποδέχεται*).

Αποδοχή NFA: τυπικοί ορισμοί

- Ένα NFA αποδέχεται συμβολοσειρά w αν $\delta(q_0,w) \cap F \neq \varnothing$
- Ένα NFA M αποδέχεται τη γλώσσα

$$L(M) = \{w \mid \delta(q_0,w) \cap F \neq \varnothing\}$$

- *Σημείωση*: η συνάρτηση δ είναι επεκτεταμένη ώστε να δέχεται σαν ορίσματα μια κατάσταση q και μια συμβολοσειρά w και να δίνει το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το αυτόματο αν ξεκινήσει από την q και διαβάσει την w
- *Παράδειγμα:* $\delta(q_0, aa) = \{q_0, q_1, q_2\}$, $\delta(q_0, ba) = \{q_0, q_1\}$

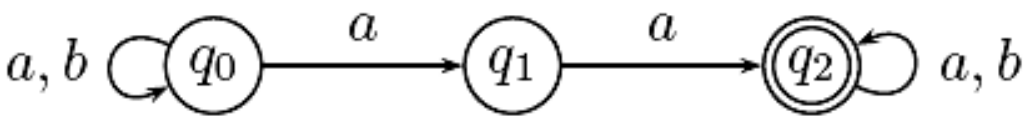
Ισοδυναμία NFA και DFA

- Θεώρημα Rabin-Scott: για κάθε NFA υπάρχει ένα DFA που αποδέχεται την ίδια γλώσσα.
- Επομένως τα DFA και τα NFA αναγνωρίζουν ακριβώς την ίδια κλάση γλωσσών: τις κανονικές γλώσσες (regular languages).
- Οι κανονικές γλώσσες αντιστοιχούν σε κανονικές παραστάσεις (regular expressions):

π.χ. $(a+b)^*bbab(a+b)^*$

NFA ➔ DFA (i)

NFA για τη γλώσσα L_4 ("2 συνεχόμενα a")



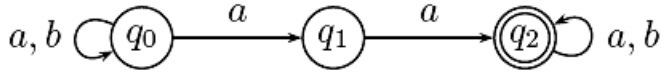
	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

Κατασκευάζουμε το *δυναμοσύνολο* των καταστάσεων. Αρχική κατάσταση: $\{q_0\}$.
Τελικές: όσες περιέχουν τελική.

Υπόδειξη: εξετάζουμε μόνο *προσβάσιμα* από $\{q_0\}$ σύνολα καταστάσεων.

NFA ➔ DFA (ii)

NFA για τη γλώσσα L_4



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

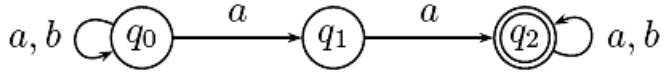
DFA για τη γλώσσα L_4

	$Q' \setminus \Sigma$	a	b
√	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$

Υπόδειξη: εξετάζουμε μόνο *προσβάσιμα* από $\{q_0\}$ σύνολα καταστάσεων.

NFA ➔ DFA (iii)

NFA για τη γλώσσα L_4



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

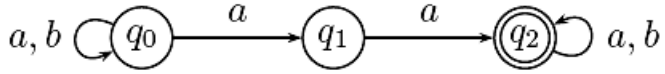
DFA για τη γλώσσα L_4

$Q' \setminus \Sigma$	a	b
√ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
√ $\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$

Υπόδειξη: εξετάζουμε μόνο *προσβάσιμα* από $\{q_0\}$ σύνολα καταστάσεων.

NFA ➔ DFA (iv)

NFA για τη γλώσσα L_4



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

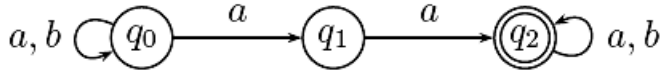
DFA για τη γλώσσα L_4

$Q' \setminus \Sigma$	a	b
√ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
√ $\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
√ $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Υπόδειξη: εξετάζουμε μόνο *προσβάσιμα* από $\{q_0\}$ σύνολα καταστάσεων.

NFA ➔ DFA (v)

NFA για τη γλώσσα L_4



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

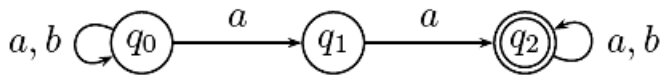
DFA για τη γλώσσα L_4

$Q' \setminus \Sigma$	a	b
√ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
√ $\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
√ $\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
√ $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Υπόδειξη: εξετάζουμε μόνο *προσβάσιμα* από $\{q_0\}$ σύνολα καταστάσεων.

NFA ➔ DFA (vi)

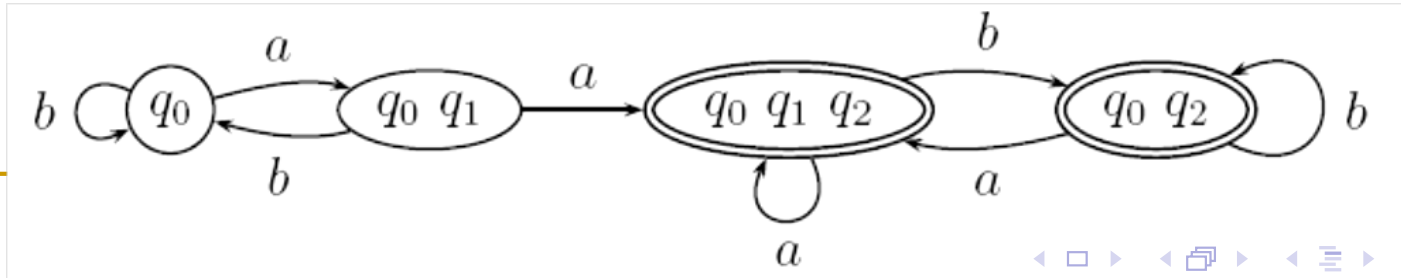
NFA για τη γλώσσα L_4



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

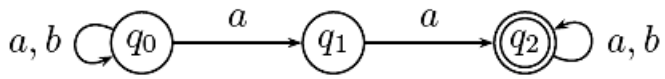
DFA για τη γλώσσα L_4

$Q' \setminus \Sigma$	a	b
√ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
√ $\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
√ $\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
√ $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$



NFA ➔ DFA (vii)

NFA για τη γλώσσα L_4

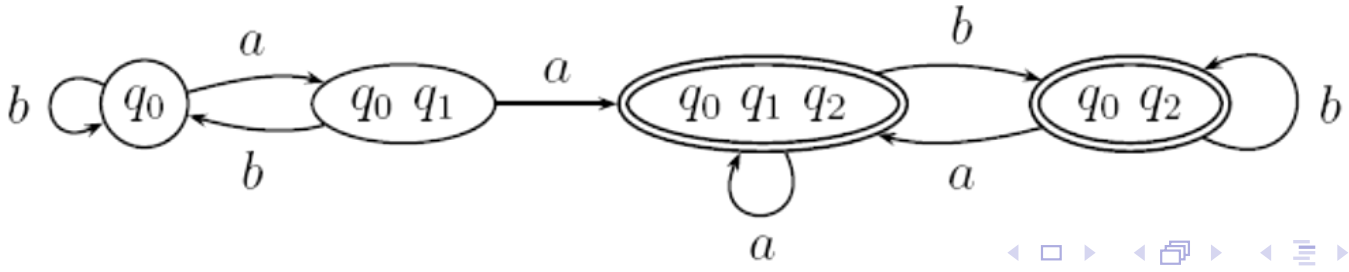


	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

DFA για τη γλώσσα L_4

	$Q' \setminus \Sigma$	a	b
	~~$\emptyset$~~	~~$\emptyset$~~	~~$\emptyset$~~
√	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	~~$\{q_1\}$~~	~~$\{q_2\}$~~	~~$\emptyset$~~
	~~$\{q_2\}$~~	~~$\{q_2\}$~~	~~$\{q_2\}$~~
√	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
√	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
	~~$\{q_1, q_2\}$~~	~~$\{q_2\}$~~	~~$\{q_2\}$~~
√	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Οι μη προσβάσιμες καταστάσεις δεν παίζουν ρόλο!



NFA ➔ DFA: η μέθοδος τυπικά

Έστω το NFA $M = (Q,\Sigma,q_0,F,\delta)$.

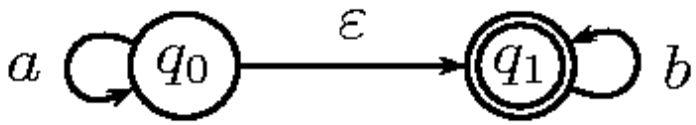
Ένα ισοδύναμο DFA $M'= (Q',\Sigma,q'_0,F',\delta')$, ορίζεται ως εξής:

- $Q' = \text{Pow}(Q)$, δηλαδή οι καταστάσεις του *M'* είναι όλα τα υποσύνολα καταστάσεων του *M*.
- $q'_0 = \{q_0\}$,
- $F' = \{R \in Q' \mid R \cap F \neq \varnothing\}$, δηλαδή μια κατάσταση του *M'* είναι τελική αν περιέχει μια τελική κατάσταση του *M*.
- $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ για $r \in R\}$, είναι δηλαδή το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το *M* ξεκινώντας από οποιαδήποτε κατάσταση του συνόλου *R* και διαβάζοντας το σύμβολο *a* (*a*-κίνηση).

Αυτόματα με ε-κινήσεις: NFA_ε

- Επιτρέπουν μεταβάσεις χωρίς να διαβάζεται σύμβολο (ισοδύναμα: με είσοδο το κενό string ε).
- Αποδέχονται τις συμβολοσειρές που μπορούν να οδηγήσουν σε τελική κατάσταση, χρησιμοποιώντας ενδεχομένως και ε-κινήσεις.

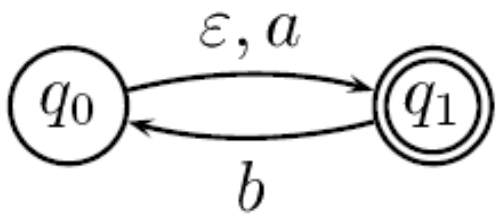
Παράδειγμα: NFA_ε για $L_5 := \{a^*b^*\} = \{a^n b^m \mid n, m \in \mathbb{N}\}$



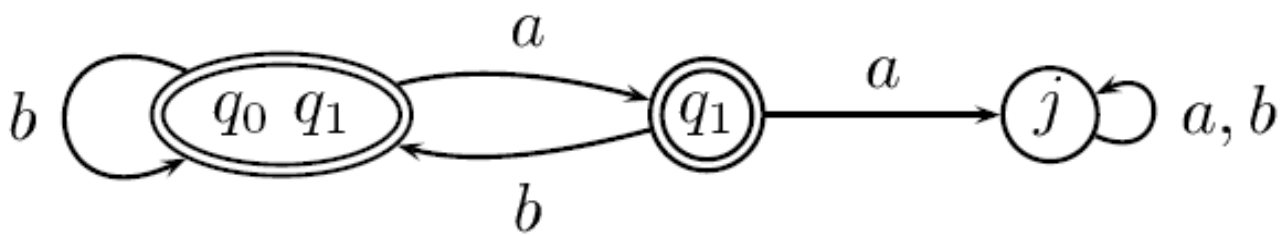
	a	b	ε
q_0	$\{q_0\}$	$\emptyset$	$\{q_0, q_1\}$
(q_1)	$\emptyset$	$\{q_1\}$	$\{q_1\}$

Ισοδυναμία NFA_ε με DFA: παράδειγμα

NFA_ε για $\overline{L_4}$ (δηλαδή "όχι δύο συνεχόμενα a"):



DFA για $\overline{L_4}$:



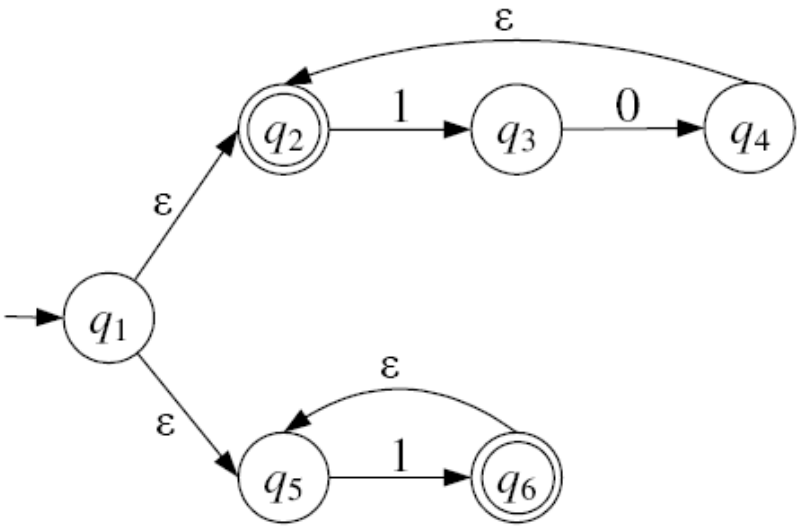
NFA_ε ➔ DFA: η μέθοδος τυπικά

Έστω το NFA_ε $M = (Q,\Sigma,q_0,F,\delta)$.

Ένα ισοδύναμο DFA $M'= (Q',\Sigma,q'_0,F',\delta')$, ορίζεται ως εξής:

- $Q' = Pow(Q)$, δηλαδή οι καταστάσεις του M' είναι όλα τα υποσύνολα καταστάσεων του M.
- $q'_0 =$ ε-κλείσιμο(q_0) (ε-κλείσιμο(q_i)= {p | p προσβάσιμη κατάσταση από την q_i μόνο με ε-κινήσεις})
- $F' = \{R \in Q' \mid R \cap F \neq \varnothing\}$, δηλαδή μια κατάσταση του M' είναι τελική αν περιέχει μια τελική κατάσταση του M.
- $\delta'(R,a) = \{q \in Q \mid q \in$ ε-κλείσιμο$(\delta(r, a))$ για $r \in R\}$, δηλαδή $\delta'(R, a)$ είναι το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το M ξεκινώντας από οποιαδήποτε κατάσταση του R , κάνοντας μία a-κίνηση και χρησιμοποιώντας στη συνέχεια οσεσδήποτε ε-κινήσεις.

NFA_ε ➔ DFA: παράδειγμα



ε-κλείσιμο της q_1={q_1,q_2,q_5}
ε-κλείσιμο της q_2={q_2}
ε-κλείσιμο της q_3={q_4}
ε-κλείσιμο της q_4={q_2,q_4}
ε-κλείσιμο της q_5={q_5}
ε-κλείσιμο της q_6={q_5,q_6}

Αρχική κατάσταση
{q_1,q_2,q_5}
Τελικές οι μπλέ

Q'	0	1
>{q_1,q_2,q_5}	{ }	{q_3,q_5,q_6}
{q_3,q_5,q_6}	{q_2,q_4}	{q_5,q_6}
{q_2,q_4}	{ }	{q_3}
{q_5,q_6}	{ }	{q_5,q_6}
{q_3}	{q_2,q_4}	{ }
{ }	{ }	{ }

NFA_ε ➔ DFA: εναλλακτική μέθοδος (σε NFA πρώτα)

- *Έστω το NFA_ε $M = (Q,\Sigma,q0,F,\delta)$.*
- Κατασκευάζουμε πρώτα ισοδύναμο NFA $M' = (Q,\Sigma,q_0,F',\delta')$ ως εξής:
  - $F' = F \cup \{q_0\}$ αν ε-κλείσιμο(q_0) περιέχει τελική,
    $F' = F$ αλλιώς.
  - $\delta'(q, a)$ = ε-κλείσιμο(δ(ε-κλείσιμο(q), a)), (δ επεκτ/νη)

  δηλαδή $\delta'(q, a)$ είναι το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το *M* ξεκινώντας από την κατάσταση *q*, χρησιμοποιώντας οσεσδήποτε ε-κινήσεις, μία *a*-κίνηση, και χρησιμοποιώντας ξανά οσεσδήποτε ε-κινήσεις.
- Από το *M'* κατασκευάζουμε ισοδύναμο DFA.

NFA_ε ➔ DFA: εναλλακτική μέθοδος (σε NFA πρώτα, ταχύτερα)

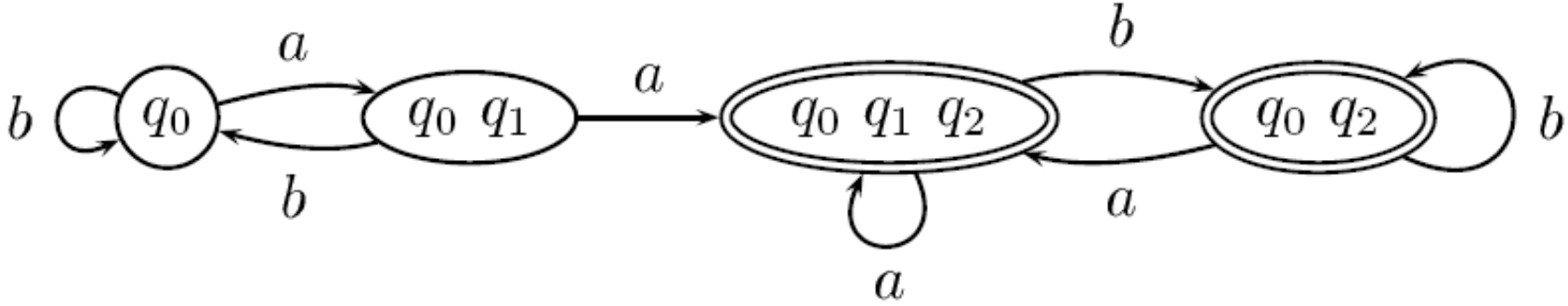
- *Έστω το* NFA_ε $M = (Q,\Sigma,q0,F,\delta)$.
- Κατασκευάζουμε πρώτα ισοδύναμο NFA $M'= (Q,\Sigma,q_0,F',\delta')$ ως εξής:
  - F' : είναι το F μαζί με *κάθε* κατάσταση q για την οποία το ε-κλείσιμο(q) περιέχει κάποια τελική.
  - $\delta'(q, a) = \delta$(ε-κλείσιμο(q), a), (δ επεκτ/νη)

  δηλαδή $\delta'(q, a)$ είναι το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το M ξεκινώντας από την κατάσταση q, χρησιμοποιώντας οσεσδήποτε ε-κινήσεις πρώτα, και μετά μία a-κίνηση *(δηλ. χωρίς ε-κινήσεις μετά το a)*.
- Από το M' κατασκευάζουμε ισοδύναμο DFA.

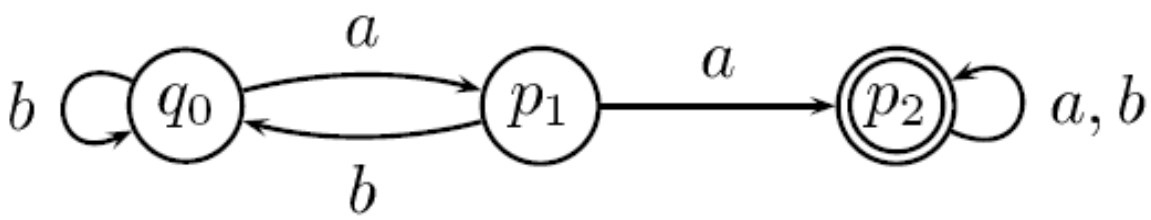
Ελαχιστοποίηση DFA: παράδειγμα

$L_4 = \{ w \in \{a,b\}^* \mid w$ περιέχει 2 συνεχόμενα $a \}$:

Αρχικό DFA

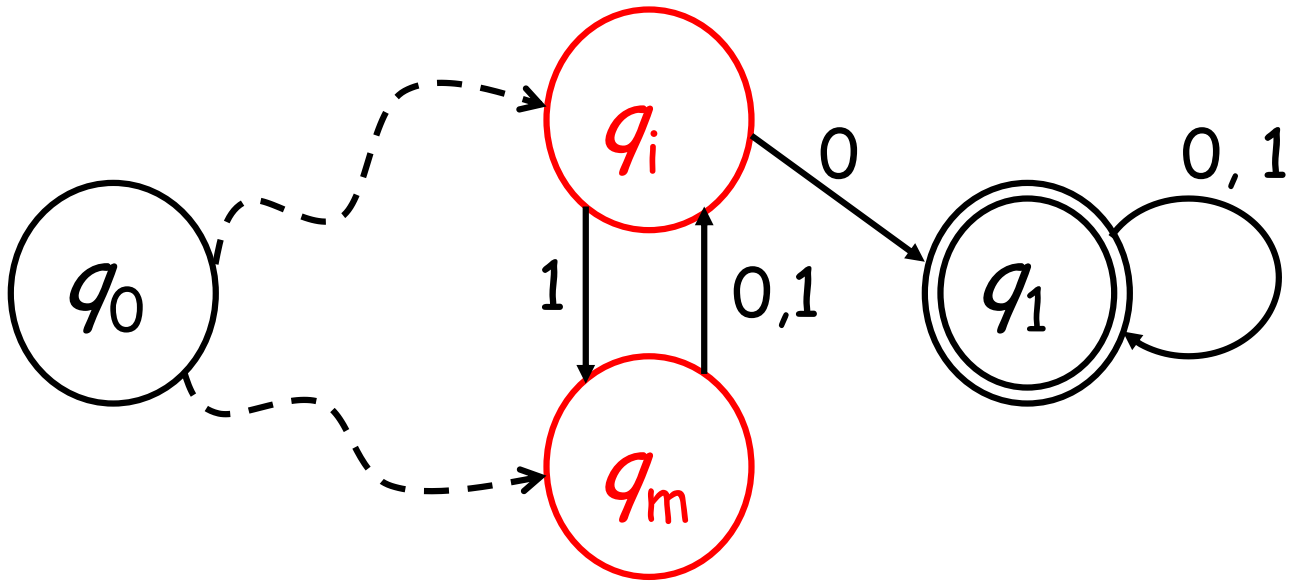


Ελάχιστο DFA



Ελαχιστοποίηση DFA (i)

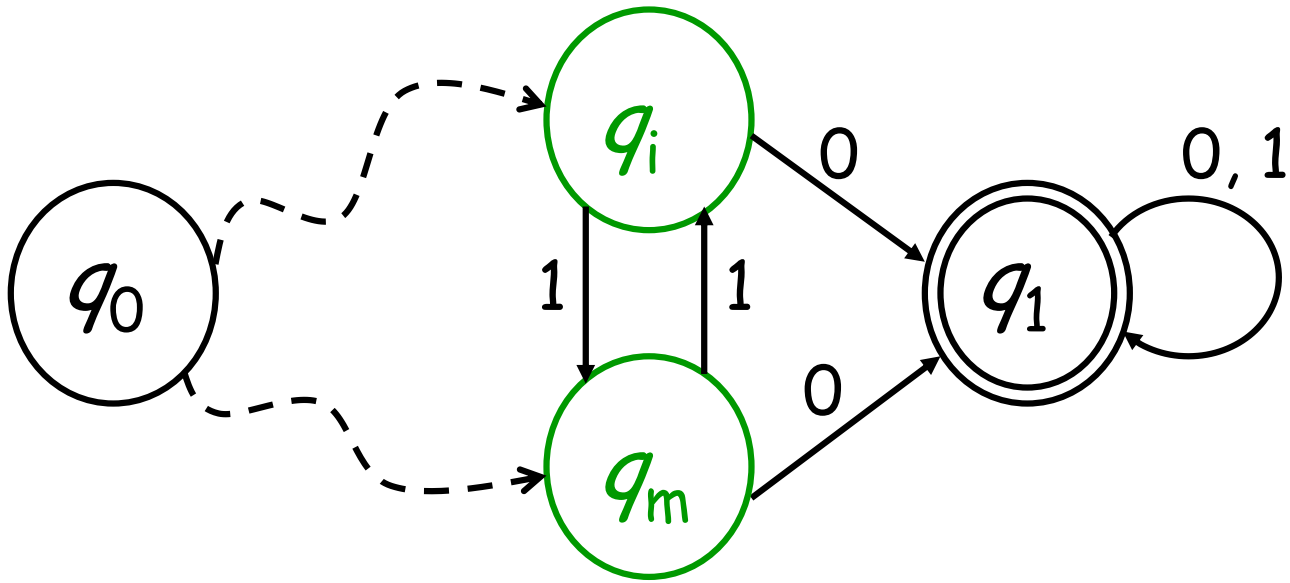
Δύο καταστάσεις DFA λέγονται *μη ισοδύναμες*, δηλαδή *διακρίσιμες*, αν υπάρχει συμβολοσειρά που να οδηγεί την μία από αυτές σε τελική κατάσταση, ενώ την άλλη όχι.



Ελαχιστοποίηση DFA (ii)

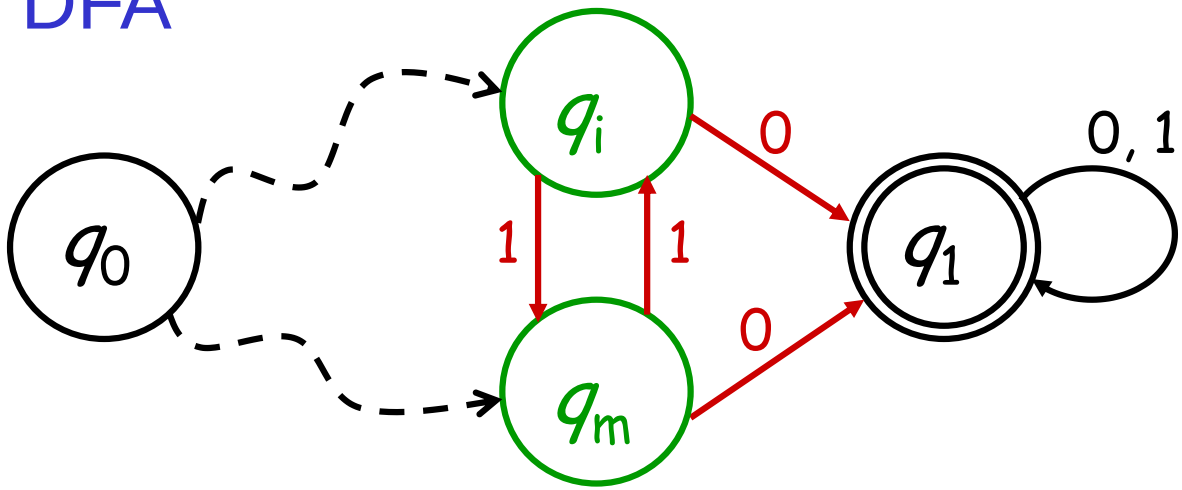
Δύο καταστάσεις μπορούν να συγχωνευτούν σε μία (είναι *ισοδύναμες*) αν:

οδηγούν με ίδιες συμβολοσειρές σε ίδιο αποτέλεσμα

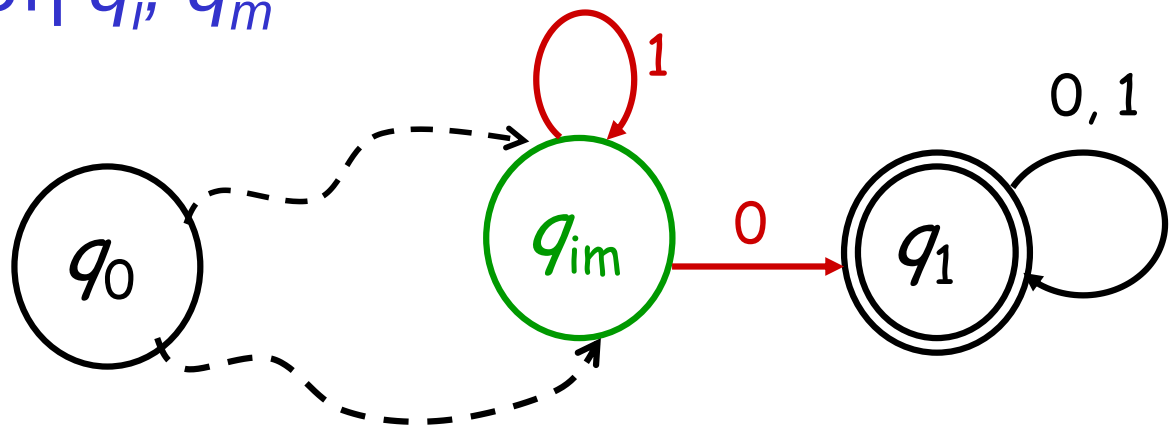


Ελαχιστοποίηση DFA (iii)

Αρχικό DFA

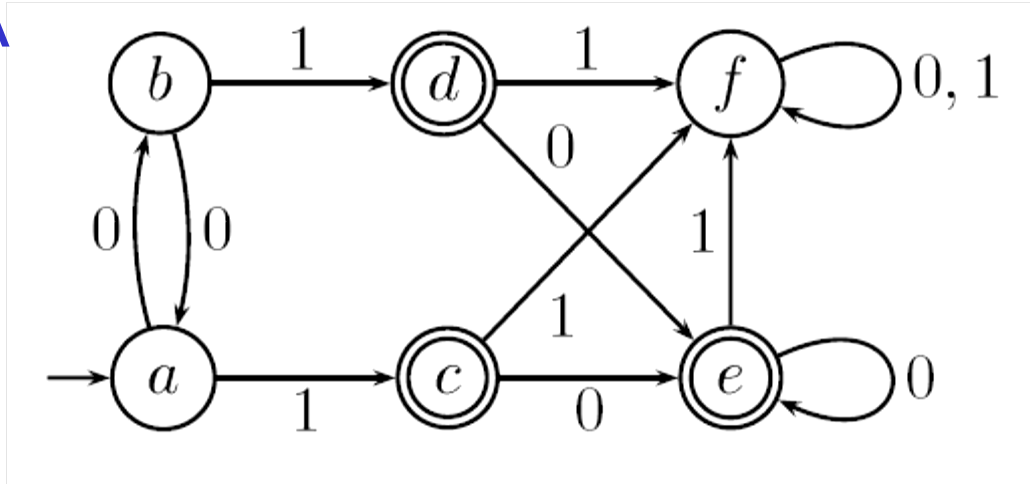


Συγχώνευση q_i, q_m

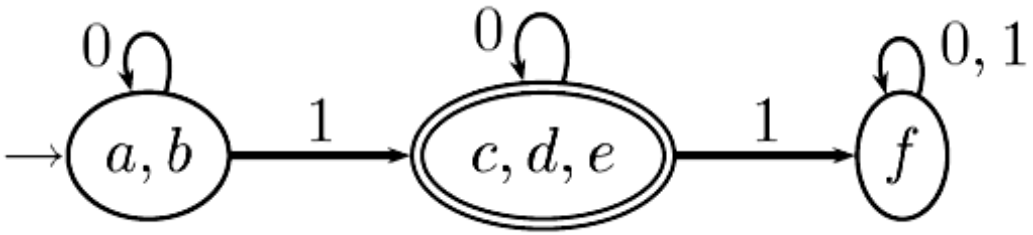


Ελαχιστοποίηση DFA: 2ο παράδειγμα

Αρχικό DFA

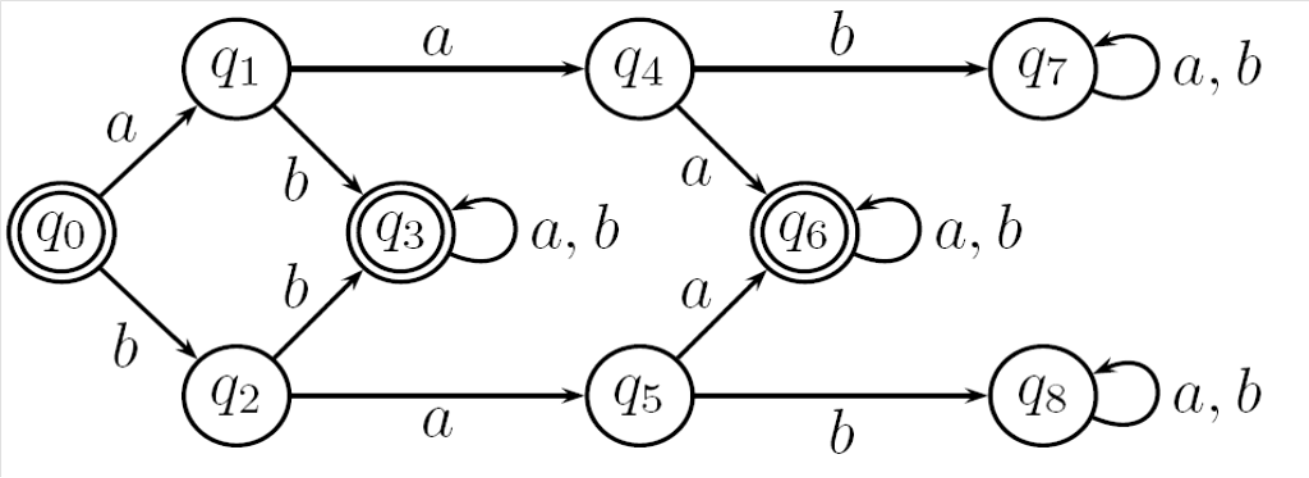


Ελάχιστο DFA

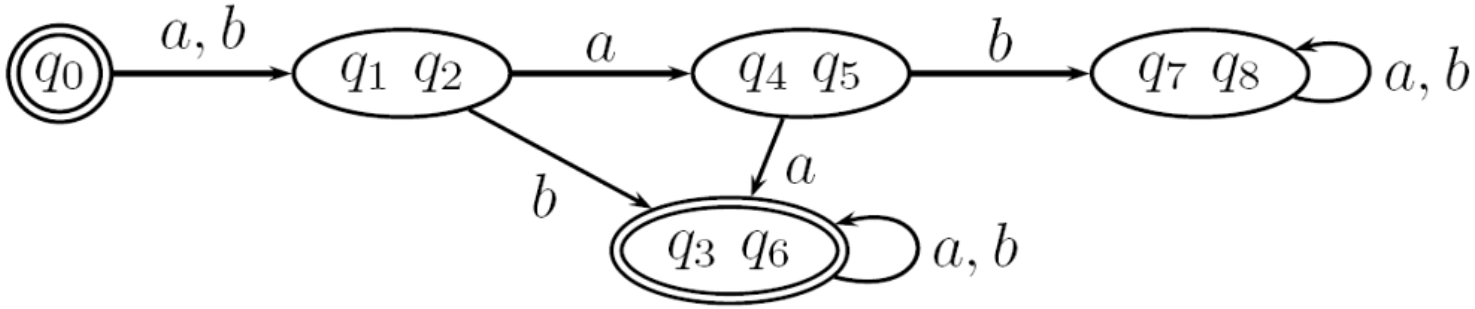


Ελαχιστοποίηση DFA: 3ο παράδειγμα

Αρχικό DFA



Ελάχιστο DFA



Μέθοδος ελαχιστοποίησης DFA

- Δύο καταστάσεις λέγονται ***k*-διακρίσιμες** αν με *κάποια συμβολοσειρά* μήκους ακριβώς *k* οδηγούν σε διαφορετικό αποτέλεσμα (και δεν είναι *i*-διακρίσιμες για κανένα *i*<*k*). Έτσι, δύο καταστάσεις είναι:
  - 0-διακρίσιμες ανν η μία είναι τελική ενώ η άλλη όχι.
  - (*i*+1)-διακρίσιμες ανν με *κάποιο σύμβολο* οδηγούν σε *i*-διακρίσιμες καταστάσεις.
- Δύο καταστάσεις λέγονται **ισοδύναμες** αν δεν είναι *k*-διακρίσιμες για *οποιοδήποτε k*.

Μέθοδος ελαχιστοποίησης DFA

- Ιδέα μεθόδου: για κάθε $i = 0, 1, 2, \ldots$ εντοπίζουμε τα *i-διακρίσιμα* ζεύγη καταστάσεων έως ότου να μην προκύπτουν άλλα. Τα υπόλοιπα ζεύγη είναι ισοδύναμα.
- Γιατί δουλεύει:

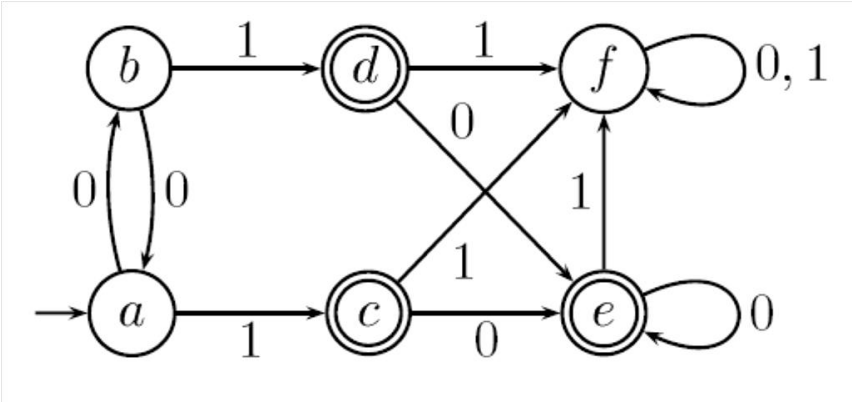
  δεν υπάρχουν (i+1)-διακρίσιμες καταστάσεις αν δεν υπάρχουν i-διακρίσιμες καταστάσεις

Η μέθοδος συστηματικά

Κατασκευάζουμε τριγωνικό πίνακα για να συγκρίνουμε κάθε ζεύγος καταστάσεων. Γράφουμε X_k στην αντίστοιχη θέση του πίνακα την πρώτη φορά που διαπιστώνουμε ότι δύο καταστάσεις είναι *k*-διακρίσιμες, ως εξής:

- Αρχικά γράφουμε X_0 σε όλα τα ζεύγη κατ/σεων που είναι 0-διακρίσιμες γιατί η μία είναι τελική και η άλλη όχι.
- Σε κάθε "γύρο" *i+1*, εξετάζουμε όλα τα μη σημειωμένα ζεύγη και γράφουμε X_{i+1} σε ένα ζεύγος αν από τις δύο καταστάσεις του με *ένα σύμβολο* το DFA πηγαίνει σε *i* -διακρίσιμες καταστάσεις (ήδη σημ/νες με X_i).
- Επαναλαμβάνουμε μέχρι που σε κάποιο γύρο *k* να μην υπάρχει ζεύγος που να σημειωθεί με X_k .
- Τα μη σημειωμένα ζεύγη αντιστοιχούν σε ισοδύναμες καταστάσεις (που επομένως συγχωνεύονται).

Παράδειγμα εφαρμογής της μεθόδου

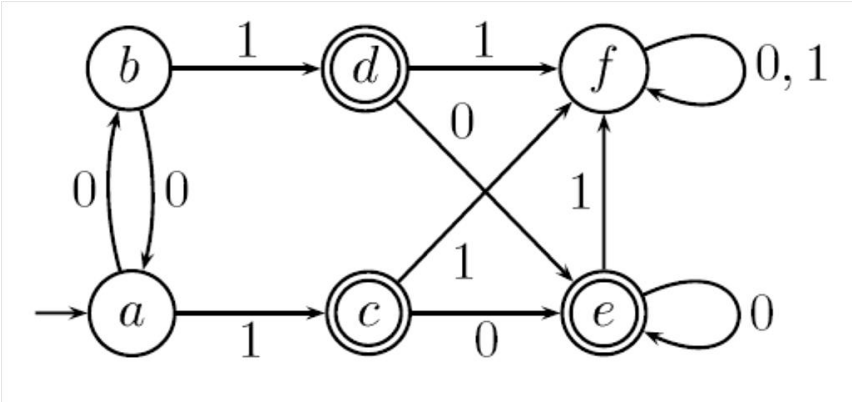


b				
c	X_0	X_0		
d	X_0	X_0		
e	X_0	X_0		
f			X_0	X_0	X_0
	a	b	c	d	e

Γύρος 0:

εννέα ζεύγη
0-διακρίσιμων
καταστάσεων

Παράδειγμα εφαρμογής της μεθόδου

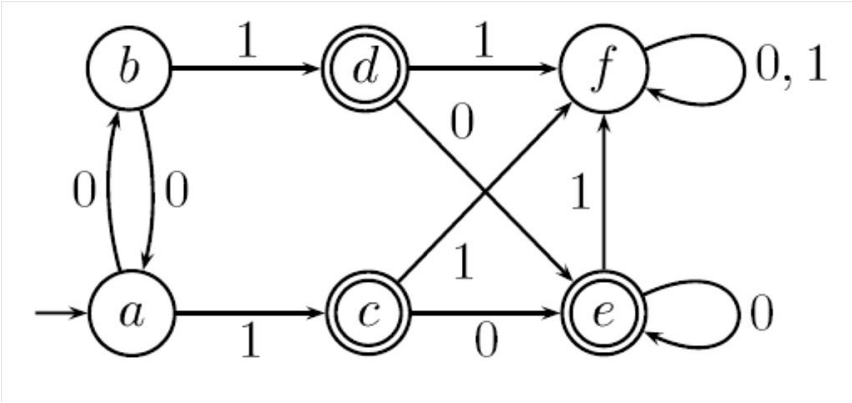


b					
ⓒ	X_0	X_0			
ⓓ	X_0	X_0			
ⓔ	X_0	X_0			
f	X_1	X_1	X_0	X_0	X_0
	a	b	ⓒ	ⓓ	ⓔ

Γύρος 1:

δύο ζεύγη
1-διακρίσιμων
καταστάσεων

Παράδειγμα εφαρμογής της μεθόδου

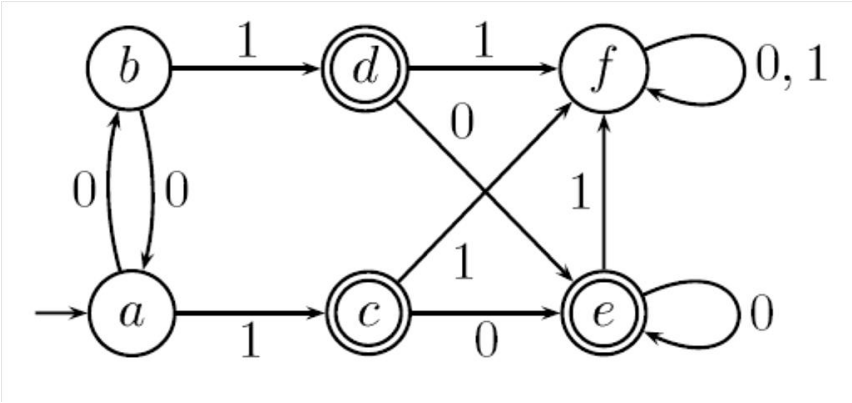


b					
ⓒ	X_0	X_0			
ⓓ	X_0	X_0			
ⓔ	X_0	X_0			
f	X_1	X_1	X_0	X_0	X_0
	a	b	ⓒ	ⓓ	ⓔ

Γύρος 2:

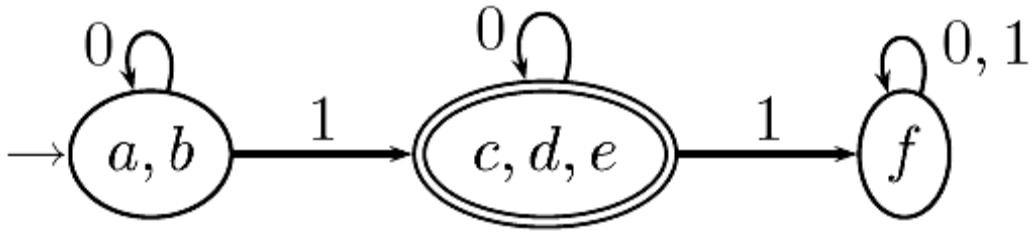
κανένα ζεύγος
2-διακρίσιμων
καταστάσεων

Παράδειγμα εφαρμογής της μεθόδου



b					
ⓒ	X_0	X_0			
ⓓ	X_0	X_0			
ⓔ	X_0	X_0			
f	X_1	X_1	X_0	X_0	X_0
	a	b	ⓒ	ⓓ	ⓔ

Τελικά οι ισοδύναμες καταστάσεις είναι $a \equiv b$, $c \equiv d \equiv e$.
Το ελάχιστο αυτόματο φαίνεται στο παρακάτω σχήμα.



Γλώσσες, αυτόματα, γραμματικές

- Τυπικές γλώσσες: χρησιμοποιούνται για την περιγραφή υπολογιστικών προβλημάτων αλλά και γλωσσών προγραμματισμού.
- Αυτόματα: χρησιμεύουν για την αναγνώριση τυπικών γλωσσών και για την κατάταξη της δυσκολίας των αντίστοιχων προβλημάτων.
- Τυπικές γραμματικές: άλλος τρόπος περιγραφής τυπικών γλωσσών. Κάθε τυπική γραμματική *παράγει* μια τυπική γλώσσα.

Θεωρία γλωσσών και γραμματικών

Εφαρμογές σε:

- Ψηφιακή Σχεδίαση,
- Γλώσσες Προγραμματισμού,
- Μεταγλωττιστές,
- Τεχνητή Νοημοσύνη,
- Θεωρία Πολυπλοκότητας

Ιστορικά σημαντικοί ερευνητές:

- Chomsky, Backus, Rabin, Scott, Kleene, Greibach, κ.α.

Τυπικές γλώσσες

- Πρωταρχικές έννοιες: σύμβολα, παράθεση.
- Αλφάβητο: πεπερασμένο σύνολο συμβόλων. Π.χ. {0,1}, {x,y,z}, {a,b}.
- Λέξη (ή συμβολοσειρά, ή πρόταση) ενός αλφαβήτου: πεπερασμένου μήκους ακολουθία συμβόλων του αλφαβήτου. Π.χ. 011001, abbbab.
- $|w|$: μήκος λέξης w.
- ε : κενή λέξη, $|\varepsilon| = 0$.
- Άλλες έννοιες: πρόθεμα (prefix), κατάληξη (suffix), υποσυμβολοσειρά (substring), αντίστροφη (reversal), παλινδρομική ή καρκινική (palindrome).

Τυπικές γλώσσες (συν.)

- vw = παράθεση λέξεων v και w.
- Ισχύει: $\varepsilon x = x\varepsilon = x$, για κάθε συμβολοσειρά x.
- ορισμός x^n με πρωταρχική αναδρομή:

$$\begin{cases} x^0 = \varepsilon \\ x^{k+1} = x^k x \end{cases}$$

- Σ^*: το σύνολο όλων των λέξεων του αλφαβήτου Σ.
- Γλώσσα από το αλφάβητο Σ: κάθε σύνολο συμβολοσειρών $L \subseteq \Sigma^*$.

Τυπικές γραμματικές

- Συστηματικός τρόπος μετασχηματισμού συμβολοσειρών μέσω κανόνων παραγωγής.
- Αλφάβητο: τερματικά και μη τερματικά σύμβολα και ένα αρχικό σύμβολο (μη τερματικό).
- Πεπερασμένο σύνολο κανόνων της μορφής $\alpha \rightarrow \beta$: ορίζουν δυνατότητα αντικατάστασης της συμβολοσειράς α με την συμβολοσειρά β.
- Κάθε τυπική γραμματική παράγει μια τυπική γλώσσα: το σύνολο των συμβολοσειρών (με *τερματικά σύμβολα* μόνο) που παράγονται από το αρχικό σύμβολο.
- Λέγονται και συστήματα μεταγραφής (rewriting systems) αλλά και γραμματικές δομής φράσεων (phrase structure grammars).

Παράδειγμα γραμματικής για την γλώσσα των περιττών αριθμών

$S \rightarrow A\ 1$

$A \rightarrow A\ 0$

$A \rightarrow A\ 1$

$A \rightarrow \varepsilon$

S: το αρχικό σύμβολο

A: μη τερματικό σύμβολο

0,1: τερματικά σύμβολα

ε: η κενή συμβολοσειρά

- Τα S και A αντικαθίστανται με βάση τους κανόνες.
- Κάθε περιττός προκύπτει από το S με κάποια σειρά έγκυρων αντικαταστάσεων.
- Κανονική παράσταση: (0+1)*1

Τυπικές γραμματικές: ορισμοί (i)

Μια **τυπική γραμματική** G αποτελείται από:

- ένα αλφάβητο V από *μη τερματικά* σύμβολα (μεταβλητές),
- ένα αλφάβητο T από *τερματικά* σύμβολα (σταθερές), τ.ώ. $V \cap T = \varnothing$,
- 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 $\alpha \rightarrow \beta$ αντί για (α,β)),
- ένα *αρχικό σύμβολο* (ή αξίωμα) $S \in V$.

Τυπικές γραμματικές: ορισμοί (ii)

Σύμβαση για τη χρήση γραμμάτων:

- $a, b, c, d, \ldots \in T$: πεζά λατινικά, τα αρχικά του αλφαβήτου, συμβολίζουν τερματικά
- $A, B, C, D, \ldots \in V$: κεφαλαία λατινικά συμβολίζουν μη τερματικά
- $u, v, w, x, y, z \ldots \in T^*$: πεζά λατινικά, τα τελευταία του αλφαβήτου, συμβολίζουν συμβολοσειρές τερματικών
- $\alpha, \beta, \gamma, \delta, \ldots \in (V \cup T)^*$: ελληνικά συμβολίζουν οποιεσδήποτε συμβολοσειρές (τερματικών και μη)

Τυπικές γραμματικές: ορισμοί (iii)

Ορισμοί για τις παραγωγές:

- Λέμε ότι $\gamma_1\alpha\gamma_2$ *παράγει* $\gamma_1\beta\gamma_2$,και συμβολίζουμε με $\gamma_1\alpha\gamma_2 \Rightarrow \gamma_1\beta\gamma_2$, αν ο $\alpha \to \beta$ είναι κανόνας παραγωγής (δηλαδή $(\alpha,\beta) \in P$).
- Συμβολίζουμε με $\overset{*}{\Rightarrow}$ το ανακλαστικό, μεταβατικό κλείσιμο του $\Rightarrow$, δηλαδή, $\alpha \overset{*}{\Rightarrow} \beta$ («το α παράγει το β») σημαίνει ότι υπάρχει μια ακολουθία:
  $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \alpha_k \Rightarrow \beta$.
- *Γλώσσα που παράγεται* από τη γραμματική G:
  $$L(G) := \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$$
- γραμματικές G_1, G_2 *ισοδύναμες* αν $L(G_1) = L(G_2)$.

Παράδειγμα τυπικής γραμματικής

$$G\colon\ V = \{S\},\ T = \{a, b\},\ P = \{S \to \varepsilon | aSb\}$$

$S \to \varepsilon \mid aSb$: σύντμηση των $S \to \varepsilon$ και $S \to aSb$

Μία δυνατή ακολουθία παραγωγής:

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

Γλώσσα που παράγεται:

$$L(G) = \{a^n b^n \mid n \in N\}$$

Ιεραρχία Γραμματικών Chomsky

- τύπου 0: γενικές γραμματικές (general, phrase structure, semi-Thue).

$$\alpha \to \beta,\ \alpha \neq \varepsilon$$

- τύπου 1: γραμματικές με συμφραζόμενα ή μονοτονικές (context sensitive, monotonic).

$$\alpha \to \beta, \quad |\alpha| \leq |\beta| \text{ (επιτρέπεται και: } S \to \varepsilon)$$

- τύπου 2: γραμματικές χωρίς συμφραζόμενα (context free).

$$A \to \alpha \quad (A \in V)$$

- τύπου 3: κανονικές γραμματικές (regular).

δεξιογραμμικές: $A \to w,\ A \to wB \quad (w \in T^*,\ A, B \in V)$ ή

αριστερογραμμικές: $A \to w,\ A \to Bw \quad (w \in T^*,\ A, B \in V)$

Γνήσια ιεράρχηση: τύπου 3 $\subset$ τύπου 2 $\subset$ τύπου 1 $\subset$ τύπου 0

Ιεραρχία Chomsky: μια εκπληκτική σύμπτωση (;)

- τύπου 0 ↔ TM (μηχανές Turing)
- τύπου 1 ↔ LBA (γραμμικά περιορισμένα αυτόματα)
- τύπου 2 ↔ PDA (pushdown automata)
- τύπου 3 ↔ DFA (και NFA)

Κανονικές Γραμματικές

- Οι **κανονικές γραμματικές** είναι γραμματικές όπου όλοι οι κανόνες είναι της μορφής:
  - Δεξιογραμμικοί (right linear)

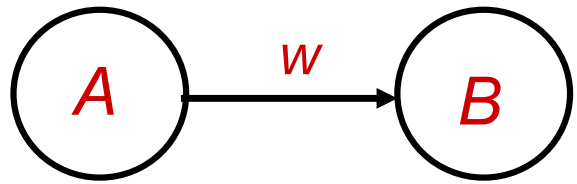
    $A \rightarrow wB$ ή $A \rightarrow w$
  - Αριστερογραμμικοί (left linear)

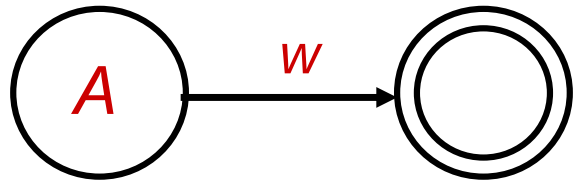
    $A \rightarrow Bw$ ή $A \rightarrow w$

  (όπου *w* είναι μια ακολουθία από τερματικά σύμβολα της γλώσσας)

- **Θεώρημα**: *οι κανονικές γλώσσες ταυτίζονται με τις γλώσσες που παράγονται από κανονικές γραμματικές.*

Ισοδυναμία κανονικών γραμματικών και DFA

- Χρησιμοποιούμε τη δεξιογραμμική μορφή:
  - $A \rightarrow wB$ αντιστοιχεί με 

  - $A \rightarrow w$ αντιστοιχεί με 

  - S αντιστοιχεί με q_0

Μία ακόμη ισοδυναμία!

Θεώρημα: *οι* κανονικές γλώσσες *ταυτίζονται με τις γλώσσες που περιγράφονται από* κανονικές παραστάσεις.

Κανονικές παραστάσεις (regular expressions)

Έστω L, L_1, L_2 γλώσσες επί του ίδιου 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
- $L^+ := \bigcup_{n=1}^{\infty} L^n$

Ορισμός κανονικών παραστάσεων

Κανονικές παραστάσεις: παριστάνουν γλώσσες που προκύπτουν από απλά σύμβολα ενός αλφαβήτου με τις πράξεις παράθεση, ένωση, και άστρο του Kleene.

- $\varnothing$: παριστάνει κενή γλώσσα
- ε : παριστάνει $\{\varepsilon\}$
- α : παριστάνει $\{\alpha\}$, $\alpha \in \Sigma$
- $(r+s)$: παριστάνει $R \cup S$, $R = L(r)$, $S = L(s)$
- (rs) : παριστάνει RS, $R = L(r)$, $S = L(s)$
- (r^*) : παριστάνει R^*, $R = L(r)$

όπου $L(t)$ η γλώσσα που παριστάνεται από καν. παρ. t

Παραδείγματα κανονικών παραστάσεων

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

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

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)

$L_5 = a^*b^*$

Προτεραιότητα τελεστών:

- άστρο Kleene
- παράθεση
- ένωση

Ισοδυναμία κανονικών παραστάσεων και αυτομάτων

Θεώρημα. Μια γλώσσα L μπορεί να παρασταθεί με *κανονική παράσταση* ανν είναι *κανονική* (δηλαδή *L=L(M)* για κάποιο πεπερασμένο αυτόματο *M*).

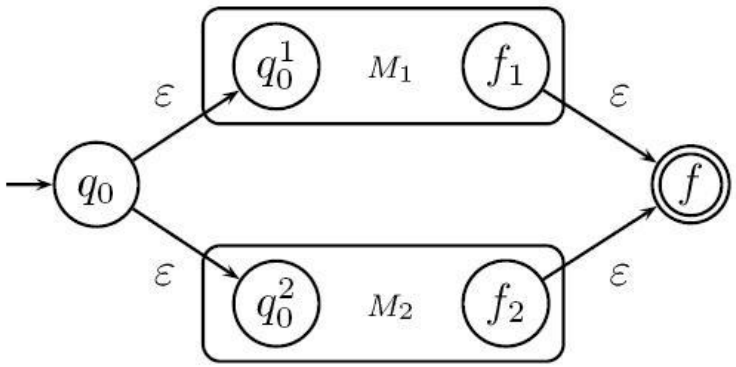
Ιδέα απόδειξης:

'=>' : Επαγωγή στη δομή της κανονικής παράστασης *r* :

1. Επαγωγική Βάση:

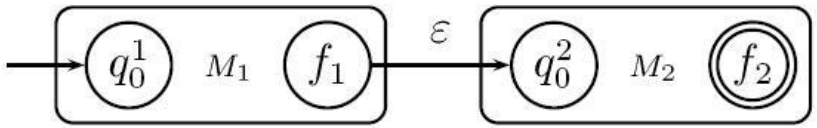
2. Επαγωγικό βήμα. Έστω ότι για r_1, r_2 έχουμε αυτόματα M_1, M_2, με τελικές καταστάσεις f_1, f_2:

Περίπτωση α: $r = r_1 + r_2$



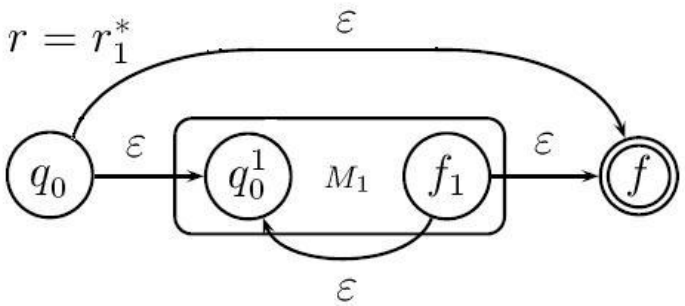
$$L(M) = L(M_1) \cup L(M_2)$$

Περίπτωση β: $r = r_1 r_2$



$$L(M) = L(M_1)L(M_2)$$

Περίπτωση γ: $r = r_1^*$

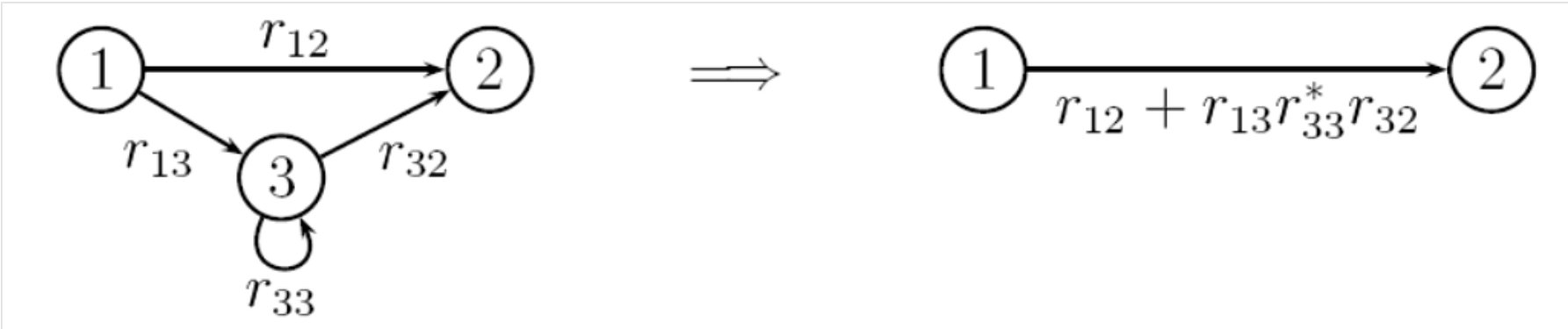


$$L(M) = L(M_1)^*$$

Ισοδυναμία κανονικών παραστάσεων και αυτομάτων (συν.)

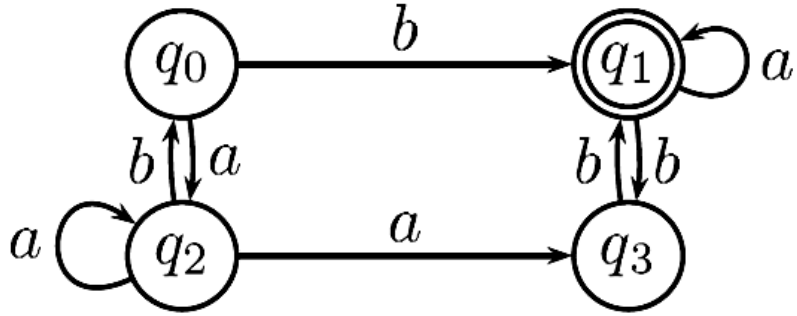
‘<=’ : Κατασκευή κανονικής παράστασης από FA (GNFA).

Απαλείφουμε ενδιάμεσες καταστάσεις σύμφωνα με το σχήμα:

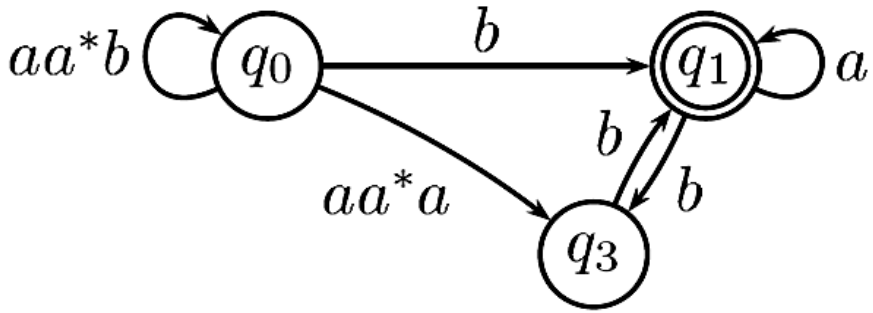


Παράδειγμα κατασκευής κανονικής παράστασης από FA

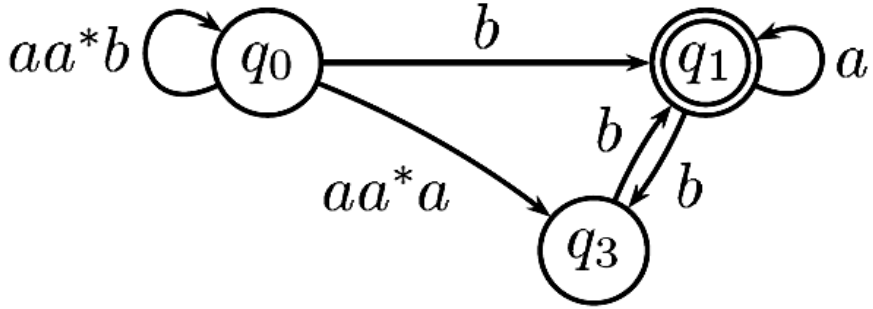
Αρχικό DFA



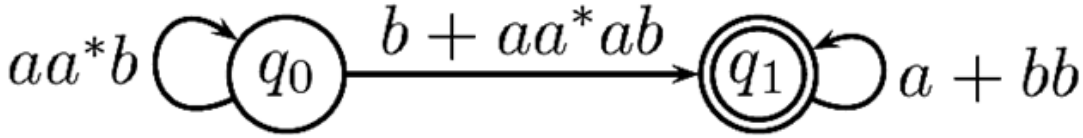
Μετά από διαγραφή q_2



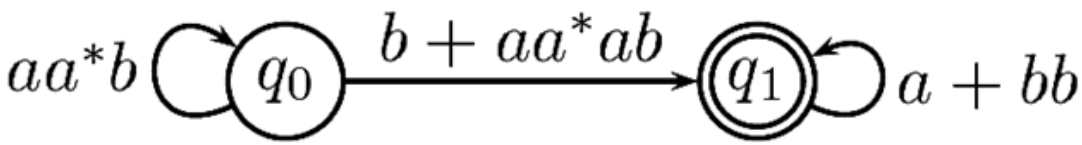
Παράδειγμα κατασκευής κανονικής παράστασης από FA



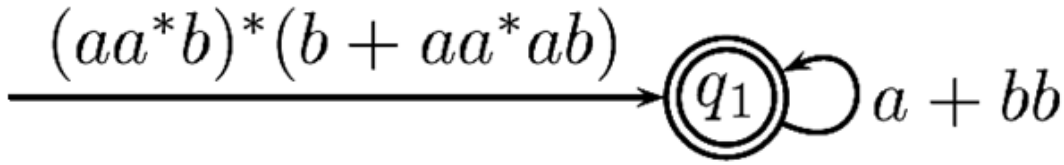
Διαγραφή q_3



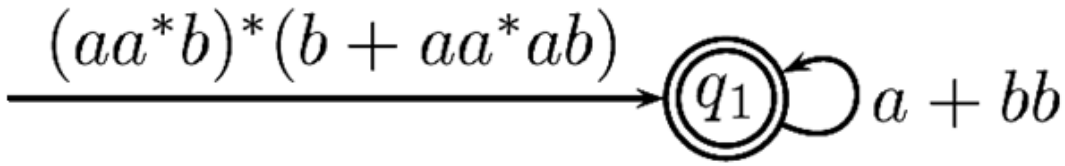
Παράδειγμα κατασκευής κανονικής παράστασης από FA



Διαγραφή q_0



Παράδειγμα κατασκευής κανονικής παράστασης από FA



Τελική παράσταση

$$(aa^*b)^*(b + aa^*ab)(a + bb)^*$$

Ποιες γλώσσες είναι κανονικές;

- Όλες οι πεπερασμένες.
- Όσες σχηματίζονται από κανονικές με τις πράξεις: παράθεση, ένωση, άστρο Kleene,
- αλλά και συμπλήρωμα, τομή, αναστροφή (άσκηση), κ.ά.
- Γινόμενο αυτομάτων (product of automata): τρόπος κατασκευής DFA για τομή (αλλά και ένωση) κανονικών γλωσσών.

Γινόμενο αυτομάτων DFA

- Έστω δύο DFA M_1, M_2 με *n*, *m* καταστάσεις αντίστοιχα ($Q_1=\{q_0, ..., q_{n-1}\}$, $Q_2=\{p_0, ..., p_{m-1}\}$) και κοινό αλφάβητο, που αναγνωρίζουν γλώσσες L_1, L_2 αντίστοιχα.
- Το **γινόμενο των M_1, M_2** είναι ένα DFA με $m \cdot n$ καταστάσεις, μία για κάθε ζεύγος καταστάσεων του αρχικού αυτομάτου (σύνολο καταστάσεων $Q = Q_1 \times Q_2$), το ίδιο αλφάβητο και αρχική κατάσταση (q_0,p_0).
- Συνάρτηση μετάβασης: $\delta'((q_i,p_k),\sigma)=(q_{i'}, p_{k'}) \Leftrightarrow \delta(q_i,\sigma)=q_{i'} \wedge \delta(p_k,\sigma)=p_{k'}$
- Τελικές καταστάσεις: ανάλογα με την πράξη μεταξύ L_1, L_2 που θέλουμε. Για τομή θέτουμε ως τελικές ζεύγη όπου και οι δύο τελικές στα M_1, M_2, για ένωση ζεύγη που περιέχουν μία τουλάχιστον τελική.
- Παρατήρηση: εύκολη υλοποίηση και άλλων πράξεων μεταξύ L_1, L_2 (διαφορά, συμμετρική διαφορά) με κατάλληλο ορισμό των τελικών καταστάσεων.

Γινόμενο αυτομάτων NFA

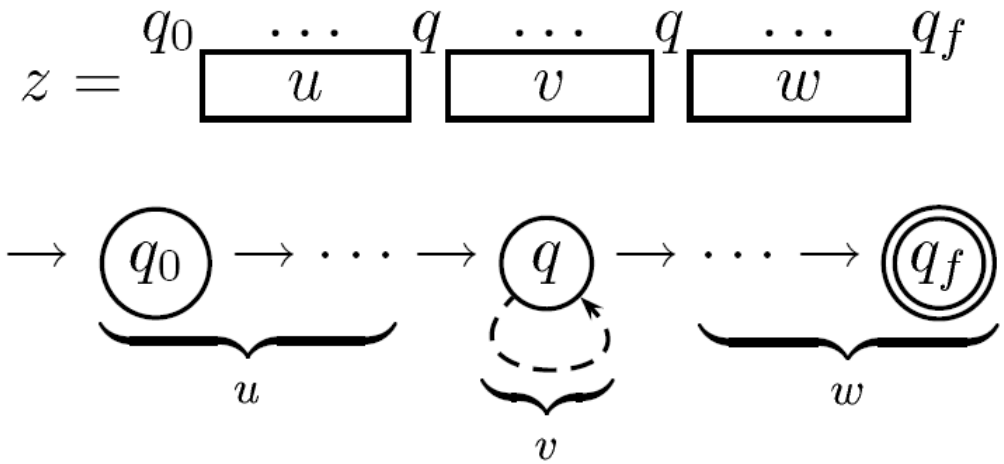
- Ορίζεται με παρόμοιο τρόπο.
- Χρειάζεται προσοχή στις ε-κινήσεις και στο συνδυασμό μεταβάσεων σε junk states με κανονικές μεταβάσεις.

Είναι όλες οι γλώσσες κανονικές;

- Η απάντηση είναι «όχι»
- Για να το αποδείξουμε χρησιμοποιούμε ένα σημαντικό θεώρημα που λέγεται Pumping Lemma (Λήμμα Άντλησης)

Pumping Lemma (διαίσθηση)

- Αν μια γλώσσα *L* είναι κανονική τότε την αποδέχεται ένα DFA με πεπερασμένο αριθμό καταστάσεων, έστω *n*.
- Έστω λέξη *z* με |*z*|>=*n* που ανήκει στη γλώσσα, άρα γίνεται αποδεκτή από το αυτόματο.
- Καθώς επεξεργαζόμαστε το *z*, το αυτόματο πρέπει να περάσει ξανά από κάποια κατάσταση (αρχή περιστερώνα):



- Αφού *z* = *uvw* ∈ *L* θα πρέπει και $uv^iw \in L$, για κάθε *i* ∈ N

Pumping Lemma (με λόγια)

Έστω κανονική γλώσσα L. Τότε:

- υπάρχει ένας φυσικός n (= πλήθος καταστάσεων του DFA) ώστε:
- για κάθε $z \in L$ με μήκος $|z| \geq n$
- υπάρχει «σπάσιμο» του z σε u, v, w, δηλαδή $z = uvw$, με $|uv| \leq n$ και $|v| > 0$
- ώστε για κάθε $i = 0, 1, 2, \ldots$:

$$uv^iw \in L$$

Απόδειξη ότι μια γλώσσα δεν είναι κανονική

Χρήση του Pumping Lemma για να δείξουμε ότι μια (μη πεπερασμένη) γλώσσα *L δεν είναι κανονική*:

Έστω η *L* κανονική. Τότε:

- το PL λέει ότι υπάρχει *n*. Εμείς για κάθε *n*
- επιλέγουμε κατάλληλο $z \in L$ με μήκος $|z| \geq n$
- το PL λέει ότι υπάρχει «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$. Εμείς για κάθε «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$
- επιλέγουμε *i* ώστε η λέξη uv^iw να μην είναι στη γλώσσα *L* ΑΤΟΠΟ

(*adversary argument*)

Παράδειγμα χρήσης Pumping Lemma (i)

- **Θεώρημα.** *Η γλώσσα* $L = \{z \mid z$ *έχει το ίδιο πλήθος 0 και 1*$\}$ *δεν είναι κανονική*.

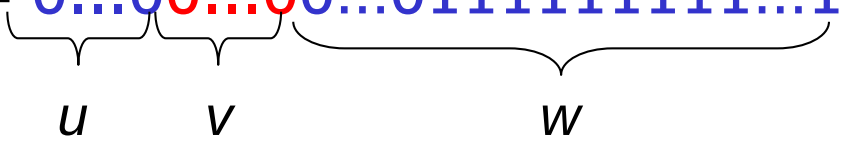
- *Απόδειξη*: Έστω L κανονική. Τότε:
  - το PL λέει ότι υπάρχει n. Εμείς για κάθε n
  - επιλέγουμε $z = 0^n1^n \in L$ με μήκος $|z| = 2n > n$

$$z = \underbrace{000000000...0}_{n}\underbrace{1111111111...1}_{n}$$

  - το PL λέει ότι υπάρχει «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$. Εμείς για κάθε «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$

Παράδειγμα χρήσης Pumping Lemma (ii)

- παρατηρούμε ότι αναγκαστικά $v = 0^k$ για κάποιο k:
  - $w = 0\ldots0\,0\ldots0\,0\ldots0\,1\,1\,1\,1\,1\,1\,1\,1\ldots1$



- και επιλέγουμε $i = 2$, διαπιστώνοντας ότι $uv^iw = uv^2w$ δεν ανήκει στην L.

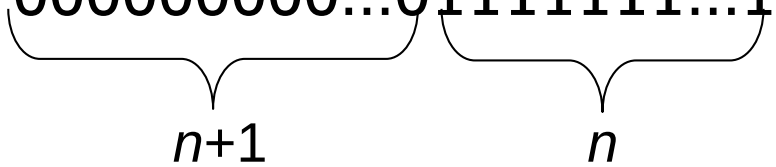
ΑΤΟΠΟ

- Επομένως η L δεν είναι κανονική.

Δεύτερο παράδειγμα χρήσης PL (i)

Θεώρημα. *Η γλώσσα* $L = \{z \mid z=0^i1^j, i > j\}$ *δεν είναι κανονική.*

Απόδειξη: Έστω L κανονική. Τότε:

- το PL λέει ότι υπάρχει n. Εμείς για κάθε n
- επιλέγουμε $z = 0^{n+1}1^n \in L$ με μήκος $|z| = 2n+1 > n$
  - $z =$ 

- το PL λέει ότι υπάρχει «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$. Εμείς για κάθε «σπάσιμο» $z = uvw$, με $|uv| \leq n$ και $|v| > 0$

Δεύτερο παράδειγμα χρήσης PL (ii)

- παρατηρούμε ότι αναγκαστικά $v = 0^k$ για κάποιο k:

$$z = \underbrace{0\ldots0}_{u}\underbrace{0\ldots0}_{v}\underbrace{0\ldots0111111111\ldots1}_{w}$$

  - όμως, η επανάληψη του v δίνει λέξεις της γλώσσας
  - από πρώτη άποψη αυτό φαίνεται προβληματικό…
  - όμως το λήμμα ορίζει ότι θα πρέπει για κάθε $i \geq 0$: $uv^iw \in L$

- επιλέγουμε $i = 0$: η λέξη uv^0w δεν είναι στην L.

ΑΤΟΠΟ

- Επομένως η L δεν είναι κανονική.

Προσοχή στη χρήση του PL

- Το Pumping Lemma είναι αναγκαία αλλά όχι και ικανή συνθήκη για να είναι μια γλώσσα κανονική.
- Υπάρχουν μη κανονικές γλώσσες που ικανοποιούν τις συνθήκες του!
- Επομένως χρησιμεύει μόνο για απόδειξη μη κανονικότητας.

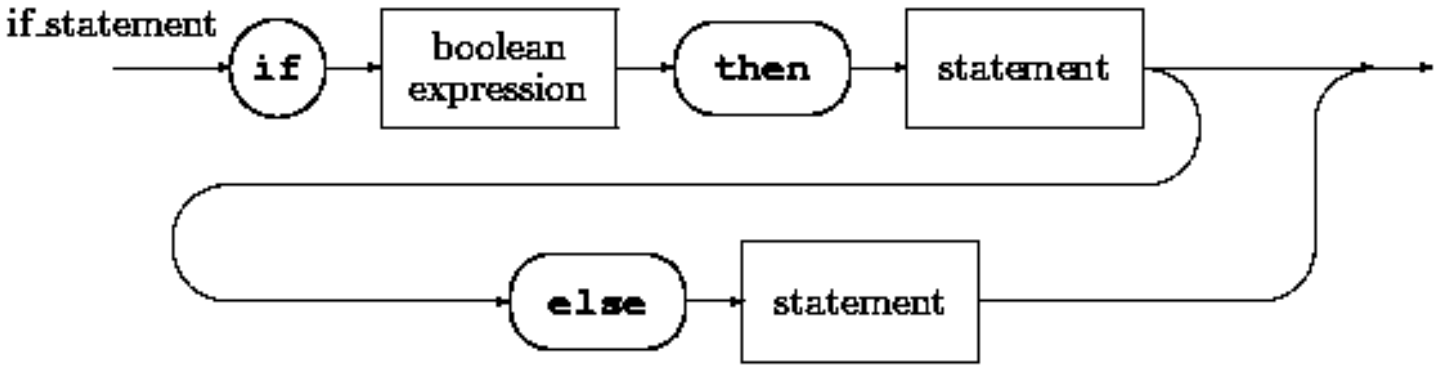
Γραμματικές για μη κανονικές γλώσσες

- Χωρίς συμφραζόμενα (context free, CF): τύπου 2, αντιστοιχία με αυτόματα στοίβας (pushdown automata, PDA)
- Με συμφραζόμενα (context sensitive, CS): τύπου 1, αντιστοιχία με γραμμικά περιορισμένα αυτόματα (linear bounded automata, LBA)
- Γενικές (general): τύπου 0, αντιστοιχία με μηχανές Turing (Turing machines, TM)

Γραμματικές χωρίς συμφραζόμενα (Context Free) (i)

Εφαρμογές σε:

- συντακτικό γλωσσών προγραμματισμού (Pascal, C, C++, Java)



- συντακτικό γλωσσών περιγραφής σελίδων web (HTML, XML), editors, ...

Γραμματικές χωρίς συμφραζόμενα (Context Free) (ii)

- Μορφή κανόνων: $A \to \alpha$, A μη τερματικό, $\alpha \in (V \cup T)^*$
- Παράδειγμα:

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

Δυνατή ακολουθία παραγωγής:

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

Γλώσσα που παράγεται:

$$L(G_1) = \{a^n b^n | n \in \mathbb{N}^*\}$$

Γραμματικές χωρίς συμφραζόμενα (Context Free) (iii)

- 2ο παράδειγμα:

G_2: $T = \{0,1,2,3,4,5,6,7,8,9,+,*\}$ $V = \{S\}$

P : $S \rightarrow S+S,$ $S \rightarrow S*S,$

$S \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Δυνατές ακολουθίες παραγωγής:

$S \Rightarrow 3,$ $S \Rightarrow S+S \Rightarrow 3+S \Rightarrow 3+S*S \Rightarrow 3+4*7$

Γραμματικές χωρίς συμφραζόμενα (Context Free) (iv)

- 3ο παράδειγμα:

G_3: $V = \{S, A, B\}$, $T = \{a, b\}$, και P περιέχει:

$$S \to aB \mid bA, \quad A \to a \mid aS \mid bAA, \quad B \to b \mid bS \mid aBB$$

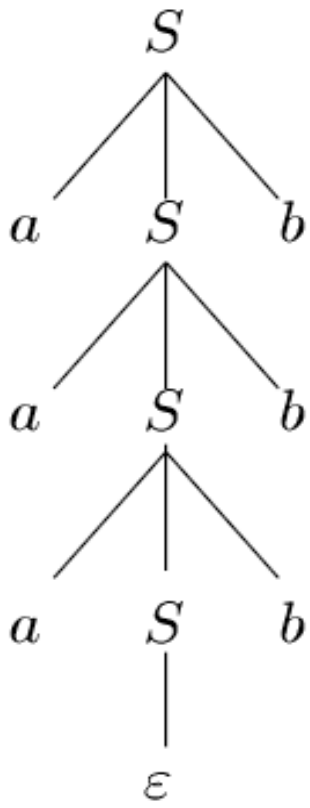
Δυνατή ακολουθία παραγωγής:

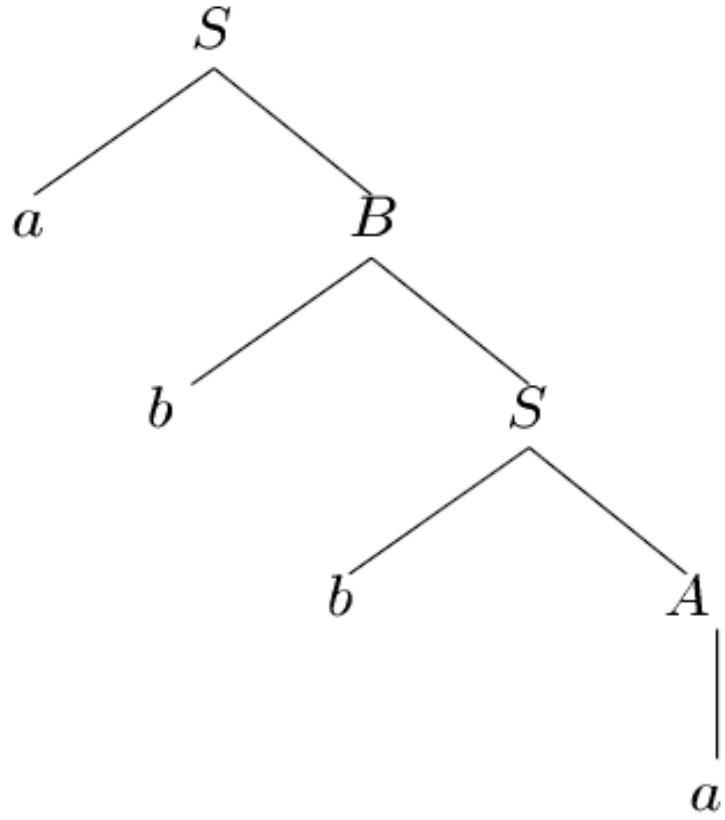
$$S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba$$

Γλώσσα που παράγεται (όχι προφανές):

$$L(G_3) = \{w \in T^+ \mid w \text{ έχει ίσο αριθμό } a \text{ και } b\}$$

Συντακτικά Δένδρα (parse trees) (i)





Φύλλωμα (leafstring): *αααbbb* και *abba* αντίστοιχα.

Συντακτικά Δένδρα (parse trees) (ii)

Έστω $G=\{V,T,P,S\}$ μια γραμματική χωρίς συμφραζόμενα.
Ένα δένδρο είναι συντακτικό δένδρο της G αν:

- Κάθε κόμβος του δένδρου έχει επιγραφή, που είναι σύμβολο (τερματικό ή μη τερματικό ή ε).
- Η επιγραφή της ρίζας είναι το S.
- Αν ένας εσωτερικός κόμβος έχει επιγραφή A, τότε το A είναι μη τερματικό σύμβολο. Αν τα παιδιά του, από αριστερά προς τα δεξιά, έχουν επιγραφές $X_1,X_2,\ldots,X_k$ τότε ο $A \rightarrow X_1,X_2,\ldots,X_k$ είναι κανόνας παραγωγής.
- Αν ένας κόμβος έχει επιγραφή ε, τότε είναι φύλλο και είναι το μοναδικό παιδί του γονέα του.

Συντακτικά Δένδρα (parse trees) (iii)

Θεώρημα. *Έστω $G=\{V,T,P,S\}$ μια γραμματική χωρίς συμφραζόμενα. Τότε $S \stackrel{*}{\Rightarrow} w$ αν και μόνο αν υπάρχει συντακτικό δένδρο της G με φύλλωμα w.*

Απόδειξη:

‘<=’ : Με επαγωγή ως προς τον αριθμό των εσωτερικών κόμβων.

‘=>’ : Με επαγωγή ως προς των αριθμό των βημάτων της ακολουθίας παραγωγών (άσκηση).

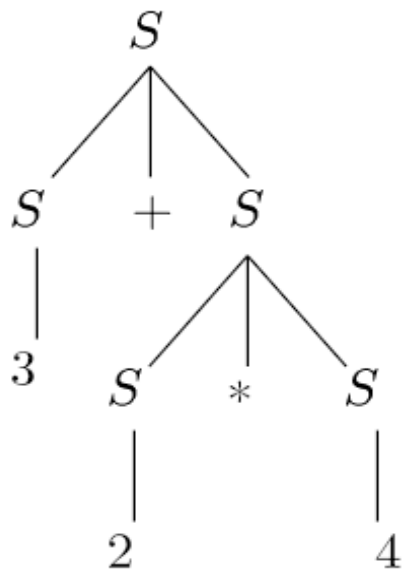
Διφορούμενες γραμματικές

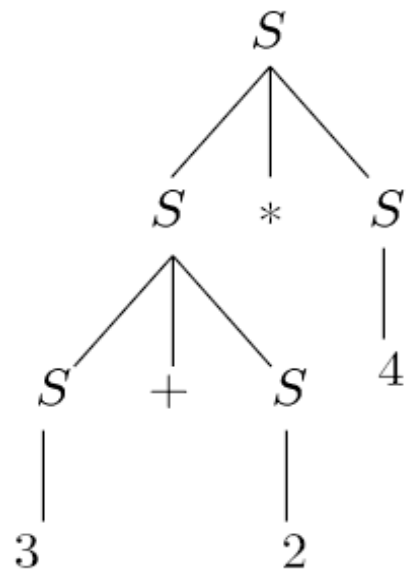
Μια γραμματική G ονομάζεται διφορούμενη (ambiguous) αν υπάρχουν δύο συντακτικά δένδρα με το ίδιο φύλλωμα $w \in L(G)$

Παράδειγμα:

G_2: $T = \{0,1,2,3,4,5,6,7,8,9,+,*\}$ $V = \{S\}$

$P: S \rightarrow S+S \mid S*S \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$





Αλγόριθμος αναγνώρισης για CF γραμματικές: CYK

- Με εξαντλητικό τρόπο μπορούμε να αποφασίσουμε αν μια συμβολοσειρά *x* παράγεται από μια γραμματική CF (χωρίς συμφραζόμενα) σε εκθετικό όμως χρόνο.
- Οι ιδιότητες της κανονικής μορφής Chomsky επιτρέπουν ταχύτερη αναγνώριση μιας συμβολοσειράς.
- Αλγόριθμος CYK (Cocke, Younger, Kasami): αποφασίζει αν μια συμβολοσειρά *x* παράγεται από μια γραμματική σε χρόνο $O(|x|^3)$, αρκεί η γραμματική να δίνεται σε Chomsky Normal Form.

Αυτόματα Στοίβας (PDA) (i)

- Έχουν ταινία εισόδου μιας κατεύθυνσης (όπως και τα FA) αλλά επιπλέον μνήμη υπό μορφή στοίβας.
- Πρόσβαση μόνο στην κορυφή της στοίβας με τις λειτουργίες:
  - push(x): τοποθετεί στοιχείο x στην κορυφή της στοίβας
  - pop: διαβάζει και αφαιρεί στοιχείο από την κορυφή της στοίβας

Αυτόματα Στοίβας (PDA) (ii)

Παράδειγμα: PDA για αναγνώριση της γλώσσας

$$L = \{wcw^R \mid w \in (0+1)^*\}$$

Περιγραφή αυτομάτου

- push(*a*) στη στοίβα για κάθε 0 στην είσοδο, push(*b*) στη στοίβα για κάθε 1 στην είσοδο, συνέχισε μέχρι να διαβαστεί *c*
- μετά pop: εφόσον πάνω στοιχείο στοίβας συμφωνεί με είσοδο (*a* με 0, *b* με 1) συνέχισε
- Αποδοχή με *κενή στοίβα*

Τυπικός ορισμός PDA

Αυτόματο στοίβας (Pushdown Automaton, PDA):
επτάδα $M = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$

- Q : το σύνολο των καταστάσεων του M (πεπερασμένο)
- Σ : αλφάβητο εισόδου
- Γ : αλφάβητο στοίβας
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \rightarrow \text{Pow}(Q \times \Gamma^*)$: συνάρτηση μετάβασης (μη ντετερμινισμός, ε-κινήσεις)
- $q_0 \in Q$: αρχική κατάσταση
- $Z_0 \in \Gamma$: αρχικό σύμβολο στοίβας
- $F \subseteq Q$: σύνολο τελικών καταστάσεων

Αυτόματα Στοίβας (PDA) (iv)

Είδη αποδοχής PDA

- Αν βρεθεί σε τελική κατάσταση (δηλ. αποδοχής) μόλις διαβαστεί όλη η είσοδος, ανεξαρτήτως περιεχομένου στοίβας
- Αν βρεθεί με κενή στοίβα μόλις διαβαστεί όλη η είσοδος, ανεξαρτήτως κατάστασης

Αντίστοιχα ορίζονται οι *γλώσσες*:

- $L_f(M)$: αποδοχή με τελική κατάσταση
- $L_e(M)$: αποδοχή με κενή στοίβα

Αυτόματα Στοίβας (PDA) (v)

- Για να γίνει αποδεκτή η γλώσσα

$$L_1 = \{ww^R \mid w \in (0+1)^*\}$$

χωρίς δηλαδή ειδικό μεσαίο σύμβολο *c* χρειαζόμαστε απαραίτητα μη ντετερμινιστικό PDA.

- Τα μη ντετερμινιστικά PDA είναι γνησίως πιο ισχυρά από τα ντετερμινιστικά.

- Με τον όρο PDA αναφερόμαστε συνήθως στα μη-ντετερμινιστικά PDA.

Ισοδυναμία CF γραμματικών και PDA

Θεώρημα. Τα παρακάτω είναι ισοδύναμα για μια γλώσσα *L:*

- $L = L_f(M)$, M είναι PDA.
- $L = L_e(M')$, M' είναι PDA.
- Η L είναι γλώσσα χωρίς συμφραζόμενα (c.f.)

Ποιες γλώσσες είναι Context Free;

- Όλες οι κανονικές.
- Επίσης όσες σχηματίζονται από γλώσσες CF με τις πράξεις: παράθεση, ένωση, άστρο Kleene.
- Αλλά όχι απαραίτητα με τις πράξεις τομή, συμπλήρωμα:

  π.χ. η γλώσσα $\{a^n b^n c^n \mid n \in \mathbf{N}\}$ δεν είναι CF, ενώ είναι τομή δύο CF γλωσσών:

  $\{a^n b^n c^n \mid n \in \mathbf{N}\} = \{a^n b^n c^m \mid n,m \in \mathbf{N}\} \cap \{a^k b^n c^n \mid k,n \in \mathbf{N}\}$

Είναι όλες οι γλώσσες Context Free;

- Η απάντηση είναι «όχι».
- Για να το αποδείξουμε χρησιμοποιούμε ένα άλλο λήμμα άντλησης, το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα.
- Βασίζεται στο συντακτικό δένδρο (περισσότερα στο μάθημα «Υπολογισιμότητα»).

Γενικές Γραμματικές (i)

τύπου 0: γενικές γραμματικές (general, phrase structure, semi-Thue).

$$\alpha \rightarrow \beta,\ \alpha \neq \varepsilon$$

Παράδειγμα: $\{a^{2^n} \mid n \in \mathbf{N}\}$

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

Γενικές Γραμματικές (ii)

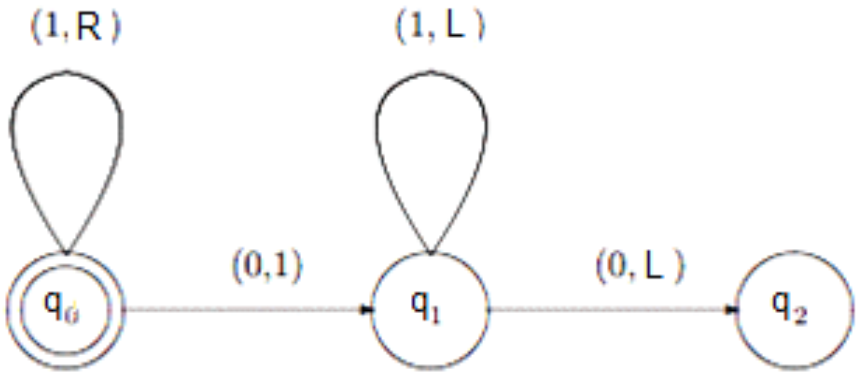
Θεώρημα. Τα παρακάτω είναι ισοδύναμα:

1. Η γλώσσα *L* γίνεται αποδεκτή από μια μηχανή Turing
2. *L=L(G)*, όπου *G* είναι γενική γραμματική

Μια τέτοια γλώσσα λέγεται και *αναδρομικά αριθμήσιμη* (recursively enumerable).

Μηχανές Turing

Αυτόματα με απεριόριστη ταινία. Η είσοδος είναι αρχικά γραμμένη στην ταινία, η κεφαλή μπορεί να κινείται αριστερά-δεξιά, καθώς και να αλλάζει το σύμβολο που διαβάζει.
Παράδειγμα συνάρτησης μετάβασης:



$< q_0, 1, q_0, R >$
$< q_0, 0, q_1, 1 >$
$< q_1, 1, q_1, L >$
$< q_1, 0, q_2, R >$

Γραμματικές με Συμφραζόμενα (context sensitive) (i)

τύπου 1: γραμματικές με συμφραζόμενα ή μονοτονικές (context sensitive, monotonic).

$$\alpha \to \beta, \quad |\alpha| \leq |\beta| \text{ (επιτρέπεται και: } S \to \varepsilon) \quad \alpha \neq \varepsilon$$

Λέγονται «με συμφραζόμενα» γιατί μπορούν να τεθούν στην εξής κανονική μορφή:

$$\underset{\substack{\nwarrow \quad \nearrow \\ context}}{\alpha_1 A \alpha_2} \to \alpha_1 \beta \alpha_2, \qquad \text{όπου } A\text{: μη τερματικό και } \beta \neq \varepsilon$$

Γραμματικές με Συμφραζόμενα (context sensitive) (ii)

Γραμματική c.s. για τη γλώσσα $1^n 0^n 1^n$:

$S \to 1Z1$

$Z \to 0 \mid 1Z0A$

$A0 \to 0A$

$A1 \to 11$

Μετατροπή σε κανονική μορφή:

$A0 \to H0$ $\quad$ $H0 \to HA$ $\quad$ $HA \to 0A$

Άλλα παραδείγματα: $\{1^i\, 0^j\, 1^k : i <= j <= k\}$,

$\{ ww \mid w \in \Sigma^* \}$, $\quad$ $\{a^n\, b^n\, a^n\, b^n \mid n \in \mathbf{N}\}$

Ισοδυναμία γραμματικών CS και LBA

Γραμμικά φραγμένο αυτόματο (*Linear Bounded Automaton*, LBA):

είναι μια μη ντετερμινιστική μηχανή Turing που η κεφαλή της είναι περιορισμένη να κινείται μόνο στο τμήμα που περιέχει την αρχική είσοδο.

Θεώρημα. Τα παρακάτω είναι ισοδύναμα (*L* χωρίς ε):

1. Η γλώσσα *L* γίνεται αποδεκτή από LBA.
2. Η γλώσσα *L* είναι context sensitive.

Παραλλαγές και επεκτάσεις αυτομάτων I

Ορισμός

Ένα **two-way deterministic FA** (2DFA) είναι μία πεντάδα $M = (Q, \Sigma, \delta, q_0, F)$, όπου τα Q, Σ, q_0 και F είναι όπως προηγουμένως και δ είναι μία απεικόνιση από το $Q \times \Sigma$ στο $Q \times \{L, R\}$, δηλαδή $\delta \colon Q \times \Sigma \to Q \times \{L, R\}$.
Εάν $\delta(q, a) = (p, L)$, τότε στην κατάσταση q, διαβάζοντας το σύμβολο a, το 2DFA πάει στην κατάσταση p και μετακινεί την κεφαλή προς τα αριστερά κατά μία θέση. Εάν $\delta(q, a) = (p, R)$, τότε το 2DFA πάει στην κατάσταση p και μετακινεί την κεφαλή προς τα δεξιά κατά μία θέση.

Παραλλαγές και επεκτάσεις αυτομάτων II

Configuration ή instantaneous description (ID) ενός 2DFA: περιγράφει το **string εισόδου**, την **τρέχουσα κατάσταση** και την **τρέχουσα θέση της κεφαλής εισόδου**.

Ένα ID του M είναι ένα string στο $\Sigma^* Q \Sigma^*$.
Το ID wqx, όπου $w, x \in \Sigma^*$, αντιπροσωπεύει τα εξής:

1. wx είναι το string εισόδου,
2. q είναι η τρέχουσα κατάσταση και
3. η κεφαλή εισόδου διαβάζει το πρώτο σύμβολο του x.

Εάν το $x = \varepsilon$, τότε η κεφαλή εισόδου έχει μετακινηθεί στο δεξί άκρο της ταινίας εισόδου.

Παραλλαγές και επεκτάσεις αυτομάτων III

Στη συνέχεια εισάγουμε την σχέση $\vdash_M$ στα ID's, έτσι ώστε $I_1 \vdash_M I_2$, αν και μόνο αν το M μπορεί να πάει από την instantaneous description I_1 στην I_2 με μία κίνηση.

Ορίζουμε τη σχέση $\vdash_M$, ή απλά $\vdash$, αν το M εννοείται, ως εξής:

1. $a_1a_2 \ldots a_{i-1}qa_i \ldots a_n \vdash a_1a_2 \ldots a_{i-1}a_ipa_{i+1} \ldots a_n$, όταν $\delta(q, a_i) = (p, R)$ και
2. $a_1a_2 \ldots a_{i-2}a_{i-1}qa_i \ldots a_n \vdash a_1a_2 \ldots a_{i-2}pa_{i-1}a_i \ldots a_n$, όταν $\delta(q, a_i) = (p, L)$ και $i > 1$.

Παραλλαγές και επεκτάσεις αυτομάτων IV

- Στα διαγράμματα, μία ακμή εκτός από το σύμβολο εισόδου έχει επίσης επιγραφή L ή R.
- Το ανακλαστικό, μεταβατικό κλείσιμο της σχέσης $\vdash$ είναι το $\vdash^*$.
- Computation is a legal finite sequence of configurations.

Ορισμός

$L(M) = \{w \mid q_0 w \vdash^* wp$ για κάποιο p στο $F\}$.

Θεώρημα

Έστω M ένα 2DFA. Τότε $L(M)$ είναι κανονική.

FA με έξοδο I

Μηχανή Moore

Ορισμός

Μία **μηχανή Moore** είναι μία εξάδα $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$, όπου τα Q, Σ, δ και q_0 είναι όπως στα DFA, με $\delta : Q \times \Sigma \to Q$. Επιπλέον:

- Δ: αλφάβητο εξόδου
- λ: απεικόνιση από το Q στο Δ, που δίνει την έξοδο που σχετίζεται με κάθε κατάσταση, δηλαδή $\lambda : Q \to \Delta$.

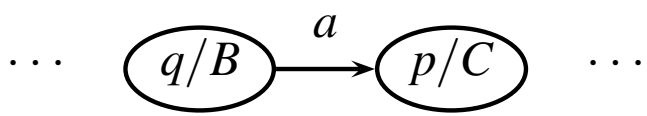
input		a_1		a_2	...	a_n	
states	q_0	q_1	q_2	...	q_{n-1}	q_n	μήκος $n+1$
output	$\lambda(q_0)$	$\lambda(q_1)$	$\lambda(q_2)$	...	$\lambda(q_{n-1})$	$\lambda(q_n)$	μήκος $n+1$

FA με έξοδο II

Μηχανή Moore

Παρατηρήσεις:

- Κάθε μηχανή Moore δίνει έξοδο $\lambda(q_0) = \flat$ για είσοδο ε.
- Το DFA μπορεί να θεωρηθεί ως μία ειδική περίπτωση μίας μηχανής Moore όπου το αλφάβητο είναι $\{0,1\}$ και η κατάσταση q αποδέχεται αν και μόνο αν $\lambda(q) = 1$.
- Μία μηχανή Moore δεν έχει τελικές καταστάσεις και έχει μήκος εξόδου $n + 1$.
- Τα διαγράμματα μηχανών Moore είναι όπως αυτά των FA, αλλά επιπλέον σημειώνουμε μαζί με την κατάσταση το αντίστοιχο σύμβολο εξόδου ως εξής $q_i/(q_i)$, π.χ.:



FA με έξοδο I

Μηχανή Mealy

Ορισμός

Μία **μηχανή Mealy** είναι μια εξάδα $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$, όπου όλα είναι όπως στη μηχανή Moore, με $\delta\colon Q \times \Sigma \to Q$, εκτός από το λ που είναι απεικόνιση από το $Q \times \Sigma$ στο Δ, δηλαδή $\lambda\colon Q \times \Sigma \to \Delta$.

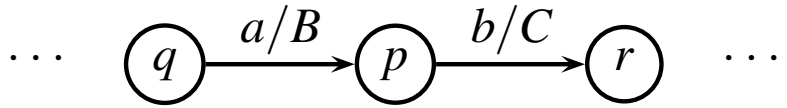
Η έξοδος του M με είσοδο το $a_1 a_2 \dots a_n$ είναι $\lambda(q_0, a_1)\lambda(q_1, a_2)\dots\lambda(q_{n-1}, a_n)$, όπου $q_0, q_1, \dots, q_n$ είναι η ακολουθία των καταστάσεων έτσι ώστε $\delta(q_{i-1}, a_i) = q_i$, για $1 \leqslant i \leqslant n$.

FA με έξοδο II

Μηχανή Mealy

Παρατηρήσεις:

- Η ακολουθία εξόδου έχει μήκος n και όχι $n+1$ όπως η μηχανή Moore και με είσοδο ε μία μηχανή Mealy δίνει έξοδο ε.
- Τα διαγράμματα μηχανών Mealy είναι όπως αυτά των FA, αλλά επιπλέον σημειώνουμε στις μεταβάσεις την αντίστοιχη έξοδο, π.χ.:



- Μία γενίκευση των μηχανών Mealy είναι το **Finite Transducer** (ή finite state machine, FSM) με συνάρτηση: $\lambda \colon Q \times \Sigma \to \Delta^*$.

Ισοδυναμία μηχανών Moore και Mealy I

Ορισμός

Έστω M μία μηχανή Moore ή Mealy. Ορίζουμε $T_M(w)$, για είσοδο w να είναι η έξοδος που παράγεται από την M με είσοδο w.

Παρατήρηση

Δεν είναι δυνατόν ποτέ να είναι ίδιες οι συναρτήσεις $T_M(w), T_{M'}(w)$, αν M είναι μία μηχανή Moore και M' μία μηχανή Mealy, επειδή $|T_M(w)|$ είναι ένα περισσότερο από $|T_{M'}(w)|$, για κάθε w. Ωστόσο, μπορούμε να δώσουμε τον παρακάτω ορισμό ισοδυναμίας:

Ορισμός

Μία μηχανή Mealy M και μία μηχανή Moore M' είναι ισοδύναμες:
$\forall w$: $\flat T_M(w) = T_{M'}(w)$, όπου $\lambda'(q_0) = \flat$ και $T_{M'}(\varepsilon) = \flat$,
$T_M(w) =$ η έξοδος μετά την επεξεργασία του w.

Ισοδυναμία μηχανών Moore και Mealy II

Μπορούμε να αποδείξουμε το ακόλουθο θεώρημα για την ισοδυναμία μεταξύ μηχανών Moore και Mealy:

Θεώρημα

$\forall M$ *Moore* $\Rightarrow$ $\exists M'$ *Mealy ισοδύναμη.*
$\forall M$ *Mealy* $\Rightarrow$ $\exists M'$ *Moore ισοδύναμη.*

Απόδειξη.

1. Έστω μία μηχανή Moore M_1. Κατασκευάζουμε μία ισοδύναμη μηχανή Mealy M_2: ορίζουμε $\lambda'(q,a) = \lambda(\delta(q,a))$ για όλες τις καταστάσεις q και για όλα τα σύμβολα εισόδου a.
2. Έστω M_1 μηχανή Mealy. Τότε $M_2 = \{Q \times \Delta, \Sigma, \Delta, \delta', \lambda', [q_0, b_0]\}$, όπου b_0 είναι τυχαίο στοιχείο του . Αυτό σημαίνει ότι οι καταστάσεις του M_2 είναι ζευγάρια της μορφής $[q,b]$, αποτελούμενα από μία κατάσταση της M_1 και ένα σύμβολο εξόδου. Ορίζουμε $\delta'([q,b],a) = [\delta(q,a), \lambda(q,a)]$ και $\lambda'([q,b]) = b$. Το δεύτερο στοιχείο μίας κατάστασης $[q,b]$ της M_2 είναι η έξοδος που παράγεται από την M_1 σε κάποια μετάβαση στην κατάσταση q. Μόνο τα πρώτα στοιχεία των καταστάσεων της M_2 καθορίζουν τις κινήσεις της M_2. Εύκολα με επαγωγή στο n δείχνεται ότι αν η M_1 πηγαίνει στις καταστάσεις $q_0, q_1, \ldots, q_n$ με είσοδο $a_1 a_2 \ldots a_n$ και δίνει έξοδο $b_1, b_2, \ldots, b_n$, τότε η M_2 πηγαίνει στις καταστάσεις $[q_0, b_0], [q_1, b_1], \ldots, [q_n, b_n]$ και δίνει έξοδο $b_0, b_1, \ldots, b_n$.

□

Pumping lemma για κανονικά σύνολα Ι

Λήμμα (Pumping lemma)

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

Pumping lemma για κανονικά σύνολα II

Χρησιμοποιώντας το λήμμα δείχνουμε ότι ένα δοσμένο σύνολο δεν είναι regular. Η μέθοδος είναι η ακόλουθη:

1. Διαλέγεις τη γλώσσα που θέλεις να αποδείξεις πως δεν είναι regular.
2. Ο αντίπαλος (PL) επιλέγει ένα n. Θα πρέπει να μπορείς για οποιοδήποτε πεπερασμένο ακέραιο n διαλέξει, να αποδείξεις ότι η L δεν είναι regular, αλλά από τη στιγμή που ο αντίπαλος έχει διαλέξει ένα n αυτό είναι σταθερό στην απόδειξη.
3. Διαλέγεις ένα string z της L έτσι ώστε $|z| \geqslant n$.
4. Ο αντίπαλος (PL) σπάει το z σε u, v και w που ικανοποιούν τους περιορισμούς $|uv| \leqslant n$ και $|v| \geqslant 1$.
5. Φτάνεις σε αντίφαση δείχνοντας ότι για κάθε u, v, w που καθορίζονται από τον αντίπαλο, υπάρχει ένα i για το οποίο $uv^i w$ δεν ανήκει στην L. Τότε μπορούμε να συμπεράνουμε ότι η L δεν είναι regular. Η επιλογή του i μπορεί να εξαρτάται από τα n, u, v, w.

Pumping lemma για κανονικά σύνολα III

Παράδειγμα

- $L = \{a^k b^k \mid k \in \mathbb{N}\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$
  1. Υποθέτουμε ότι L είναι regular και χρησιμοποιούμε το pumping lemma.
  2. PL: $\exists n \in N$
  3. Διαλέγουμε $z = a^n b^n$. Εντάξει επιλογή, διότι $z \in L$, $|z| = 2n \geqslant n$.
  4. PL: z μπορεί να γραφεί: $z = uvw$ με $|uv| \leqslant n \ \wedge |v| \geqslant 1$, ώστε $v = a^l$ με $l \geqslant 1$.
  5. Διαλέγουμε $i = 2$: $uvvw = a^{n+l} b^n \in L$.

Άτοπο

Pumping lemma για κανονικά σύνολα IV

Παράδειγμα

Το σύνολο $L = \{0^{k^2} \mid k \geqslant 1\} = \{0, 0000, 000000000, \ldots\}$, δεν είναι regular.

1. Υποθέτουμε ότι L είναι regular και χρησιμοποιούμε το pumping lemma.
2. PL: $\exists n \in N$.
3. Διαλέγουμε $z = 0^{n^2}$. Εντάξει επιλογή, διότι $|z| = n^2 \geqslant n$.
4. PL: z μπορεί να γραφεί: $z = uvw$ με $|uv| \leqslant n \ \wedge |v| \geqslant 1$, ώστε $v = 0^l$ με $1 \leqslant l \leqslant n$.
5. Διαλέγουμε $i = 2$: $uvvw = 0^{n^2+l} \in L$, αλλά $n^2 + l$ δεν είναι τέλειο τετράγωνο, διότι: $\underline{n^2} < n^2 + 1 \leqslant n^2 + l \leqslant n^2 + n = n(n+1) < \underline{(n+1)^2}$.

Άτοπο

Ιδιότητες κλειστότητας για κανονικά σύνολα

Λογικές (boolean) πράξεις

Υπενθύμιση: Οι boolean (λογικές) πράξεις είναι *η ένωση, η τομή και το συμπλήρωμα*.

Θεώρημα

*Η κλάση των regular sets είναι μηχανιστικά κλειστή (effectively closed) ως προς τις λογικές πράξεις, την παράθεση (concatenation) και το Kleene *.*

Ιδιότητες κλειστότητας για κανονικά σύνολα Ι

Αντικαταστάσεις και ομομορφισμοί

Αντικατάσταση: $f\colon \Sigma \to \mathrm{Pow}(\Delta^*)$

Ειδική περίπτωση:
Ομομορφισμός: $h\colon \Sigma \to \Delta^*$

Αντίστροφος ομομορφισμός:

$h^{-1}(w) = \{x \mid h(x) = w\}$ για string και
$h^{-1}(L) = \{x \mid h(x) \in L\}$ για γλώσσα.

Ιδιότητες: $h(h^{-1}(L)) \subseteq L$ ενώ $h^{-1}(h(L)) \supseteq L$, για κάθε γλώσσα L.

Η απεικόνιση f μπορεί να επεκταθεί και σε strings ως εξής:

$$\begin{cases} \tilde{f}(\varepsilon) = \varepsilon \\ \tilde{f}(wa) = \tilde{f}(w)f(a) \end{cases}$$

Η f επεκτείνεται και σε γλώσσες: $f^{\times}(L) = \bigcup_{x \in L} f(x)$.

Ιδιότητες κλειστότητας για κανονικά σύνολα II

Αντικαταστάσεις και ομομορφισμοί

Ορισμός

Κανονική αντικατάσταση (regular substitution):
αντικατάσταση (substitution) τέτοια ώστε $\forall a \in \Sigma$: $f(a)$ είναι regular.

Θεώρημα

Η κλάση των regular sets είναι μηχανιστικά κλειστή (effectively closed) ως προς τις πράξεις της κανονικής αντικατάστασης, ομομορφισμού και αντίστροφου ομομορφισμού.

Ιδιότητες κλειστότητας για κανονικά σύνολα

Πηλίκα Γλωσσών (quotients of languages)

Πηλίκο γλωσσών: $L_1/L_2 = \{w \mid \exists v \in L_2 : \ wv \in L_1\}$

Παράδειγμα

$L_1 = ab^*c$ και $L_2 = b^*c$ τότε $L_1/L_2 = ab^*$.

Θεώρημα

Η κλάση των regular sets είναι κλειστή ως προς πηλίκο με τυχαίο σύνολο.

Αλγόριθμοι απόφασης για κανονικά σύνολα Ι

Θεώρημα

Έστω M ένα DFA με $|Q| = n$. Τότε

1. $L(M) \neq \varnothing \Leftrightarrow \exists z \in L(M)$ *με* $|z| < n$.
2. $|L(M)| = \infty \Leftrightarrow \exists z \in L(M)$ *με* $n \leqslant |z| < 2n$.

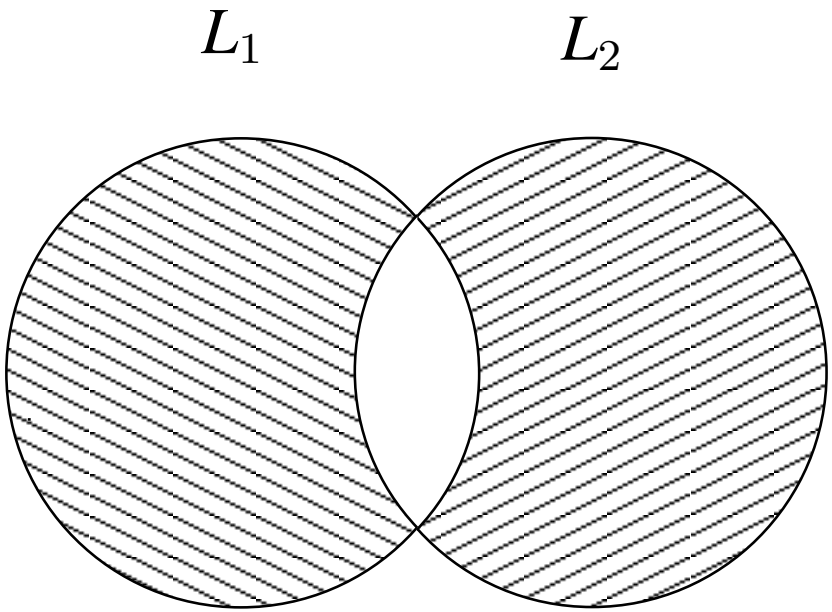
Θεώρημα

Υπάρχει μηχανιστικός αλγόριθμος που αποφασίζει αν δύο πεπερασμένα αυτόματα αποδέχονται το ίδιο σύνολο, δηλαδή αν είναι ισοδύναμα.

Απόδειξη.

Έστω M_1, M_2 τα αυτόματα που αποδέχονται την L_1, L_2 αντίστοιχα. Γνωρίζουμε από θεώρημα ότι υπάρχει αυτόματο M που αποδέχεται το $L = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$. Ισχύει $L = \varnothing \Leftrightarrow L_1 = L_2$. Έτσι από το προηγούμενο θεώρημα υπάρχει αλγόριθμος που αποφασίζει αν $L_1 = L_2$. □

Αλγόριθμοι απόφασης για κανονικά σύνολα II



Σχήμα: Ισοδυναμία πεπερασμένων αυτομάτων

Το θεώρημα Myhill-Nerode I

Για κάθε γλώσσα L θα ορίσουμε μία αντίστοιχη σχέση ισοδυναμίας R_L ως εξής:

Ορισμός

Ορίζουμε για $x, y \in \Sigma^*$, $x \, R_L \, y$, αν και μόνον αν για κάθε $z \in \Sigma^*$, $xz \in L$ ακριβώς όταν $yz \in L$ (δηλαδή για κάθε $z \in \Sigma^*$ είτε αμφότερα τα xz, yz ανήκουν στην γλώσσα είτε κανένα από τα δύο δεν ανήκει στην γλώσσα).

Ο **δείκτης** μίας σχέσης ισοδυναμίας είναι το πλήθος των κλάσεων ισοδυναμίας αυτής. Έτσι, λέμε ότι μία ισοδυναμία είναι *πεπερασμένου δείκτη* αν το πλήθος των κλάσεων ισοδυναμίας της είναι πεπερασμένο.

Παρατήρηση: Όπως θα δείξουμε παρακάτω, κάτι που χαρακτηρίζει κάθε κανονική γλώσσα L είναι ότι η R_L είναι πεπερασμένου δείκτη.

Το θεώρημα Myhill-Nerode II

Επίσης, για κάθε πεπερασμένο αυτόματο M με αλφάβητο Σ ορίζουμε μία αντίστοιχη σχέση R_M στο Σ^* ως εξής:

Ορισμός

$x \, R_M \, y$ αν και μόνον αν $\tilde{\delta}(q_0, x) = \tilde{\delta}(q_0, y)$.

Πρόταση

Η R_M είναι **δεξιά αναλλοίωτη** *(right invariant), ως προς την παράθεση, δηλαδή για κάθε $x, \, y \in \Sigma^*$ με $x \, R_M \, y$ έχουμε ότι για κάθε $z \in \Sigma^*$ ισχύει $xz \, R_M \, yz$.*

Απόδειξη.

$\delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz)$. □

Το θεώρημα Myhill-Nerode III

Θεώρημα (Myhill-Nerode)

Τα παρακάτω είναι ισοδύναμα για μία γλώσσα L:

(i) *η L είναι κανονική*

(ii) *η L είναι ένωση κάποιων από τις κλάσεις ισοδυναμίας μίας δεξιά αναλλοίωτης σχέσης ισοδυναμίας πεπερασμένου δείκτη*

(iii) *η σχέση R_L είναι πεπερασμένου δείκτη.*

Απόδειξη: Θα δείξουμε τις εξής κατευθύνσεις: (i) $\rightarrow$ (ii) $\rightarrow$ (iii) $\rightarrow$ (i).

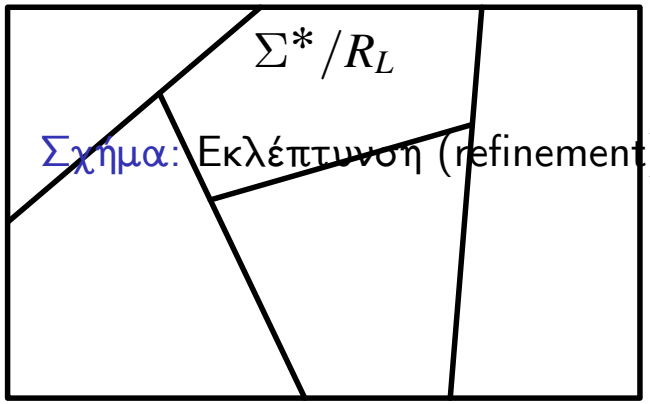
(i) $\implies$ (ii). Έστω DFA $M = (Q, \Sigma, \delta, q_0, F)$ που αποδέχεται την L. Όπως είδαμε, η R_M είναι δεξιά αναλλοίωτη. Οι κλάσεις ισοδυναμίας της R_M είναι όσες και οι καταστάσεις του αυτομάτου M, δηλαδή η R_M είναι επιπλέον πεπερασμένου δείκτη. Προφανώς, η γλώσσα που γίνεται αποδεκτή από το αυτόματο M είναι ένωση των κλάσεων ισοδυναμίας που αντιστοιχούν σε τελικές καταστάσεις του αυτομάτου.

Το θεώρημα Myhill-Nerode IV

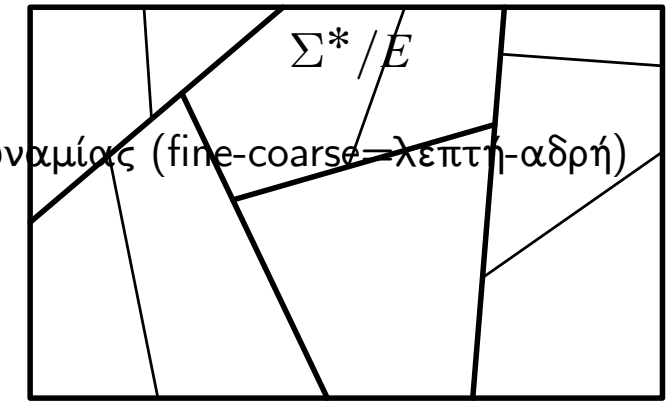
(ii) $\Longrightarrow$ (iii). Έστω ισοδυναμία E που ικανοποιεί την ιδιότητα που δίνεται στο (ii). Θα δείξουμε την συνεπαγωγή: Αν για $x, y \in \Sigma^*$ ισχύει $x\ E\ y$, τότε $x\ R_L\ y$. Πράγματι έστω $x\ E\ y$. Αφού η E είναι δεξιά αναλλοίωτη, έχουμε ότι για κάθε $z \in \Sigma^*$ ισχύει $xz\ E\ yz$, δηλαδή είτε αμφότερα τα xz, yz ανήκουν στην γλώσσα L, είτε κανένα από τα δύο δεν ανήκει στην γλώσσα L, δηλαδή (από τον ορισμό της R_L) $x\ R_L\ y$.

Επομένως, κάθε κλάση ισοδυναμίας της E περιέχεται σε μία κλάση ισοδυναμίας της R_L. Αυτό σημαίνει ότι η σχέση E αποτελεί *εκλέπτυνση* (refinement, βλέπε σχήμα) της σχέσης R_L και αφού η E είναι πεπερασμένου δείκτη, δεν μπορεί παρά και η R_L να είναι πεπερασμένου δείκτη.

Το θεώρημα Myhill-Nerode V



Σχήμα: Εκλέπτυνση (refinement) σχέσης ισοδυναμίας (fine-coarse=λεπτή-αδρή)



Το θεώρημα Myhill-Nerode VI

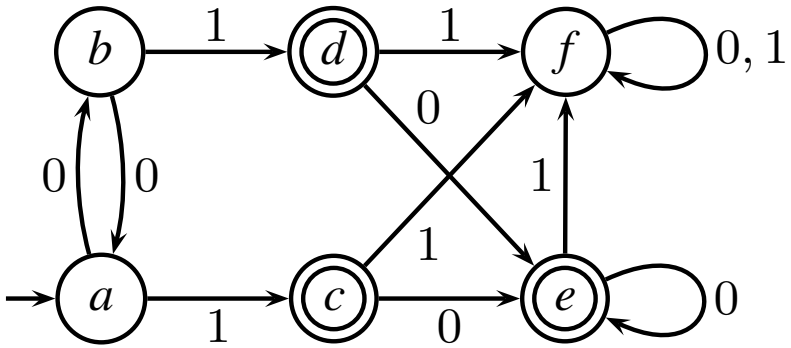
(iii) $\Longrightarrow$ (i). Αποδεικνύουμε πρώτα τον εξής ισχυρισμό: Η R_L είναι δεξιά αναλλοίωτη. Πράγματι, αν $x\ R_L\ y$ τότε για κάθε $u \in \Sigma^*$ έχουμε $xu\ R_L\ yu$ αφού από τον ορισμό της R_L για κάθε $v \in \Sigma^*$ έχουμε $xuv \in L \iff yuv \in L$ (αρκεί να θέσουμε $z = uv$).

Τελικά, θα κατασκευάσουμε αυτόματο $M = (Q, \Sigma, \delta, q_0, F)$ που αποδέχεται την γλώσσα L, δεδομένης της σχέσης R_L. Την κλάση ισοδυναμίας στην R_L του τυχόντος στοιχείου $x \in \Sigma^*$ συμβολίζουμε με $[x]$. Το σύνολο των καταστάσεων Q είναι οι κλάσεις ισοδυναμίας της R_L, δηλαδή $Q = \{[x] \mid x \in \Sigma^*\}$. Αρχική κατάσταση είναι η $[\varepsilon]$ (η κλάση ισοδυναμίας στην οποία ανήκει η κενή συμβολοσειρά). Το σύνολο των τελικών καταστάσεων είναι $F = \{[x] \mid x \in L\}$. Η συνάρτηση μετάβασης ορίζεται $\delta([x], a) = [xa]$ (ο ορισμός δεν εξαρτάται από το x και είναι συνεπής δεδομένου ότι η R_L είναι δεξιά αναλλοίωτη). Το αυτόματο αποδέχεται την L αφού $\delta(q_0, x) = [x]$ κι επομένως $x \in L(M)$ αν και μόνον αν $[x] \in F$. □

Ελαχιστοποιώντας πεπερασμένα αυτόματα Ι

Myhill-Nerode $\Rightarrow$ μοναδικό DFA ελάχιστων καταστάσεων.

Παράδειγμα:
Έστω το αυτόματο M που φαίνεται στο σχήμα, το οποίο αποδέχεται την γλώσσα $L = 0^*10^*$.



Ελαχιστοποιώντας πεπερασμένα αυτόματα II

Στην $L(M)$ υπάρχουν 6 κλάσεις ισοδυναμίας, οι ακόλουθες:

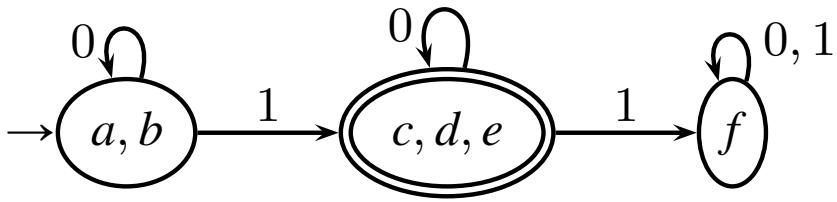
$$C_a = (00)^*, \quad C_b = (00)^*0, \quad C_c = (00)^*1,$$
$$C_d = (00)^*01, \quad C_e = 0^*100^*, \quad C_f = 0^*10^*1(0+1)^*,$$

οπότε και $L = C_c \cup C_d \cup C_e$.

Τα C_a, C_b είναι το C_1 (χωρίς 1) $= 0^*$. Τα C_c, C_d, C_e είναι το C_2 (με ένα 1) $= 0^*10^*$ και το C_f είναι το C_3 (περισσότερα του ενός 1) $= 0^*10^*1(0+1)^*$.

Ελαχιστοποιώντας πεπερασμένα αυτόματα III

Το ελάχιστο αυτόματο φαίνεται στο παρακάτω σχήμα.



Η ίδια μέθοδος συστηματικά με πίνακα:

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

Ελαχιστοποιώντας πεπερασμένα αυτόματα IV

Μέθοδος για το δεύτερο:

- 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.

Στο πίνακα οι δείκτες 1 και 2 του X δείχνουν σε ποια επανάληψη εγγράφουμε το X.

Ελαχιστοποιώντας πεπερασμένα αυτόματα V

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	ⓒ	ⓓ	ⓔ

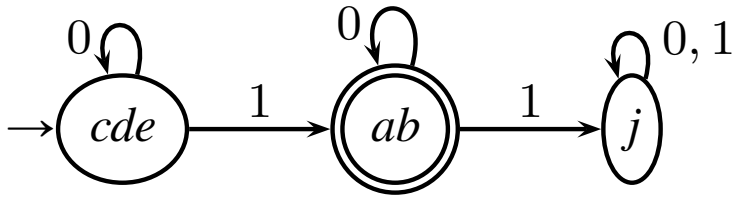
Τελικά οι ισοδύναμες καταστάσεις είναι $a \equiv b$, $c \equiv d \equiv e$.

Ελαχιστοποιώντας πεπερασμένα αυτόματα Ι

Μέθοδος Beckmann

Άλλη μέθοδος για την κατασκευή ενός ελάχιστου DFA είναι η **μέθοδος Beckmann**.

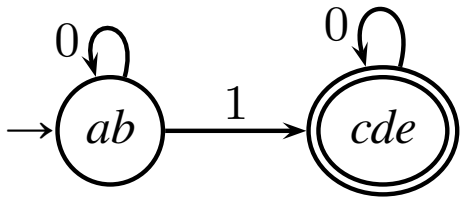
1. Κατασκεύασε το καθρέφτισμα του FA: δηλαδή ανάστρεψε τη φορά των τόξων. Αντάλλαξε τελικές με αρχικές καταστάσεις.
2. Κατασκεύασε DFA ισοδύναμο με το προκύπτον.



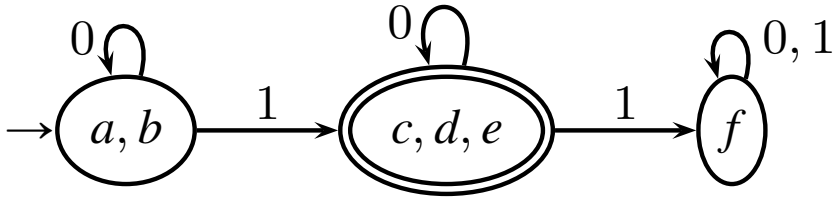
Ελαχιστοποιώντας πεπερασμένα αυτόματα II

Μέθοδος Beckmann

3. Ξανακατασκεύασε το καθρέφτισμα.



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



Περιεχόμενα

1 Πεπερασμένα Αυτόματα και Κανονικά Σύνολα

- Παραλλαγές, επεκτάσεις και εφαρμογές FA/REGEXP
- Ιδιότητες κανονικών συνόλων
- Αλγεβρική περιγραφή κανονικών συνόλων. Ελαχιστοποίηση DFA

2 Τυπικές Γλώσσες

- Τυπικές Γραμματικές
- Απλοποίηση c.f. γραμματικών
- Αυτόματα στοίβας (pushdown automata)
- Ιδιότητες c.f. γλωσσών

3 Μοντέλα Υπολογισμού

- Ιστορία - Εισαγωγή
- LOOP: Μια απλή γλώσσα προγραμματισμού
- LOOP-υπολογίσιμες και πρωταρχικές αναδρομικές συναρτήσεις

Τυπικές Γραμματικές Ι

Μια **τυπική γραμματική (formal grammar)** $G = (V, T, P, S)$ αποτελείται από:

- ένα πεπερασμένο αλφάβητο V από **μη τερματικά σύμβολα (non terminals)** ή **μεταβλητές (variables)**,
- ένα πεπερασμένο αλφάβητο T από **τερματικά σύμβολα (terminals)** ή **σταθερές (constants)**, τ.ώ. $V \cap T = \varnothing$,
- ένα πεπερασμένο σύνολο P από **κανόνες παραγωγής (production rules)** ή απλούστερα **παραγωγές (productions)**, 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 (start symbol)** ή **αξίωμα** $S \in V$.

Τυπικές Γραμματικές ΙΙ

Θα χρησιμοποιήσουμε την εξής σύμβαση για τη χρήση γραμμάτων:

$$a, b, c, d, \ldots \in T$$
$$A, B, C, D, \ldots \in V$$
$$z, y, x, w, v, u, \ldots \in T^*$$
$$\alpha, \beta, \gamma, \delta, \ldots \in (V \cup T)^*$$

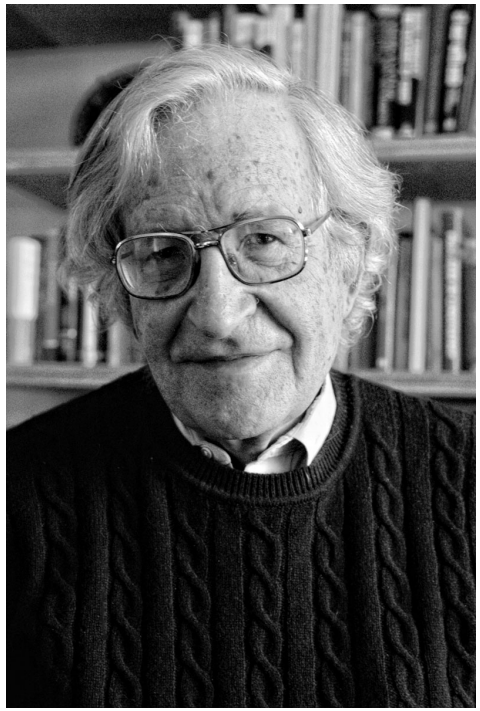
- Θα γράφουμε $\alpha \to \beta|\gamma|\delta$ ως ένα κανόνα στο P αντί για τους τρεις κανόνες $\alpha \to \beta$, $\alpha \to \gamma$, $\alpha \to \delta$ στο P .
- Για μια γραμματική G λέμε ότι το $\gamma_1\beta\gamma_2$ **προκύπτει (is derived from)** από το $\gamma_1\alpha\gamma_2$ και γράφουμε $\gamma_1\alpha\gamma_2 \underset{G}{\Rightarrow} \gamma_1\beta\gamma_2$, αν ο $\alpha \to \beta$ είναι κανόνας παραγωγής (δηλαδή $(\alpha, \beta) \in P$).

Τυπικές Γραμματικές ΙΙΙ

- Συμβολίζουμε με $\overset{*}{\Rightarrow}$ το ανακλαστικό, μεταβατικό κλείσιμο του $\Rightarrow$, δηλαδή $\alpha \overset{*}{\Rightarrow} \beta$ σημαίνει ότι υπάρχει μια ακολουθία παραγωγών (derivation): $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \alpha_k \Rightarrow \beta$.
- Μια συμβολοκολουθία $\alpha \in (V \cup T)^*$ ονομάζεται **προτασιακή μορφή (sentential form)** αν ισχύει $S \overset{*}{\Rightarrow} \alpha$.
- Ως **γλώσσα που παράγεται από τη γραμματική** G ορίζουμε την $L(G) := \{w \in T^* | S \overset{*}{\Rightarrow} w\}$.
- Δύο γραμματικές G_1 , G_2 θα ονομάζονται **ισοδύναμες** αν $L(G_1) = L(G_2)$.

Ιεραρχία Chomsky I

Ο Noam Chomsky (1956) ταξινόμησε τις τυπικές γραμματικές σε μια ιεραρχία σύμφωνα με τη μορφή των κανόνων παραγωγής τους.



Noam Chomsky

Ιεραρχία Chomsky II

τύπου 0: *γενικές* γραμματικές (general, phrase structure, semi-Thue). Μορφή:
$P \subseteq \{\alpha \to \beta \mid \alpha, \beta \in (V \cup T)^* \wedge \alpha \neq \varepsilon\}$

τύπου 1: γραμματικές **με συμφραζόμενα** ή μονοτονικές (context sensitive, monotonic). Μορφή:
$P \subseteq (\{\alpha \to \beta \mid \alpha, \beta \in (V \cup T)^* \wedge |\alpha| \leqslant |\beta| \wedge a \neq \varepsilon\} \cup \{S \to \varepsilon\})$

τύπου 2: γραμματικές **χωρίς συμφραζόμενα** (context free). Μορφή:
$P \subseteq \{A \to \alpha \mid A \in V, \alpha \in (V \cup T)^*\}$

τύπου 3: **κανονικές** γραμματικές (regular). Μορφή:

1. **δεξιογραμμική**:
   $P \subseteq (\{A \to w \mid A \in V, w \in T^*\} \cup \{A \to wB \mid A, B \in V, w \in T^*\})$
2. ή **αριστερογραμμική**:
   $P \subseteq (\{A \to w \mid A \in V, w \in T^*\} \cup \{A \to Bw \mid A, B \in V, w \in T^*\})$

Η ιεράρχηση αυτή είναι γνήσια, δηλαδή ισχύει

$$\text{τύπου } 3 \subset \text{τύπου } 2 \subset \text{τύπου } 1 \subset \text{τύπου } 0$$

Κανονικές γραμματικές Ι

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 $w \in T^*$.

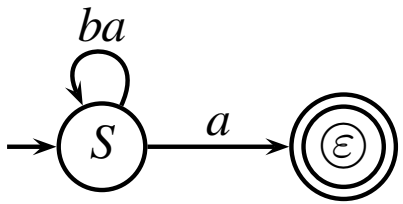
Παράδειγμα

$(ba)^*a$
δεξιογραμμικά: $S \to baS \mid a$
αριστερογραμμικά: $S \to Ra,\ R \to Rba \mid \varepsilon$

Κανονικές γραμματικές Ι

Από γραμματική σε αυτόματο

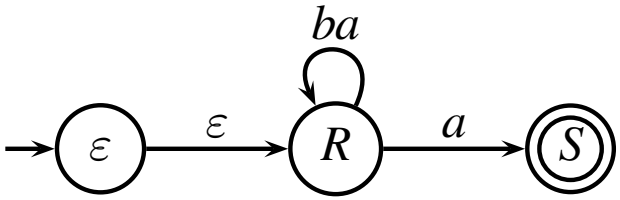
- Κατασκευάζουμε F.A. που αποδέχεται την κανονική γλώσσα που παράγεται από τη δεξιογραμμική γραμματική ($S \rightarrow baS \mid a$):



Κανονικές γραμματικές II

Από γραμματική σε αυτόματο

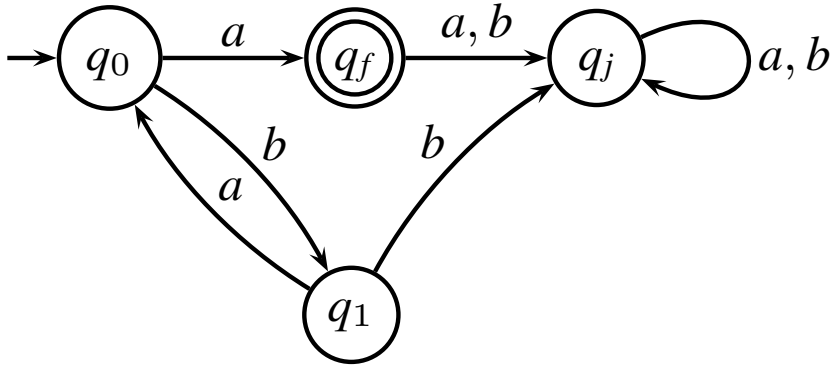
- Κατασκευάζουμε F.A. που αποδέχεται την κανονική γλώσσα που παράγεται από την αριστερογραμμική γραμματική ($S \to Ra$, $R \to Rba \mid \varepsilon$):



Κανονικές γραμματικές Ι

Από αυτόματο σε γραμματική

- Κατασκευή δεξιογραμμικής γραμματικής από αυτόματο.
  Έστω M:



$$\begin{cases} q_0 \to a \mid bq_1 \mid aq_f & (S := q_0) \\ q_1 \to aq_0 \mid bq_j \\ q_f \to aq_j \mid bq_j \\ q_j \to aq_j \mid bq_j \end{cases}$$

Κανονικές γραμματικές II

Από αυτόματο σε γραμματική

Απαλοιφή συμβόλων που δεν παράγουν τίποτα (non yielding) q_j, q_f και παραγωγών που σχετίζονται με αυτά:

$$\begin{cases} q_0 \to a \mid bq_1 & (S := q_0) \\ q_1 \to aq_0 \end{cases}$$

- Κατασκευή αριστερογραμμικής γραμματικής (αντιστροφή βελών).

$$\begin{cases} q_f \to q_0 a & (S := q_f) \\ q_0 \to q_1 a \mid \varepsilon \\ q_1 \to q_0 b \\ q_j \to q_j b \mid q_j a \mid q_f b \mid q_f a \mid q_1 b \end{cases}$$

Απαλοιφή συμβόλων στα οποία δεν φθάνουμε ποτέ (unreachable): q_j, καθώς και παραγωγών που σχετίζονται με αυτά:

$$\begin{cases} q_f \to q_0 a & (S := q_f) \\ q_0 \to q_1 a \mid \varepsilon \\ q_1 \to q_0 b \end{cases}$$

Κανονικές γραμματικές Ι

Ισοδυναμία με αυτόματο

Θεώρημα

Τα ακόλουθα είναι ισοδύναμα:

1. $L = L(G_1)$, *όπου* G_1 *είναι δεξιογραμμική γραμματική.*
2. $L = L(G_2)$, *όπου* G_2 *είναι αριστερογραμμική γραμματική.*
3. $L = L(M)$, *όπου* M *είναι F.A.*

Context-free grammars I

Παράδειγμα (1)

Έστω η γραμματική

$$G_1\colon V = \{S\},\ T = \{a, b\},\ P = \{S \to \varepsilon | aSb\}.$$

Μια δυνατή ακολουθία παραγωγών είναι η:
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
Η γλώσσα που παράγεται από την G_1 είναι η $L(G_1) := \{a^n b^n | n \in \mathbb{N}\}$

Παράδειγμα (2)

$G_2 = (V, T, P, S)$ όπου $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *\}$, $V = \{S\}$ και το P περιέχει τους κανόνες:

$$S \to S + S$$
$$S \to S * S$$
$$S \to 0|1|2|3|4|5|6|7|8|9$$

Context-free grammars II

Παράδειγμα (3)

$G_3 = (V, T, P, S)$, $V = \{S, A, B\}$, $T = \{a, b\}$ και το P περιέχει τους κανόνες:

$$S \to aB|bA$$
$$A \to a|aS|bAA$$
$$B \to b|bS|aBB$$

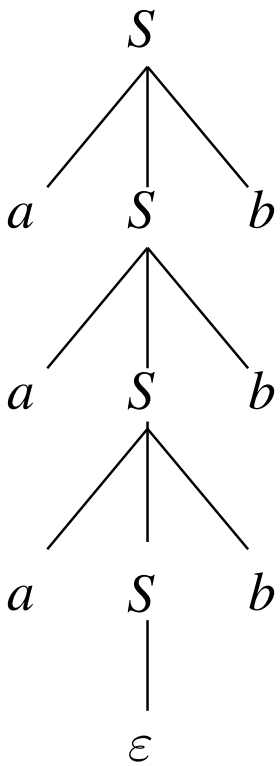
Μια δυνατή ακολουθία παραγωγών της πιο πάνω γραμματικής είναι:
$S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba$
Αν και δεν είναι προφανές, $L(G_3) = \{w \in T^+ \mid w$ έχει ίσο αριθμό a και $b\}$

Συντακτικά δένδρα Ι

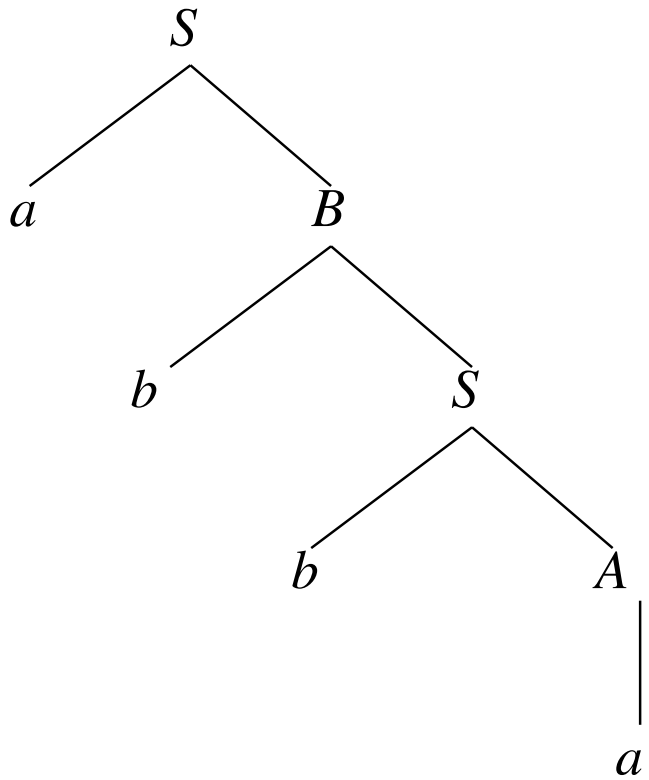
Θεώρημα

Έστω $G = (V, T, P, S)$ μια c.f. γραμματική. Τότε $S \overset{*}{\Rightarrow} w$ αν υπάρχει συντακτικό δένδρο με φύλλωμα το w.

Συντακτικά δένδρα II



(1)

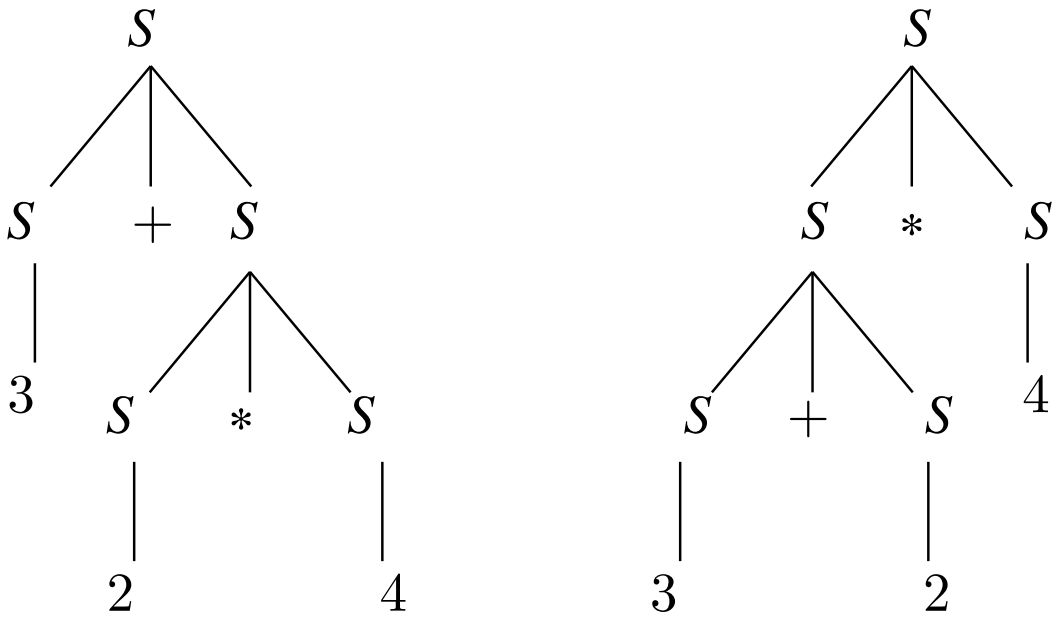


(3)

Συντακτικά δένδρα III

Ορισμός

Μια γραμματική ονομάζεται **διφορούμενη (ambiguous)** αν υπάρχουν δύο συντακτικά δένδρα που να έχουν ως φύλλωμα το ίδιο $w \in L(G)$.



Παράδειγμα διφορούμενης γραμματικής. Η συμβολοσειρά $3 + 2 * 4$ αποτελεί φύλλωμα δύο πιθανών συντακτικών δένδρων της γραμματικής G_2 του παραδείγματος (2).

Άχρηστα σύμβολα (unreachable + non-yielding)

Ορισμός

Ένα σύμβολο X λέγεται **reachable** ανν υπάρχει μια παραγωγή της μορφής $S \overset{*}{\Rightarrow} \alpha X \beta$ για κάποια $\alpha, \beta \in (V \cup T)^*$. Διαφορετικά λέγεται **unreachable**.

Ορισμός

Ένα σύμβολο X λέγεται **yielding** ανν υπάρχει μια παραγωγή της μορφής $X \overset{*}{\Rightarrow} w$ για κάποιο $w \in T^*$. Διαφορετικά λέγεται **non-yielding**.

Θεώρημα

Κάθε μη κενή c.f. γλώσσα παράγεται από μια c.f. γραμματική χωρίς άχρηστα σύμβολα.

Απόδειξη.

Απαλείφουμε τα non-reachable και non-yielding σύμβολα (η διαδικασία αυτή μπορεί να γίνει μηχανιστικά). □

ε-productions I

Ορισμός

Παραγωγές τις μορφής $A \to \varepsilon$ ονομάζονται **ε-productions**.

Ορισμός

Μια μεταβλητή A λέγεται **nullable** αν υπάρχει μια ακολουθία παραγωγών τέτοια ώστε $A \overset{*}{\Rightarrow} \varepsilon$.

Θεώρημα

Δεδομένης μιας c.f. γραμματικής $G = (V, T, P, S)$ μπορούμε να κατασκευάσουμε μια c.f. γραμματική G' χωρίς άχρηστα σύμβολα και ε-productions τέτοια ώστε $L(G') = L(G) \setminus \{\varepsilon\}$.

ε-productions II

Παράδειγμα

'Εστω η γραμματική

$$S \to aAbAc$$
$$A \to aaBa|\varepsilon$$
$$B \to bbb$$

τότε παίρνω

$$S \to aAbAc|abAc|aAbc|abc$$
$$A \to aaBa$$
$$B \to bbb$$

unit productions

Ορισμός

Μια παραγωγή της μορφής $A \to B$ καλείται **unit production**.

Θεώρημα

Κάθε c.f. γλώσσα που δεν περιέχει το ε παράγεται από μία γραμματική χωρίς άχρηστα σύμβολα, ε-productions και unit productions.

Παράδειγμα

'Εστω οι παραγωγές:

$$A \to B|abb$$
$$B \to cab$$

τότε παίρνω

$$A \to cab|abb$$
$$B \to cab$$

Απλοποίηση c.f. γραμματικών

Συνοψίζοντας, τα βήματα που πρέπει να ακολουθήσει κανείς για να απλοποιήσει μια c.f. γραμματική είναι:

1. Απαλοιφή ε-productions.
2. Απαλοιφή non-yielding συμβόλων.
3. Απαλοιφή unreachable συμβόλων.
4. Απαλοιφή unit productions.

Chomsky Normal Form I

Η ιδέα της **κανονικής μορφής Chomsky** είναι να περιορίσουμε το μέγεθος του δεξιού μέλους κάθε παραγωγής, έτσι ώστε μια μεταβλητή να αντικαθίσταται από δύο μόνο άλλες μεταβλητές το πολύ σε κάθε βήμα.

Παράδειγμα

Η γλώσσα

$$S \to SaS|b$$

μετατρέπεται αρχικά σε

$$S \to SC_aS|b,\ C_a \to a$$

και, εν τέλει, σε

$$S \to SD_1|b,\ C_a \to a,\ D_1 \to C_aS$$

Chomsky Normal Form II

Θεώρημα (Κανονική μορφή 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.

Παράδειγμα: Έστω η γραμματική $G_1 = (V, T, P, S)$, $V = \{S, A, B\}$, $T = \{a, b\}$ και P:

$$S \to aB|bA$$
$$A \to a|aS|bAA$$
$$B \to b|bS|aBB$$

σύμφωνα με το πιο πάνω θεώρημα, η $G_2 = (V', T, P', S)$ θα έχει $V' = V \cup \{C_a, C_b\}$ και P':

$$S \to C_aB|C_bA$$
$$A \to a|C_aS|C_bAA$$
$$B \to b|C_bS|C_aBB$$
$$C_a \to a$$
$$C_b \to b$$

Chomsky Normal Form III

τέλος μετατρέπουμε την G_2 στην $G_3 = (V'', T, P'', S)$, η οποία είναι σε CNF, και έχει $V'' = V' \cup \{D_1, D_2\}$ και P'':

$$
\begin{aligned}
S &\to C_a B | C_b A \\
A &\to a | C_a S | C_b D_1 \\
D_1 &\to AA \\
B &\to b | C_b S | C_a D_2 \\
D_2 &\to BB \\
C_a &\to a \\
C_b &\to b
\end{aligned}
$$

Greibach Normal Form I

Η ιδέα πίσω από την **κανονική μορφή Greibach (GNF)** είναι να μετατρέψουμε τους κανόνες παραγωγής με τέτοιο τρόπο ώστε το αποτέλεσμα μιας αντικατάστασης μιας μεταβλητής, σύμφωνα με τους κανόνες παραγωγής, να έχει πάντοτε ως πρώτο σύμβολο ένα τερματικό.

Παράδειγμα

Μια γραμματική με κανόνες παραγωγής:

$$S \to Sac|Sc|da$$

μετατρέπεται σε

$$S \to da$$
$$S \to daB$$
$$B \to ac|c$$
$$B \to acB|cB$$

Greibach Normal Form II

Λήμμα

Έστω μια c.f. γραμματική G. Έστω επιπλέον $A \to \alpha_1 B \alpha_2$ μια παραγωγή στο P και $B \to \beta_1 | \beta_2 | \dots | \beta_r$ όλες οι B-παραγωγές. Ονομάζουμε $G_1 = (V, T, P_1, S)$ την γραμματική που προκύπτει από την G αν αφαιρέσουμε την παραγωγή $A \to \alpha_1 B \alpha_2$ και στη θέση της προσθέσουμε τις παραγωγές $A \to \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \dots | \alpha_1 \beta_r \alpha_2$. Τότε ισχύει $L(G) = L(G_1)$.

Λήμμα

Έστω μια c.f. γραμματική $G = (V, T, P, S)$. Έστω επίσης $A \to A\alpha_1 | A\alpha_2 | \dots | A\alpha_r$ όλες οι A-παραγωγές στις οποίες το A είναι το αριστερότερο σύμβολο στο δεξί μέλος μιας παραγωγής (left recursion) και $A \to \beta_1 | \beta_2 | \dots | \beta_s$ οι υπόλοιπες A-παραγωγές. Ονομάζουμε $G_1 = (V \cup \{B\}, T, P_1, S)$ την c.f. γραμματική που προκύπτει από την G προσθέτοντας την μεταβλητή B στο V και αντικαθιστώντας όλες τις A-παραγωγές με τις παραγωγές:

$$\begin{cases} A \to \beta_1 | \beta_2 | \dots | \beta_s | \beta_1 B | \beta_2 B | \dots | \beta_s B \\ B \to \alpha_1 | \alpha_2 | \dots | \alpha_r | \alpha_1 B | \alpha_2 B | \dots | \alpha_r B \end{cases}$$

Τότε $L(G_1) = L(G)$.

Greibach Normal Form III

Θεώρημα (Κανονική μορφή 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 $\alpha \in V^*$, $a \in T$.

Απόδειξη:

Βήμα πρώτο. Μετατρέπουμε επαγωγικά τις παραγωγές έτσι ώστε αν $A_i \to A_j\gamma$, $\gamma \in V^*$ είναι μια παραγωγή, τότε $j > i$.

Έστω ότι όλες οι A_i-παραγωγές με $1 \leqslant i < k$, για κάποιο k, έχουν τροποποιηθεί έτσι ώστε αν $A_i \to A_j\gamma$ είναι μια παραγωγή, τότε $j > i$ (επαγωγική υπόθεση). Θα τροποποιήσουμε τώρα τις A_k παραγωγές έτσι ώστε να έχουν και αυτές την επιθυμητή ιδιότητα. Αν $A_k \to A_j\gamma$ είναι μια παραγωγή με $j < k$ και $A_j \to \beta_1|\beta_2|\ldots|\beta_r$, δημιουργούμε ένα νέο σύνολο παραγωγών αντικαθιστώντας το A_j με το δεξί μέλος κάθε A_j παραγωγής, σύμφωνα με το πρώτο λήμμα. Έτσι λαμβάνουμε τις παραγωγές

$$A_k \to \beta_1\gamma|\beta_2\gamma|\ldots|\beta_r\gamma$$

Τώρα μεγάλωσε ο δείκτης της πρώτης μεταβλητής των A_k-παραγωγών.

Greibach Normal Form IV

Επαναλαμβάνοντας αυτή τη διαδικασία το πολύ $k-1$ φορές, καταλήγουμε σε παραγωγές της μορφής $A_k \to A_\ell \gamma$, $\ell \geqslant k$. Οι παραγωγές με $\ell = k$ (left recursion) αντικαθίστανται στη συνέχεια σύμφωνα με το δεύτερο λήμμα, εισάγοντας μια νέα μεταβλητή B_k. Προσθέτουμε λοιπόν κανόνες παραγωγής της μορφής $B_k \to \gamma$, $B_k \to \gamma B_k$ και $A_k \to \beta B_k$, για κάθε παραγωγή της μορφής $A_k \to \beta$ που το β δεν ξεκινά με A_k.

Επαναλαμβάνοντας την πιο πάνω διαδικασία για κάθε μεταβλητή του V, καταλήγουμε να έχουμε μόνο παραγωγές με μορφή μια από τις παρακάτω:

1. $A_i \to A_j \gamma$, όπου $j > i$,
2. $A_i \to a\gamma$, όπου $a \in T$,
3. $B_i \to \gamma$, όπου και εδώ αλλά και πριν $\gamma \in (V \cup \{B_1, B_2, \ldots, B_{i-1}\})^*$

Greibach Normal Form V

Βήμα δεύτερο. Παρατηρούμε ότι το αριστερότερο σύμβολο στη δεξιά πλευρά κάθε A_m-παραγωγής πρέπει να είναι τερματικό, καθώς η A_m είναι αριθμημένη με τον μεγαλύτερο δείκτη. Το αριστερότερο σύμβολο στη δεξιά πλευρά κάθε A_{m-1}-παραγωγής θα πρέπει να είναι είτε A_m είτε κάποιο τερματικό σύμβολο. Όπου είναι A_m, μπορούμε να κατασκευάζουμε νέες παραγωγές αντικαθιστώντας το A_m σύμφωνα με το πρώτο λήμμα. Αυτές οι παραγωγές θα έχουν δεξιά μέλη που θα ξεκινούν με τερματικά σύμβολα. Με τον ίδιο τρόπο, προχωρούμε στη συνέχεια στις παραγωγές των A_{m-2}, A_{m-3}, ...A_2, A_1, μέχρι το δεξί μέλος κάθε παραγωγής να ξεκινά με κάποιο τερματικό σύμβολο.

Βήμα τρίτο Τέλος, ελέγχουμε τις παραγωγές των νέων μεταβλητών B_1, B_2,..., B_m. Αφού ξεκινήσαμε με μια γραμματική σε CNF, είναι εύκολο να δείξουμε με επαγωγή στον αριθμό των εφαρμογών των λημμάτων ότι το δεξί μέλος κάθε B_i-παραγωγής θα ξεκινάει είτε με κάποιο τερματικό είτε με κάποιο A_j. Οπότε, άλλη μία εφαρμογή του πρώτου λήμματος, για να μετατραπούν οι παραγωγές της μορφής $B_i \rightarrow A_j\gamma$, ολοκληρώνει την κατασκευή.

□

Greibach Normal Form VI

Παράδειγμα:
Έστω η γραμματική $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$, όπου το P περιέχει τους εξής κανόνες παραγωγής:

$$A_1 \to A_2A_3$$
$$A_2 \to A_3A_1|b$$
$$A_3 \to A_1A_2|a$$

σε Chomsky Normal Form.

Βήμα πρώτο. Οι παραγωγές για A_1, A_2 έχουν την επιθυμητή μορφή, ενώ η $A_3 \to A_1A_2$ όχι, καθώς ξεκινάει με μεταβλητή μικρότερου δείκτη. Εφαρμόζοντας το πρώτο λήμμα παίρνουμε

$$A_3 \to A_2A_3A_2|a$$

που επίσης δεν είναι στην επιθυμητή μορφή, άρα ξαναεφαρμόζουμε το πρώτο λήμμα και παίρνουμε

$$A_3 \to A_3A_1A_3A_2|bA_3A_2|a$$

Greibach Normal Form VII

Τώρα έχουμε $A_3 \to A_3\gamma$ (left recursion), άρα εφαρμόζουμε το δεύτερο λήμμα. Εισάγουμε μια καινούρια μεταβλητή B_3 και οι κανόνες παραγωγής γίνονται:

$$A_3 \to bA_3A_2B_3|aB_3|bA_3A_2|a$$
$$B_3 \to A_1A_3A_2|A_1A_3A_2B_3$$

Βήμα δεύτερο. Τώρα όλες οι A_3-παραγωγές ξεκινούν με τερματικά. Για να συμβεί το ίδιο και στις υπόλοιπες A_i μεταβλητές, εφαρμόζουμε επαναληπτικά το πρώτο λήμμα. Η διόρθωση των A_2-παραγωγών δίνει:

$$A_2 \to bA_3A_2B_3A_1 \qquad A_2 \to bA_3A_2A_1$$
$$A_2 \to aB_3A_1 \qquad A_2 \to aA_1$$
$$A_2 \to b$$

Greibach Normal Form VIII

Ενώ η διόρθωση των A_1-παραγωγών:

$$A_1 \to bA_3A_2B_3A_1A_3$$
$$A_1 \to aB_3A_1A_3$$
$$A_1 \to bA_3$$

$$A_1 \to bA_3A_2A_1A_3$$
$$A_1 \to aA_1A_3$$

Greibach Normal Form IX

Βήμα τρίτο. Οι B_3 παραγωγές μετατρέπονται με εφαρμογή του πρώτου λήμματος σε κανονική μορφή, προσθέτοντας έτσι άλλες 10 παραγωγές. Το τελικό σύνολο από παραγωγές είναι το εξής:

$$A_3 \to bA_3A_2B_3|bA_3A_2|aB_3|a$$
$$A_2 \to bA_3A_2B_3A_1|bA_3A_2A_1|aB_3A_1|aA_1|b$$
$$A_1 \to bA_3A_2B_3A_1A_3|bA_3A_2A_1A_3|aB_3A_1A_3|aA_1A_3|bA_3$$
$$B_3 \to bA_3A_2B_3A_1A_3A_3A_2B_3|bA_3A_2B_3A_1A_3A_3A_2$$
$$B_3 \to aB_3A_1A_3A_3A_2B_3|aB_3A_1A_3A_3A_2$$
$$B_3 \to bA_3A_3A_2B_3|bA_3A_3A_2$$
$$B_3 \to bA_3A_2A_1A_3A_3A_2B_3|bA_3A_2A_1A_3A_3A_2$$
$$B_3 \to aA_1A_3A_3A_2B_3|aA_1A_3A_3A_2$$

Εισαγωγή στα PDA I

Ένα **αυτόματο στοίβας** (push down automaton ή για συντομία PDA) αποτελείται από μία **ταινία εισόδου**, αλλά επιπλέον, σε σχέση με τα FA, έχει και μία **στοίβα** (μη φραγμένη σε μέγεθος μνήμη, αλλά με περιορισμένες δυνατότητες πρόσβασης σε αυτήν).

Η πρόσβαση στην στοίβα γίνεται μόνον στην κορυφή αυτής με τις εξής δύο λειτουργίες:

1. **push**: Τοποθετεί ένα στοιχείο που δίνεται στην κορυφή της στοίβας.
2. **pop**: Αφαιρεί ένα στοιχείο για χρήση από την κορυφή της στοίβας.

Παρατήρηση

Είναι προφανές ότι αν κάποιο στοιχείο βρίσκεται βαθιά μέσα στην στοίβα, δηλαδή αφού έχει μπει στην στοίβα με push, έχουν επίσης μπει με push άλλα στοιχεία πάνω από αυτό, προκειμένου να έχουμε πρόσβαση σε αυτό θα πρέπει να κάνουμε pop σε όλα τα στοιχεία που είναι από πάνω του.

Εισαγωγή στα PDA II

Παράδειγμα:

Θεωρήστε την γλώσσα $L = \{wcw^R \mid w \in (0+1)^*\}$. Για παράδειγμα $110c011 \in L$. Να πώς μπορούμε να αναγνωρίσουμε την παραπάνω γλώσσα με ένα PDA:

1. push(a) στην στοίβα για κάθε 0 που συναντάς στην είσοδο,
   push(b) στην στοίβα για κάθε 1 που συναντάς στην είσοδο,
   μέχρι να συναντήσεις το c
2. διάβασε το c,
   κάνε pop τα στοιχεία της στοίβας και διάβαζε παράλληλα την είσοδο,
   αποδέξου αν υπάρχει συμφωνία των στοιχείων εισόδου με τα στοιχεία της στοίβας
   (a με 0, b με 1).

Εισαγωγή στα PDA III

Η συμβολοσειρά $110c011$ γίνεται αποδεκτή ως εξής:

$$q_0, 110c011, \underline{\left|\right|} \vdash_M q_0, 10c011, \underline{\left|\begin{matrix}\\ b\end{matrix}\right|} \vdash_M q_0, 0c011, \underline{\left|\begin{matrix}b\\ b\end{matrix}\right|} \vdash_M q_0, c011, \underline{\left|\begin{matrix}a\\ b\\ b\end{matrix}\right|}$$
$$\vdash_M q_2, 011, \underline{\left|\begin{matrix}a\\ b\\ b\end{matrix}\right|} \vdash_M q_2, 11, \underline{\left|\begin{matrix}b\\ b\end{matrix}\right|} \vdash_M q_2, 1, \underline{\left|\begin{matrix}\\ b\end{matrix}\right|} \vdash_M q_2, , \underline{\left|\right|}$$

Αντιθέτως, η συμβολοσειρά $01c01$ δεν γίνεται αποδεκτή. Ένας υπολογισμός είναι μία πεπερασμένη ακολουθία από configurations ή *στιγμιαίες περιγραφές*.

Τυπικός ορισμός PDA

Ορισμός

Ένα **αυτόματο στοίβας** (push down automaton ή για συντομία PDA) είναι μία πλειάδα $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, όπου:

- Q: πεπερασμένο σύνολο καταστάσεων,
- Σ: αλφάβητο εισόδου,
- Γ: αλφάβητο στοίβας,
- δ: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \text{Pow}(Q \times \Gamma^*)$ (πεπερασμένα υποσύνολα), η συνάρτηση μετάβασης (επιτρέπονται ε-κινήσεις και μη ντετερμινισμός),
- $q_0 \in Q$: αρχική κατάσταση,
- $Z_0 \in \Gamma$: αρχικό σύμβολο στην στοίβα,
- $F \subseteq Q$: τελικές καταστάσεις.

Υπάρχουν δύο είδη PDA ως προς το αποδέχεσθαι:

1. αποδέξου όταν βρίσκεσαι σε τελική κατάσταση αφού έχεις διαβάσει όλη την ταινία εισόδου, ανεξάρτητα από το τι υπάρχει στην στοίβα, ή,
2. αποδέξου όταν η στοίβα είναι άδεια αφού έχεις διαβάσει όλη την ταινία εισόδου, ανεξάρτητα από την κατάσταση στην οποία ευρίσκεσαι (σύμβαση: $F = \varnothing$).

Ερμηνεία της συνάρτησης μετάβασης δ I

$$\delta(q, a, z) = \{(q_1, \gamma_1), (q_2, \gamma_2), \ldots, (q_m, \gamma_m)\} \quad \text{όπου}$$

- $q, q_1, q_2, \ldots, q_m \in Q$
- $a \in \Sigma \cup \{\varepsilon\}$
- $z \in \Gamma$: αλφάβητο στοίβας,
- $\gamma_1, \gamma_2, \ldots, \gamma_m \in \Gamma^*$

Επίλεξε ένα από τα (q_i, γ_i) μη ντετερμινιστικά:

- η επόμενη κατάσταση είναι η q_i
- κάνε pop το z από την στοίβα και κάνε push όλο το γ_i (το τελευταίο σύμβολο πρώτα). Για παράδειγμα:

$\gamma_i = abc$: $\left|\begin{matrix} z \\ * \end{matrix}\right| \rightarrow \left|\begin{matrix} a \\ b \\ c \\ * \end{matrix}\right|$ $\qquad$ $\gamma_i = \varepsilon$: $\left|\begin{matrix} z \\ * \end{matrix}\right| \rightarrow \left|\begin{matrix} * \end{matrix}\right|$

Ερμηνεία της συνάρτησης μετάβασης δ II

Παράδειγμα

PDA που αποδέχεται την $\{wcw^R \mid w \in (0+1)^*\}$:

$$M = (\{q_0, q_2\}, \{0, 1, c\}, \{a, b, B\}, \delta, q_0, B, \varnothing), \quad \text{όπου } \delta\text{:}$$

$$\delta(q_0, 0, \begin{matrix} B \\ a \\ b \end{matrix}) = \{(q_0, \begin{matrix} B \\ a \\ b \end{matrix} a)\}, \quad \delta(q_0, 1, \begin{matrix} B \\ a \\ b \end{matrix}) = \{(q_0, \begin{matrix} B \\ a \\ b \end{matrix} b)\}, \quad \delta(q_0, c, \begin{matrix} B \\ a \\ b \end{matrix}) = \{(q_2, \begin{matrix} B \\ a \\ b \end{matrix})\}$$

$$\delta(q_2, 0, a) = \{(q_2, \varepsilon)\}, \quad \delta(q_2, 1, b) = \{(q_2, \varepsilon)\}, \quad \delta(q_2, \varepsilon, B) = \{(q_2, \varepsilon)\}$$

(Είναι ντετερμινιστικό!) Ένας υπολογισμός τυπικά:

$$\begin{aligned}(q_0, 110c011, B) &\vdash (q_0, 10c011, bB) \vdash (q_0, 0c011, bbB) \vdash (q_0, c011, abbB) \\ &\vdash (q_2, 011, abbB) \vdash (q_2, 11, bbB) \vdash (q_2, 1, bB) \vdash (q_2, \varepsilon, B) \vdash (q_2, \varepsilon, \varepsilon)\end{aligned}$$

Γλώσσα που αποδέχεται ένα PDA

Ας ορίσουμε τυπικά και την γλώσσα που αποδέχεται ένα PDA:

Ορισμός

1. Γλώσσα που αποδέχεται σε τελική κατάσταση
   $$L_f(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \gamma) \wedge q \in F\}$$
2. Γλώσσα που αποδέχεται με κενή στοίβα
   $$L_e(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \varepsilon)\ \}$$

Ντετερμινισμός και μη

Προκειμένου να γίνει αποδεκτή η $L_1 = \{ww^R \mid w \in (0+1)^*\}$ χωρίς το σημάδι c στην μέση της συμβολοσειράς χρειαζόμαστε απαραιτήτως ένα μη ντετερμινιστικό PDA. Τα μη ντετερμινιστικά PDA είναι γνησίως πιο ισχυρά από τα ντετερμινιστικά.

'Ασκηση. Κατασκευάστε PDA που αποδέχεται την L_1.

Σε ένα ντετερμινιστικό PDA δεν υπάρχουν επιλογές, αλλά επιτρέπουμε ε-κινήσεις. Αν μία ε-κίνηση είναι δυνατή δεν είναι δυνατή καμμία άλλη κίνηση. Τυπικά:

Ορισμός

Ένα PDA είναι ντετερμινιστικό όταν:

1. $|\delta(q, a, z)| \leqslant 1$ για κάθε $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $z \in \Gamma$ και
2. για κάθε $q \in Q$, $a \in \Sigma$, $z \in \Gamma$, αν $\delta(q, \varepsilon, z) \neq \varnothing$, τότε $\delta(q, a, z) = \varnothing$.

Ισοδυναμία PDA με διαφορετικές αποδοχές Ι

Ας θυμηθούμε τις δύο γλώσσες αποδοχής που αντιστοιχούν σε ένα PDA:

- $L_f(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \gamma) \wedge q \in F\}$
- $L_e(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \varepsilon)\}$

Θα χρησιμοποιήσουμε μία επιπλέον γλώσσα αποδοχής:

$$L(M) = \{w \mid (q_0, w, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \varepsilon) \wedge q \in F\}$$

Προφανώς, σε αυτήν το αυτόματο αποδέχεται όταν ταυτόχρονα αφού έχει διαβάσει όλη την είσοδο βρίσκεται σε τελική κατάσταση και επιπλέον η στοίβα είναι άδεια.

Ισοδυναμία PDA με διαφορετικές αποδοχές II

Θεώρημα

Τα παρακάτω είναι ισοδύναμα για μία γλώσσα L:

1. $L = L_f(M_2)$, M_2 *είναι PDA.*
2. $L = L_e(M_1)$, M_1 *είναι PDA.*
3. $L = L(M_b)$, M_b *είναι PDA.*
4. $L = L_f(M_a) = L_e(M_a) = L(M_a)$, M_a *είναι PDA.*
5. *Η L είναι γλώσσα χωρίς συμφραζόμενα (context free).*

Απόδειξη: Θα δείξουμε τις εξής κατευθύνσεις:

$$1 \iff 4, \quad 2 \iff 4, \quad 3 \iff 4, \quad 4 \iff 5$$

Ισοδυναμία PDA με διαφορετικές αποδοχές III

$1 \implies 4$: Κατασκευάζουμε pda M_a που εξομοιώνει το M_2. Το M_a θα πρέπει να αδειάζει την στοίβα όταν το M_2 φτάνει σε τελική κατάσταση (χρησιμοποιούμε την κατάσταση q_e για αυτόν τον λόγο). Ακόμη χρησιμοποιούμε ένα στοιχείο X_0 στον πάτο της στοίβας έτσι ώστε το αυτόματο να μην αποδέχεται σε οποιαδήποτε άλλη περίπτωση. Τέλος, χρησιμοποιούμε μία νέα τελική κατάσταση q_f προκειμένου να έχουμε και τις δύο συνθήκες τερματισμού ταυτόχρονα.

$$M_a = (Q \cup \{q_0', q_e, q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \{q_f\})$$

1. πρώτη μετάβαση:
   $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0 X_0)\}$
2. όλες οι παλιές μεταβάσεις:
   $\delta'(q, a, z) \supseteq \delta(q, a, z)$, για κάθε $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $z \in \Gamma$
3. μεταβάσεις για άδειασμα στοίβας:
   $\delta'(q, a, z) \ni (q_e, \varepsilon)$, για κάθε $q \in F \cup \{q_e\}$, $z \in \Gamma$
4. τελική μετάβαση:
   $\delta'(q, \varepsilon, X_0) = \{(q_f, \varepsilon)\}$, για κάθε $q \in F \cup \{q_e\}$

Ισοδυναμία PDA με διαφορετικές αποδοχές IV

Θα δείξουμε $L_f(M_2) = L_e(M_a) = L_f(M_a) = L(M_a)$. Έστω $x \in L_f(M_2)$, δηλαδή ισχύει: $(q_0, x, Z_0) \overset{*}{\underset{M_2}{\vdash}} (q, \varepsilon, \gamma) \wedge q \in F$. Τότε θα έχουμε

$(q_0', x, X_0) \underset{M_a}{\vdash} (q_0, x, Z_0X_0) \overset{*}{\underset{M_a}{\vdash}} (q, \varepsilon, X_0) \overset{*}{\underset{M_a}{\vdash}} (q_e, \varepsilon, X_0) \underset{M_a}{\vdash} (q_f, \varepsilon, \varepsilon)$ και επομένως το x ανήκει σε κάθε ένα από τα $L_e(M_a)$, $L_f(M_a)$, $L(M_a)$. Παρόμοια δείχνουμε και τις άλλες συμπεριλήψεις. $\Diamond$

$2 \implies 4$: Κατασκευάζουμε pda M_a που εξομοιώνει το M_1. Το M_a θα πρέπει να μπαίνει στην τελική κατάσταση όταν το M_1 αδειάζει την στοίβα του. Χρησιμοποιούμε ένα νέο στοιχείο X_0 στον πάτο της στοίβας έτσι ώστε να έχουμε και τις δύο συνθήκες τερματισμού ταυτόχρονα.

$$M_a = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \{q_f\})$$

1. πρώτη μετάβαση:
   $\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$

Ισοδυναμία PDA με διαφορετικές αποδοχές V

2. όλες οι παλιές μεταβάσεις:
   $\delta'(q, a, z) \supseteq \delta(q, a, z)$, για κάθε $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $z \in \Gamma$
3. τελική μετάβαση:
   $\delta'(q, \varepsilon, X_0) = \{(q_f, \varepsilon)\}$, για κάθε $q \in Q$

Τελικά, όπως παραπάνω δείχνουμε $L_e(M_1) = L_f(M_a) = L_e(M_a) = L(M_a)$. $\Diamond$

$3 \implies 4$: Χρησιμοποιούμε ένα νέο στοιχείο X_0 στον πάτο της στοίβας και μία νέα (μοναδική) τελική κατάσταση q_f για να αποφύγουμε την περίπτωση να ισχύει μόνον μία συνθήκη τερματισμού.

Η κατασκευή είναι παρόμοια με αυτήν του $2 \implies 4$. Η μόνη διαφορά είναι στο 3. (τελική μετάβαση), όπου το "για κάθε $q \in Q$" γίνεται "για κάθε τελική κατάσταση ($q \in F$)". $\Diamond$

Τα $4 \implies 1$, $4 \implies 2$, $4 \implies 3$ είναι προφανή. $\Diamond$

Ισοδυναμία PDA με διαφορετικές αποδοχές VI

$5 \implies 4$: Έστω ότι δίνεται γραμματική χωρίς συμφραζόμενα G τέτοια ώστε $\varepsilon \notin L(G)$ (σε αντίθετη περίπτωση η κατασκευή μας είναι ελαφρά διαφορετική). Υποθέτουμε επίσης ότι δίνεται σε κανονική μορφή Greibach. Κατασκευάζουμε ένα pda M_a που εξομοιώνει αριστερότερες ακολουθίες παραγωγών της G:

$$M_a = (\{q_0\}, T, V, \delta, q_0, S, \{q_0\})$$

και για την συνάρτηση μετάβασης:

$$(q_0, \gamma) \in \delta(q_0, a, A) \quad \textbf{αννν} \quad (A \to a\gamma) \in P \qquad \text{(μη ντετ.)}$$

Τότε για $x \in T^*$, $\alpha \in V^*$ ισχύει:

$$S \overset{*}{\Rightarrow} x\alpha \quad \textbf{αννν} \quad (q_0, x, S) \overset{*}{\underset{M_a}{\vdash}} (q_0, \varepsilon, \alpha)$$

Η κατεύθυνση "$\Leftarrow$" αποδεικνύεται με επαγωγή στο πλήθος των υπολογιστικών βημάτων και η "$\Rightarrow$" με επαγωγή στο πλήθος των βημάτων της ακολουθίας παραγωγής. Επομένως: $S \overset{*}{\Rightarrow} x$ **αννν** $(q_0, x, S) \overset{*}{\underset{M_a}{\vdash}} (q_0, \varepsilon, \varepsilon)$, δηλαδή $x \in L(G)$ **αννν** $x \in L(M_a)$. $\Diamond$

Ισοδυναμία PDA με διαφορετικές αποδοχές VII

$4 \implies 5$: Έστω pda $M_a = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, q_f)$. Θα δείξουμε ότι για την $L = L(M_a)$ υπάρχει γραμματική χωρίς συμφραζόμενα. Μάλιστα, την κατασκευάζουμε ως εξής:

$$G = (V, T, P, S), \text{ με } T = \Sigma,\ V = \{S\} \cup (Q \times \Gamma \times Q)$$

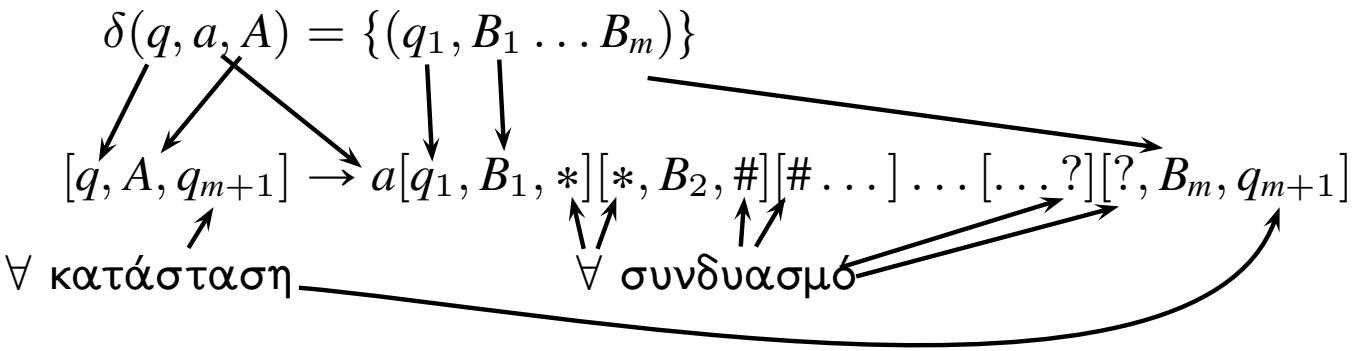
και σύνολο κανόνων παραγωγής P, οι οποίοι αποσκοπούν στο $[q, A, q_f] \overset{*}{\Rightarrow}_G x$ *ανν* $(q, x, A) \overset{*}{\underset{M_a}{\vdash}} (q_f, \varepsilon, \varepsilon)$, δηλαδή το M_a πηγαίνει από την κατάσταση q στην κατάσταση q_f επεξεργαζόμενο την συμβολοσειρά x και σβήνοντας το A από την στοίβα.
Συγκεκριμένα:

1. $S \to [q_0, Z_0, q_f]$
2. Για κάθε $q,\ q_{m+1} \in Q,\ A \in \Gamma,\ a \in \Sigma \cup \{\varepsilon\}$:

$$[q, A, q_{m+1}] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$$

για κάθε συνδυασμό καταστάσεων $q_2, \dots, q_m$ αν και μόνον αν ισχύει $\delta(q, a, A) \ni (q_1, B_1 \dots B_m)$. (Π.χ. για $m = 0$: $[q, A, q_1] \to a$.)

Ισοδυναμία PDA με διαφορετικές αποδοχές VIII



Σχήμα: Κατασκευή κανόνα παραγωγής από κανόνα μετάβασης

Πλέον αρκεί να δείξουμε με επαγωγή στο πλήθος των υπολογιστικών βημάτων και το μήκος της ακολουθίας παραγωγών ότι ισχύει: $[q, A, p] \overset{*}{\Rightarrow}_G x$ **ανν** $(q, x, A) \overset{*}{\underset{M_a}{\vdash}} (p, \varepsilon, \varepsilon)$ κι επομένως $[q_0, Z_0, q_f] \overset{*}{\Rightarrow}_G x$ **ανν** $(q_0, x, Z_0) \overset{*}{\underset{M_a}{\vdash}} (q_f, \varepsilon, \varepsilon)$ και άρα $S \overset{*}{\Rightarrow}_G x$ **ανν** $(q_0, x, Z_0) \overset{*}{\underset{M_a}{\vdash}} (q_f, \varepsilon, \varepsilon)$, δηλαδή $L(G) = L(M_a)$.

Ισοδυναμία PDA με διαφορετικές αποδοχές IX

Παράδειγμα:

Έστω το pda:

$$M = (\{q_0, q_f\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \{q_f\}),$$

όπου:

$$\delta(q_0, 0, Z_0) = \{q_0, XZ_0\}, \quad \delta(q_0, 0, X) = \{q_0, XX\},$$
$$\delta(q_0, 1, X) = \delta(q_f, 1, X) = \delta(q_f, \varepsilon, X) = \delta(q_f, \varepsilon, Z_0) = \{q_f, \varepsilon\}.$$

Το pda λειτουργεί ως εξής: push(X) όσο συναντάς 0 και μετά pop(X) όσο συναντάς 1 ή αν φτάσεις στο τέλος της εισόδου μέχρι να αδειάσει η στοίβα. Η γλώσσα που αναγνωρίζεται από το pda είναι:

$$L(M) = \{0^m 1^n \mid m \geqslant n \geqslant 1\}.$$

Θα κατασκευάσουμε γραμματική $G = (V, T, P, S)$ που παραγάγει την $L(M)$ με την μέθοδο της προηγούμενης απόδειξης. Για λόγους αναγνωσιμότητας έχουν παραλειφθεί τα κόμματα στις τριάδες. Έχουμε $T = \{0, 1\}$ και

$$V = \{S, [q_0Xq_0], [q_0Xq_f], [q_fXq_0], [q_fXq_f], [q_0Z_0q_0], [q_0Z_0q_f], [q_fZ_0q_0], [q_fZ_0q_f]\}$$

Ισοδυναμία PDA με διαφορετικές αποδοχές X

Οι κανόνες παραγωγής P είναι:

$$S \to [q_0 Z_0 q_f] \tag{1}$$

$$\left.\begin{array}{l}[q_0 Z_0 q_f] \to 0[q_0 X q_0][q_0 Z_0 q_f] \\ [q_0 Z_0 q_f] \to 0[q_0 X q_f][q_f Z_0 q_f]\end{array}\right\} \quad \text{αφού } \delta(q_0, 0, Z_0) = \{q_0, XZ_0\} \tag{2}$$

$$\left.\begin{array}{l}[q_0 X q_0] \to 0[q_0 X q_0][q_0 X q_0] \\ [q_0 X q_0] \to 0[q_0 X q_f][q_f X q_0]\end{array}\right\} \quad \text{αφού } \delta(q_0, 0, X) = \{q_0, XX\} \tag{3}$$

$$[q_0 X q_f] \to 1 \quad \text{αφού } \delta(q_0, 1, X) = \{q_f, \varepsilon\} \tag{4}$$

$$[q_f 0 q_f] \to \varepsilon \quad \text{αφού } \delta(q_f, \varepsilon, Z_0) = \{q_f, \varepsilon\} \tag{5}$$

$$\left.\begin{array}{l}[q_0 X q_f] \to 0[q_0 X q_0][q_0 X q_f] \\ [q_0 X q_f] \to 0[q_0 X q_f][q_f X q_f]\end{array}\right\} \quad \text{αφού } \delta(q_0, 0, X) = \{q_0, XX\} \tag{6}$$

$$[q_f X q_0] \to \text{ πουθενά} \quad \text{αφού } \delta(q_f, ...) = \{q_0, ..\} \text{ δεν ορίζεται} \tag{7}$$

$$[q_f X q_f] \to \varepsilon \mid 1 \quad \text{αφού } \delta(q_f, \begin{matrix}\varepsilon \\ 1\end{matrix}, X) = \{q_f, \varepsilon\} \tag{8}$$

Ισοδυναμία PDA με διαφορετικές αποδοχές XI

Αν παραλείψουμε τα μη παραγωγικά σύμβολα $([q_f X q_0], [q_0 X q_0])$ και τις παραγωγές στις οποίες εμφανίζονται και θέτοντας:

$$S \equiv [q_0 Z_0 q_f], \quad A \equiv [q_0 X q_f], \quad B \equiv [q_f Z_0 q_f], \quad C \equiv [q_f X q_f],$$

προκύπτει η γραμματική:

$$S \to 0AB, \quad A \to 0AC \mid 1, \quad B \to \varepsilon, \quad C \to \varepsilon \mid 1,$$

η οποία απλοποιείται στην:

$$S \to 0A, \quad A \to 0A \mid 0A1 \mid 1,$$

και τελικά στην:

$$S \to 0S \mid 0S1 \mid 01,$$

οπότε προφανώς $L(G) = L(M)$.

Ισοδυναμία PDA με διαφορετικές αποδοχές XII

Εναλλακτικές αποδείξεις:

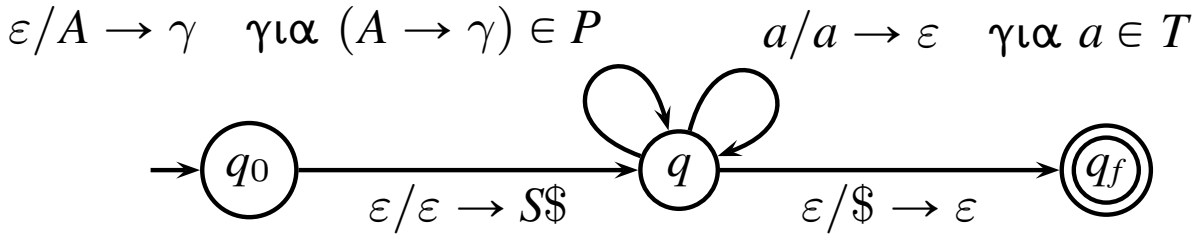
Μία άλλη ιδέα για κατασκευή pda δεδομένης γραμματικής χωρίς συμφραζόμενα, μόνον με μία κατάσταση στο pda: $M_a = (\{q_0\}, T, T \cup V, \delta, q_0, S, \{q_0\})$

1. $\delta(q_0, \varepsilon, A) \ni (q_0, \gamma)$ **ανν** $(A \to \gamma) \in P$,
2. $\delta(q_0, a, a) \ni (q_0, \varepsilon)$, για κάθε $a \in T$.

Δείχνουμε $S \overset{*}{\Rightarrow}_G x$ **ανν** $(q_0, x, S) \overset{*}{\underset{M_a}{\vdash}} (q_0, \varepsilon, \varepsilon)$.

Ισοδυναμία PDA με διαφορετικές αποδοχές XIII

Στο βιβλίο του ο Sipser παριστάνει τα pda με διαγράμματα παρόμοια με αυτά των FA: οι καταστάσεις παριστάνονται μέσα σε κύκλο ενώ οι μεταβάσεις με βέλη που είναι επιγεγραμμένα με παραστάσεις της μορφής $x/y \to z$ και σημαίνουν διάβασε το x από την είσοδο αν υπάρχει το y στην κορυφή της στοίβας, pop(y) από την στοίβα και push(z) στην στοίβα. Έτσι, δεδομένης γραμματικής G το αντίστοιχο pda είναι:



Επίσης, για την κατασκευή γραμματικής χωρίς συμφραζόμενα δεδομένου pda ο Sipser χρησιμοποιεί μη τερματικά σύμβολα: $V = \{A_{pq} \mid p, q \in Q\}$ και κανόνες της μορφής:

- $\left.\begin{array}{l}(r,t) \in \delta(p,a,\varepsilon)\\(q,\varepsilon) \in \delta(s,b,t)\end{array}\right\} : A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$

Pumping λήμματα για c.f. γλώσσες Ι

Λήμμα (Pumping Lemma για c.f. γλώσσες)

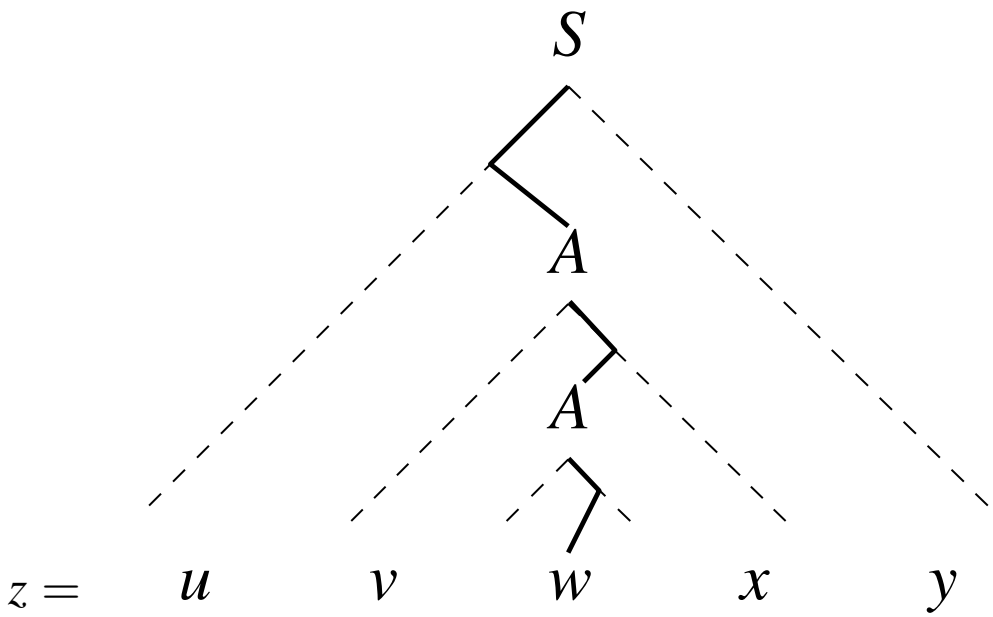
Αν L είναι μία γλώσσα χωρίς συμφραζόμενα τότε υπάρχει $n \in \mathbb{N}$ τέτοιο ώστε για κάθε $z \in L$ με $|z| \geqslant n$ υπάρχουν $u, v, w, x, y \in \Sigma^*$ τέτοια ώστε:

$$z = uvwxy \wedge |vwx| \leqslant n \wedge |vx| \geqslant 1 \wedge (\forall i \in \mathbb{N}: uv^i wx^i y \in L)$$

Απόδειξη:

Έστω G γραμματική σε κανονική μορφή Chomsky που παράγει την $L - \{\varepsilon\}$. Έστω $k = |V|$ (= πλήθος των μη τερματικών συμβόλων). Θεωρούμε $n = 2^k$ και $|z| \geqslant n$. Ένα συντακτικό δένδρο για το z έχει ύψος μεγαλύτερο ή ίσο του $\log n + 1 = k + 1$, επομένως υπάρχει κάποιο μονοπάτι από την ρίζα S σε ένα φύλλο του συντακτικού δένδρου με μήκος μεγαλύτερο ή ίσο του $k + 1$. Θεωρούμε το μακρύτερο τέτοιο μονοπάτι P. Διατρέχουμε αυτό το μονοπάτι από το φύλλο προς την ρίζα και βρίσκουμε το πρώτο μη τερματικό σύμβολο, που επαναλαμβάνεται, έστω A (σίγουρα κάποιο μη τερματικό επαναλαμβάνεται επειδή υπάρχουν μόνον k διαφορετικά μη τερματικά).

Pumping λήμματα για c.f. γλώσσες II



Σχήμα: Συντακτικό δένδρο pumping λήμματος

Pumping λήμματα για c.f. γλώσσες III

Στο συντακτικό δένδρο του σχήματος έχουμε $|vwx| \leqslant n$, επειδή το A είναι το πρώτο μη τερματικό από το φύλλο που επαναλαμβάνεται και $|vx| \geqslant 1$ επειδή το συντακτικό δένδρο προκύπτει από γραμματική σε μορφή Chomsky (άρα δυαδικό).

Έτσι, $A \overset{*}{\Rightarrow} vAx$ και $A \overset{*}{\Rightarrow} w$. Επομένως, θα έχουμε και $A \overset{*}{\Rightarrow} v^i w x^i$ για κάθε $i \in \mathbb{N}$ καθώς και $S \overset{*}{\Rightarrow} uv^i w x^i y$, δηλαδή $uv^i w x^i y \in L$ για κάθε $i \in \mathbb{N}$. □

Pumping λήμματα για c.f. γλώσσες IV

Όπως, και στις κανονικές γλώσσες, το παραπάνω pumping λήμμα μας βοηθάει να αποδείξουμε ότι κάποιες γλώσσες δεν είναι context free.

Παράδειγμα

$L = \{a^i b^i c^i \mid i \in \mathbb{N}\}$. Ας υποθέσουμε ότι η L είναι c.f. Τότε θεωρώντας το n του pumping λήμματος επιλέγουμε $z = a^n b^n c^n$. Τότε θα υπάρχουν $uvwxy = z$ με $|vwx| \leqslant n$ και $|vx| \geqslant 1$. Διακρίνουμε περιπτώσεις για το vwx και ελέγχουμε αν $uwy \in L$ (δηλαδή το $uv^i wx^i y$ για $i = 0$ από το P.L.)

1. $vwx = a^k$, τότε το uwy έχει περισσότερα c από a.
2. $vwx = a^k b^l$, τότε το uwy έχει περισσότερα c από a ή b.
3. $vwx = b^k$, τότε το uwy έχει περισσότερα c από b.
4. $vwx = b^k c^l$, τότε το uwy έχει περισσότερα a από b ή c.
5. $vwx = c^k$, τότε το uwy έχει περισσότερα a από c.

Δηλαδή σε καμμία περίπτωση δεν έχουμε $uwy \in L$. Άτοπο.

Pumping λήμματα για c.f. γλώσσες V

Το Pumping λήμμα μπορεί να χρησιμοποιηθεί παρομοίως, όπως παραπάνω, για να δειχθεί ότι η γλώσσα $\{a^i b^j c^k \mid i \leqslant j \leqslant k\}$ δεν είναι context free.

Άλλο ένα παράδειγμα, κάπως διαφορετικής γλώσσας, είναι το εξής:

Παράδειγμα

$$L = \{a^i b^j c^i d^j \mid i, j \in \mathbb{N}\}$$

Η παραπάνω γλώσσα επίσης δεν είναι context free. Παρομοίως, όπως παραπάνω, υποθέτουμε ότι είναι c.f., οπότε θα ισχύει το Pumping λήμμα.
Για το n του P.L. θεωρούμε την $z = a^n b^n c^n d^n$. Όπως παραπάνω, για κάθε πιθανό vwx (περιπτώσεις a^k, $a^k b^l$, b^k, $b^k c^l$, c^k, $c^k d^l$, d^k) δείχνουμε ότι $uwy \notin L$. Άτοπο.

Pumping λήμματα για c.f. γλώσσες VI

Το pumping λήμμα, αν και είναι πολύ χρήσιμο, υπάρχουν περιπτώσεις μη c.f. γλωσσών για τις οποίες δεν βοηθάει. Για τέτοιες γλώσσες χρειαζόμαστε ένα γενικευμένο P.L. που να επιτρέπει να κάνουμε pump επιλεκτικά σε προκαθορισμένες θέσεις.

Λήμμα (Ogden's Lemma)

Αν L είναι μία γλώσσα χωρίς συμφραζόμενα τότε υπάρχει $n \in \mathbb{N}$ τέτοιο ώστε για κάθε συμβολοσειρά $z \in L$ στην οποία σημαδεύουμε *n ή περισσότερες θέσεις υπάρχουν $u, v, w, x, y \in \Sigma^*$ τέτοια ώστε:*

$$z = uvwxy \wedge \text{ το } vwx \text{ έχει το πολύ } n \text{ σημαδεμένες θέσεις } \wedge$$

$$\text{τα } v \text{ και } x \text{ έχουν το λιγότερο } 1 \text{ σημαδεμένη θέση } \wedge (\forall i \in \mathbb{N} : uv^i wx^i y \in L)$$

Σκαρίφημα απόδειξης.

Παρόμοια με την απόδειξη του Pumping λήμματος. Θέτουμε $n = 2^k + 1$. Θεωρούμε το «μακρύτερο» μονοπάτι με τους περισσότερους απογόνους με σημαδεμένες θέσεις. Υπάρχουν τουλάχιστον $k + 1$ σημεία διακλάδωσης (σημεία όπου και οι δύο απόγονοι καταλήγουν σε σημαδεμένη θέση) στο P. Επομένως, υπάρχει μη τερματικό που επαναλαμβάνεται και η απόδειξη συνεχίζεται όπως στο P.L. □

Pumping λήμματα για c.f. γλώσσες VII

Παράδειγμα

Θα αποδείξουμε ότι η $L = \{a^i b^j c^k d^l \mid i = 0 \vee j = k = l\}$ δεν είναι context free. Παρομοίως, όπως παραπάνω, υποθέτουμε ότι είναι c.f., οπότε θα ισχύει το λήμμα του Ogden. Για το n του Ogden θεωρούμε την $z = ab^n c^n d^n$, όπου σημαδεύουμε όλα τα b: $a\underline{b^n} c^n d^n$. Τότε θα υπάρχουν τα $uvwxy = z$ του λήμματος του Ogden. Διακρίνουμε δύο περιπτώσεις:

1. το v ή το x περιέχουν δύο διαφορετικά σύμβολα, τότε το pumping καταστρέφει την σειρά·
2. ένα από τα v, x περιέχει b και το άλλο άλλα σύμβολα, τότε το pumping καταστρέφει το ισάριθμο των συμβόλων.

Ιδιότητες κλεισίματος Ι

Θεώρημα

Οι γλώσσες χωρίς συμφραζόμενα είναι κλειστές ως προς ένωση, παράθεση και κλείσιμο Kleene.

Απόδειξη.

Έστω γραμματικές για γλώσσες χωρίς συμφραζόμενα με αξιώματα S_1 και S_2. Χωρίς βλάβη της γενικότητας, υποθέτουμε ότι οι γραμματικές δεν έχουν κοινά μη τερματικά σύμβολα (αυτή η υπόθεση γίνεται και στις παρακάτω αποδείξεις). Κατασκευάζουμε γραμματική με αρχικό σύμβολο S, τους κανόνες των επί μέρους γραμματικών και έναν επιπλέον κανόνα για κάθε πράξη:

$$\cup : S \to S_1 \mid S_2$$
$$\text{παράθεση} : S \to S_1 S_2$$
$$* : S \to S_1 S \mid \varepsilon$$

□

Ιδιότητες κλεισίματος II

Θεώρημα

Οι γλώσσες χωρίς συμφραζόμενα είναι κλειστές ως προς i) context free αντικατάσταση, ii) ομομορφισμό και iii) αντίστροφο ομομορφισμό.

Απόδειξη.

i) Έστω η αντικατάσταση $f: a \mapsto L_a$ με L_a γλώσσα χωρίς συμφραζόμενα, για κάθε τερματικό σύμβολο a της αρχικής γραμματικής. Έστω ότι η κάθε L_a περιγράφεται με γραμματική G_a με αξίωμα S_a. Κατασκευάζουμε νέα γραμματική που περιέχει όλους τους κανόνες της αρχικής γραμματικής G και των G_a και αντικαθιστούμε κάθε εμφάνιση των τερματικών συμβόλων της G στους κανόνες της G με το αντίστοιχο αξίωμα S_a.

ii) Υποπερίπτωση του i).

iii) Έστω M pda που αποδέχεται την L και ομομορφισμός $h: \Sigma' \to \Sigma^*$. Κατασκευάζουμε pda M' που αποδέχεται την $h^{-1}(L)$. Το M' με είσοδο a παράγει το $h(a)$ και εξομοιώνει το $h(a)$ στο αυτόματο M. Κάτι τέτοιο είναι δυνατό δεδομένου του πεπερασμένου πλήθους των μη τερματικών συμβόλων (οπότε η εξομοίωση κάθε συμβόλου αποθηκεύεται στον πεπερασμένο έλεγχο του pda M').

□

Ιδιότητες κλεισίματος III

Θεώρημα

Οι γλώσσες χωρίς συμφραζόμενα δεν είναι κλειστές ως προς i) τομή και ii) συμπλήρωμα.

Απόδειξη.

i) Αντιπαράδειγμα:

$$\underbrace{\{a^i b^i c^j \mid i,j \in \mathbb{N}\}}_{\text{c.f. γλώσσα}} \cap \underbrace{\{a^i b^j c^j \mid i,j \in \mathbb{N}\}}_{\text{c.f. γλώσσα}} = \underbrace{\{a^i b^i c^i \mid i \in \mathbb{N}\}}_{\text{όχι c.f.}}$$

(Βλέπε παράδειγμα σελ. 91 για την $\{a^i b^i c^i \mid i \in \mathbb{N}\}$.)

ii) Από το νόμο de Morgan: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, επειδή οι c.f. γλώσσες είναι κλειστές ως προς ένωση, αν ήταν κλειστές και ως προς συμπλήρωμα, τότε θα ήταν και ως προς τομή· άτοπο. □

Παρ' όλα αυτά, η τομή μίας γλώσσας χωρίς συμφραζόμενα με μία κανονική γλώσσα είναι γλώσσα χωρίς συμφραζόμενα:

Ιδιότητες κλεισίματος IV

Θεώρημα

Έστω L γλώσσα χωρίς συμφραζόμενα και R κανονική γλώσσα. Τότε η $L \cap R$ είναι γλώσσα χωρίς συμφραζόμενα.

Ιδέα απόδειξης.

Κατασκευάζουμε ένα pda που εξομοιώνει παράλληλα το pda που αναγνωρίζει την L και το fa (πεπερασμένο αυτόματο) που αναγνωρίζει την R. □

Δίνουμε και μία εφαρμογή του παραπάνω θεωρήματος:

Παράδειγμα

Η $L = \{ww \mid w \in (a+b)^*\}$ δεν είναι c.f. Πράγματι η τομή της L με την κανονική γλώσσα $a^*b^*a^*b^*$:

$$L \cap a^*b^*a^*b^* = \{a^i b^j a^i b^j \mid i, j \in \mathbb{N}\} = L_1$$

δεν είναι c.f. (Για το τελευταίο, θεωρήστε τον ομομορφισμό $h(a) = h(c) = a$, $h(b) = h(d) = b$ και δείτε ότι $h^{-1}(L_1) \cap a^*b^*c^*d^* = \{a^i b^j c^i d^j \mid i, j \in \mathbb{N}\} = L_2$ και για την L_2 έχουμε ήδη αποδείξει ότι δεν είναι c.f., σε προηγούμενο παράδειγμα.)

Αλγόριθμοι για c.f. γλώσσες Ι

Θεώρημα

Υπάρχουν αλγόριθμοι που αποκρίνονται αν μία γλώσσα χωρίς συμφραζόμενα είναι α) κενή, β) πεπερασμένη ή άπειρη.

Απόδειξη.

α) Είτε με την βοήθεια του pumping λήμματος, είτε με εύρεση των χρήσιμων (δηλαδή προσβάσιμων από το αξίωμα και παραγωγικών) μη τερματικών συμβόλων.

β) Πάλι, είτε με την βοήθεια του pumping λήμματος, είτε, θεωρώντας ότι η γραμματική είναι σε κανονική μορφή Chomsky με σχεδιασμό του γράφου με κόμβους τα χρήσιμα μη τερματικά σύμβολα και κατευθυνόμενες ακμές, έστω π.χ. από το A στο B, αν υπάρχει κανόνας $A \to BC$ ή $A \to CB$. Η γλώσσα είναι άπειρη αν και μόνον αν υπάρχει κύκλος στον γράφο (ισοδύναμα υπάρχει re-entrant μη τερματικό σύμβολο). □

Αλγόριθμοι για c.f. γλώσσες II

Δεδομένης γραμματικής χωρίς συμφραζόμενα G, υπάρχει μηχανιστικός αλγόριθμος ο οποίος για οποιαδήποτε συμβολοσειρά x αποκρίνεται αν $x \in L(G)$ ή όχι.

Π.χ. αν συστηματικά κατασκευάσουμε όλες τις παραγόμενες συμβολοσειρές κατά αύξουσα σειρά μήκους, τότε μπορούμε να αποφασίσουμε εάν $x \in L(G)$. Ο αλγόριθμος όμως είναι εκθετικού χρόνου ως προς το μήκος της συμβολοσειράς εισόδου.

Αν όμως η γραμματική δίνεται σε κανονική μορφή Chomsky, τότε υπάρχει ταχύτερος αλγόριθμος, πολυπλοκότητας $O(|x|^3)$, ο λεγόμενος αλγόριθμος CYK (από τους Cocke, Younger, Kasami).

Αλγόριθμοι για c.f. γλώσσες III

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

Ο αλγόριθμος ουσιαστικά ελέγχει όλα τα δυνατά συντακτικά δένδρα, αρχίζοντας από τα τερματικά σύμβολα στα φύλλα και αποδίδοντας σε αυτά πιθανά μη τερματικά σύμβολα που τα παράγουν. Ο αλγόριθμος συνεχίζει αποδίδοντας πιθανά μη τερματικά σύμβολα σε κάθε υποσυμβολοσειρά της συμβολοσειράς εισόδου και τέλος ελέγχει αν στην συμβολοσειρά εισόδου έχει αποδοθεί το αξίωμα S.

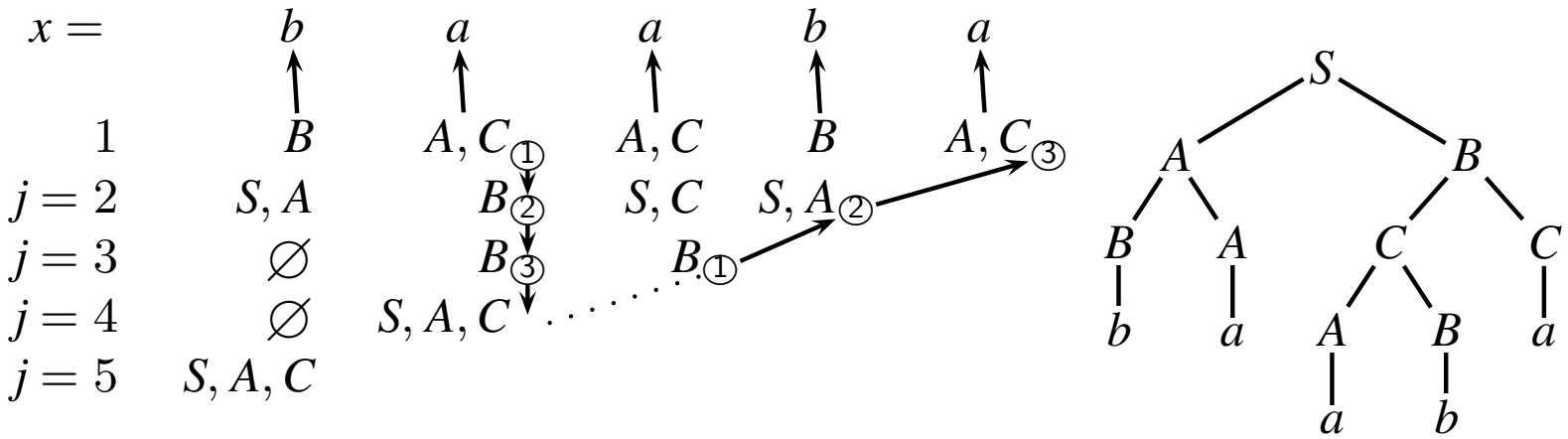
Αλγόριθμοι για c.f. γλώσσες IV

Παράδειγμα

Έστω γραμματική σε κανονική μορφή Chomsky:

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

Στο σχήμα δίνουμε την εκτέλεση του αλγορίθμου και το αντίστοιχο συντακτικό δένδρο για είσοδο $x = baaba$. Με τα βέλη, δείχνουμε την ακολουθία υπολογισμού για $j = 4$ (μήκος substring) και για το substring $(x)_{2..5}$ (αρχίζει, δηλαδή, από την θέση $i = 2$): Μέσα σε κύκλο και με τον ίδιο αριθμό συμβολίζουμε τα δύο substrings (συνολικού μήκους 4) που συνδυάζονται για να δώσουν το $(x)_{2..5}$.



Σχήμα: Εκτέλεση αλγορίθμου CYK

Αποτελέσματα μη αποκρισιμότητας Ι

Για τις γραμματικές χωρίς συμφραζόμενα, υπάρχουν αρκετά σημαντικά προβλήματα απόφασης για τα οποία δεν υπάρχει *μηχανιστικός αλγόριθμος*. Αναφέρουμε, χωρίς απόδειξη, τα εξής αποτελέσματα:

- Δεν υπάρχει μηχανιστικός αλγόριθμος που αποκρίνεται δεδομένων δύο γραμματικών χωρίς συμφραζόμενα G_1 και G_2, αν $L(G_1) = L(G_2)$. (ισότητα — equality)
- Δεν υπάρχει μηχανιστικός αλγόριθμος που αποκρίνεται δεδομένων δύο γραμματικών χωρίς συμφραζόμενα G_1 και G_2, αν $L(G_1) \cap L(G_2) \neq \varnothing$. (μη κενή τομή — intersection non-emptiness)
- Δεν υπάρχει μηχανιστικός αλγόριθμος που αποκρίνεται δεδομένης γραμματικής χωρίς συμφραζόμενα G, αν η $L(G)$ είναι κανονική. (κανονικότητα — regularity)

Η κατάσταση σε σχέση με τις κανονικές γλώσσες έχει όπως φαίνεται στον επόμενο πίνακα.

Αποτελέσματα μη αποκρισιμότητας II

	αποκρίσιμα	**μη αποκρίσιμα**
regular	ισότητα μη κενή τομή (μη) κενή γλώσσα άπειρη γλώσσα ανήκειν	
c.f.	ανήκειν (μη) κενή γλώσσα άπειρη γλώσσα	μη κενή τομή ισότητα κανονικότητα

Πίνακας: Προβλήματα απόφασης για regular και c.f. γλώσσες

Γραμμικότητα και ημιγραμμικότητα Ι

Η παρακάτω μέθοδος μας βοηθάει να αποδείξουμε ότι μια γλώσσα δεν είναι c.f..

Ορισμός

Ένα υποσύνολο του $\mathbb{N}$ ονομάζεται **γραμμικό** αν τα στοιχεία του σχηματίζουν αριθμητική πρόοδο, είναι δηλαδή της μορφής $\{x_0 + kd \mid k \in \mathbb{N}\}$, όπου x_0, d σταθερές.

Παράδειγμα

$A = \{y \mid y = 3x + 1, x \in \mathbb{N}\} = \{1, 4, 7, 10, \dots\}$

Ορισμός

Ένα υποσύνολο του $\mathbb{N}$ ονομάζεται **ημιγραμμικό** αν είναι πεπερασμένη ένωση γραμμικών συνόλων.

Γραμμικότητα και ημιγραμμικότητα II

Παράδειγμα

$B = \{y \mid y = 4x - 3, x \in \mathbb{N}\} = \{1, 5, 9, 13, \dots\}$ γραμμικό.
$A \cup B = \{1, 4, 5, 7, 9, 10, \dots\}$ ημιγραμμικό σύνολο.

Παρατήρηση

Στα ημιγραμμικά σύνολα οι αποστάσεις μεταξύ των στοιχείων δεν αυξάνονται.

Ισχύει το παρακάτω σημαντικό θεώρημα:

Θεώρημα (Parikh)

Για κάθε γραμματική χωρίς συμφραζόμενα G το σύνολο $\{|x| \mid x \in L(G)\}$ είναι ημιγραμμικό.

Παράδειγμα

Από την G: $S \to aSb \mid \varepsilon$ προκύπτει η γλώσσα $L(G) = \{a^n b^n \mid n \in \mathbb{N}\}$ και το σύνολο μηκών $\{2n \mid n \in \mathbb{N}\}$ που είναι γραμμικό.

Γραμμικότητα και ημιγραμμικότητα III

Παρατήρηση

Το Θ. Parikh ισχύει για οποιαδήποτε μετρική ιδιότητα των strings (δηλ. όχι μόνο για το μήκος των strings)

Παράδειγμα

Η $\{a^2, a^4, a^8, \ldots, a^{2^n}, \ldots\}$ δεν είναι c.f.

Πόρισμα

Κάθε γλώσσα χωρίς συμφραζόμενα με αλφάβητο ενός συμβόλου είναι κανονική.

Απόδειξη.

Αφήνεται ως άσκηση. □

Γραμμικότητα και ημιγραμμικότητα IV

Παράδειγμα

Επειδή η $\{a^2, a^4, a^8, \ldots, a^{2^n}, \ldots\}$ δεν είναι regular, δεν είναι ούτε c.f.

Παράδειγμα

$\{a^n b^m \mid m \geqslant n \vee (m \text{ πρώτος} \wedge n \geqslant m)\}$ δεν είναι c.f.

Περιεχόμενα

1 Πεπερασμένα Αυτόματα και Κανονικά Σύνολα

- Παραλλαγές, επεκτάσεις και εφαρμογές FA/REGEXP
- Ιδιότητες κανονικών συνόλων
- Αλγεβρική περιγραφή κανονικών συνόλων. Ελαχιστοποίηση DFA

2 Τυπικές Γλώσσες

- Τυπικές Γραμματικές
- Απλοποίηση c.f. γραμματικών
- Αυτόματα στοίβας (pushdown automata)
- Ιδιότητες c.f. γλωσσών

3 Μοντέλα Υπολογισμού

- Ιστορία - Εισαγωγή
- LOOP: Μια απλή γλώσσα προγραμματισμού
- LOOP-υπολογίσιμες και πρωταρχικές αναδρομικές συναρτήσεις

Το πρόγραμμα του Leibni(t)z

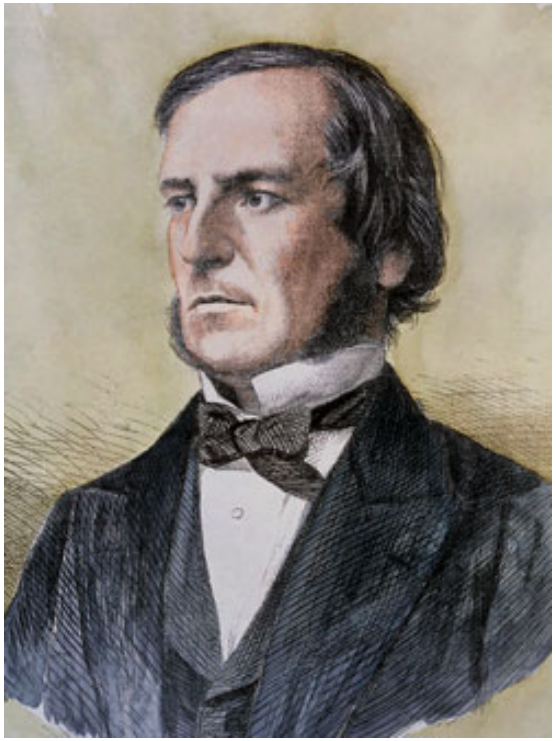
Ο Leibniz πρότεινε το εξής πρόγραμμα:

1. Να δημιουργηθεί μια **τυπική γλώσσα** (formal language), με την οποία να μπορούμε να περιγράψουμε όλες τις μαθηματικές έννοιες και προτάσεις.
2. Να δημιουργηθεί μια **μαθηματική θεωρία** (δηλαδή ένα σύνολο από αξιώματα και συμπερασματικούς κανόνες συνεπαγωγής), με την οποία να μπορούμε να αποδεικνύουμε όλες τις ορθές μαθηματικες προτάσεις.
3. Να αποδειχθεί ότι αυτή η θεωρία είναι **συνεπής** (consistent), (δηλαδή ότι η πρόταση "Α και όχι Α" $(A \wedge \neg A)$ δεν είναι δυνατόν να αποδειχθεί σ' αυτή τη θεωρία).

Το πρόγραμμα του Leibni(t)z

Gottfried Wilhelm Leibniz

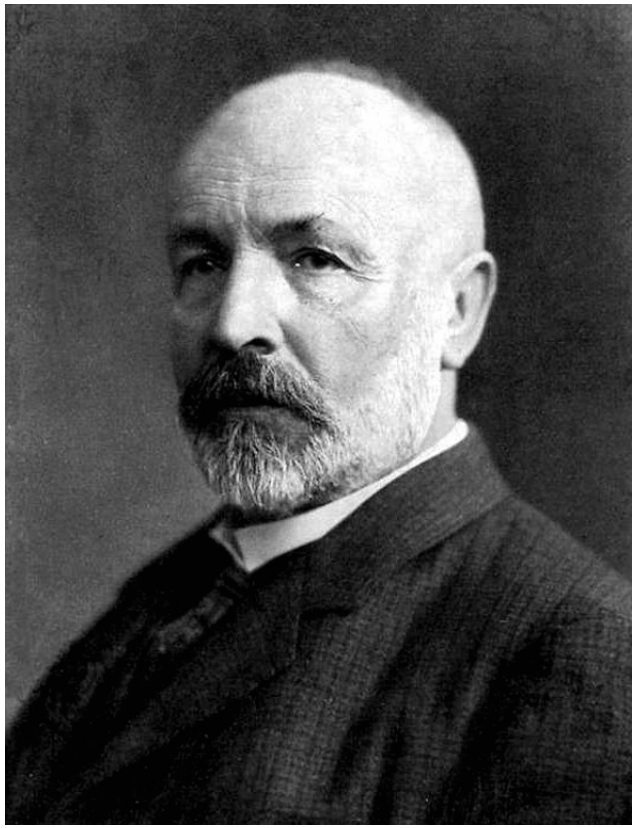
Υλοποίηση ... Γλώσσα



George Boole

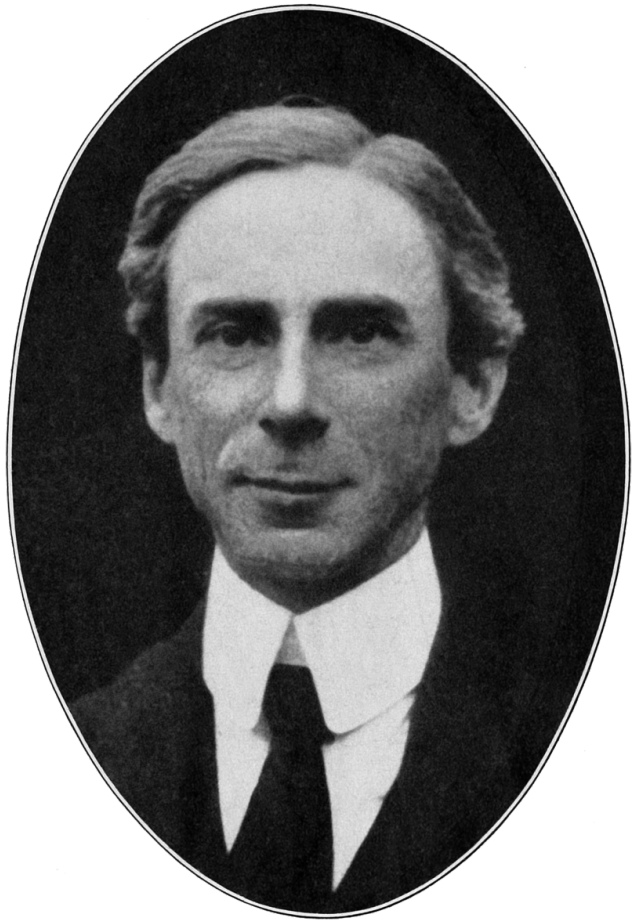
Gottlob Frege

Υλοποίηση ... Ενιαία θεωρία



Georg Cantor

Μη συνέπεια της αφελούς συνολοθεωρίας



Bertrand Russel

Πρόγραμμα Hilbert, Συνέπεια

David Hilbert

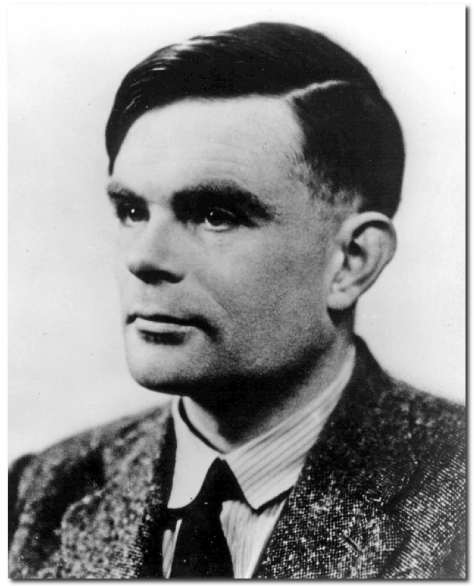
Kurt Gödel

Θέση (thesis) Church-Turing

Θέση (thesis) Church-Turing

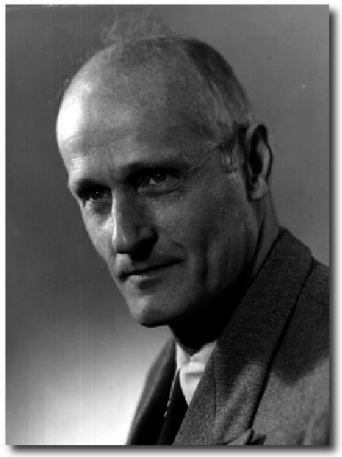
Όλα τα γνωστά και "άγνωστα" μοντέλα της έννοιας "υπολογίσιμος" είναι μηχανιστικά ισοδύναμα (effectively equivalent).

Alonzo Church



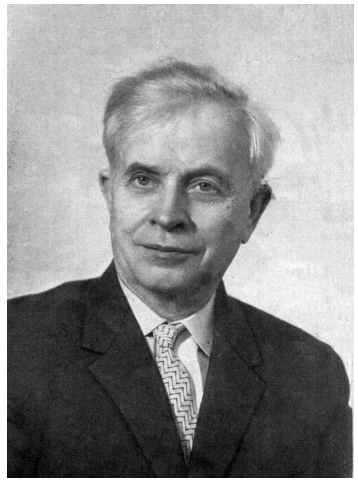
Alan Turing

Άλλα υπολογιστικά μοντέλα



Stephen Kleene

Emil Post



Andrei Andreevich Markov, jr.

Halting Problem

Θεώρημα

Το **Halting Problem (HP)** *είναι μη αποκρίσιμο.*

Απόδειξη.

1. Έστω ότι $\pi_0, \pi_1, \pi_2, \ldots$ είναι μια μηχανιστική απαρίθμηση (effective enumeration) όλων των προγραμμάτων.
2. π : read(n); **if** $\pi_n(n)$ terminates **then** loop_forever **else** halt
3. Διαγωνιοποίηση.

□

Υπολογισιμότητα

Ορισμός

Ένα σύνολο S λέγεται **αποκρίσιμο** ή **υπολογίσιμο** ή **επιλύσιμο** (decidable, computable, solvable) αν και μόνο αν υπάρχει ένας αλγόριθμος που σταματάει ή μια υπολογιστική μηχανή που δίνει έξοδο "ναι" για κάθε είσοδο $\alpha \in S$ και έξοδο "όχι" για κάθε είσοδο $\alpha \notin S$.

Ορισμός

Ένα σύνολο S λέγεται **καταγράψιμο** (με μηχανιστική γεννήτρια) (listable, effectively generatable) αν και μόνο αν υπάρχει μια γεννήτρια διαδικασία ή μηχανή που καταγράφει όλα τα στοιχεία του S. Στην, πιθανώς άπειρη, λίστα εξόδου επιτρέπονται οι επαναλήψεις και δεν υπάρχει περιορισμός για την διάταξη των στοιχείων.

Παρατήρηση

Το **HP** δεν είναι decidable είναι όμως listable.
Απόδειξη: Dovetailing.

Αυτοαναφορά

- Αυτή η πρόταση είναι ψευδής.
- **Παράδοξο (αντινομία) του Russel**:
  $A = \{x \mid x \notin x\}$. Τότε $A \in A$ ή $A \notin A$?
- Τον κουρέα σε ένα χωριό που ξυρίζει όλους όσους δεν ξυρίζονται μόνοι τους, ποιός τον ξυρίζει;
- Μεταπαιχνίδι.

Διαγωνιοποίηση

- $A : n \times n$ boolean πίνακας
- $d_i = A_{ii}$, για κάθε i, με $1 \leqslant i \leqslant n$.
- $D_i = 1 - d_i$, για κάθε i, με $1 \leqslant i \leqslant n$.
- Το διάνυσμα D δεν εμφανίζεται ούτε ως γραμμή, ούτε ως στήλη στην πίνακα A.

Διαγωνιοποίηση και Halting Problem

- Μηχανιστική απαρίθμηση των προγραμμάτων με μία είσοδο:

$$\pi_0, \pi_1, \pi_2, \ldots$$

- Γράφουμε $\pi_x(y) \downarrow$ αν το x-οστό πρόγραμμα με είσοδο $y \in \mathbb{N}$ τερματίζει. Διαφορετικά αν δεν τερματίζει γράφουμε $\pi_x(y) \uparrow$.
- Έστω ότι υπάρχει πρόγραμμα $\text{halt}(x, y)$ που λύνει το halting problem.
- π : read(n); **if** halt(n, n) **then** loop_forever **else** halt
- $\exists i : \pi_i = \pi$
- $\pi_i(i) \downarrow \Leftrightarrow \pi_i(i) \uparrow$

Διαγωνιοποίηση και μη πληρότητα του Gödel

- Μηχανιστική απαρίθμηση των τύπων (στην αριθμητική Peano) με μία ελεύθερη μεταβλητή:

$$\phi_0, \phi_1, \phi_2, \ldots$$

- Συνέπεια: Αν ϕ αποδείξιμος, τότε ϕ αληθής.
- Πληρότητα: Αν ϕ αληθής, τότε ϕ αποδείξιμος.
- $\phi(x) =$"δεν υπάρχει απόδειξη για $\phi_x(x)$".
- $\exists i : \phi_i = \phi$.
- $\phi_i(i)$: δεν μπορεί να είναι ψευδής.

Διαγωνιοποίηση και μη πληρότητα του Gödel

Θεώρημα (μη πληρότητας, Gödel)

Κάθε συνεπής αξιωματικοποίηση της αριθμητικής Peano είναι μη πλήρης.

Σημείωση: Δεν ισχύει το παραπάνω για την αριθμητική Presburger (που δεν έχει την πράξη *).

Περιγραφή της LOOP

- Δε χρειάζονται **δηλώσεις** (declarations)
- Τέσσερα ήδη **αναθέσεων** (assignments)

$$x := 0,\ x := y,\ x := y + 1,\ x := y \dot{-} 1$$

Παρατήρηση: $0 \dot{-} 1 = 0,\ (x + 1) \dot{-} 1 = x$

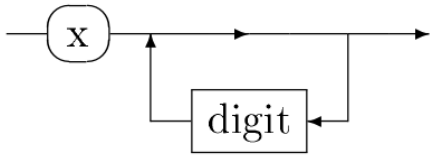
- **for** loop:
  for $i := 1$ **to** y **do** ... **end**

Παραδείγματα LOOP

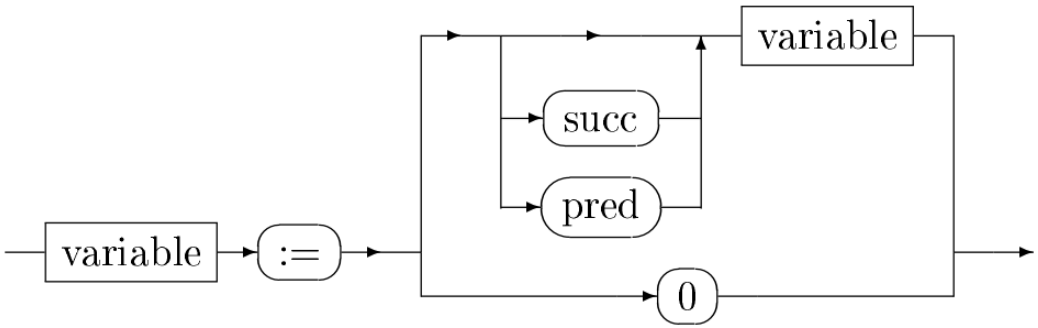
1. Πρόγραμμα για $x := y + z$ (δηλ. $\mathsf{add}(y, z)$):
   $x := y$; **for** $w := 1$ **to** z **do** $x := x + 1$ **end**
2. $x := y * z$ (δηλ. $\mathsf{mult}(y, z)$):
   $x := 0$; **for** $w := 1$ **to** z **do** $x := \mathsf{add}(x, y)$ **end**
3. $x := y \dot{-} z$ (δηλ. $\mathsf{sub}(y, z)$): $\left(\text{Παρατήρηση: } y \dot{-} z = \begin{cases} 0, & y < z \\ y - z, & \text{αλλιώς} \end{cases}\right)$
   $x := y$; **for** $w := 1$ **to** z **do** $x := x \dot{-} 1$ **end**
4. **if** $y = 0$ **then** $x := 0$ **else** $x := 1$ (δηλ. $\mathsf{ifnzero}(y)$):
   $x := 0$; $z := 0$; **for** $w := 1$ **to** y **do** $x := z + 1$ **end**
5. **if** $y = 0$ **then** $x := 1$ **else** $x := 0$ (δηλ. $\mathsf{ifzero}(y)$):
   $x := 0$; $x := x + 1$; **for** $w := 1$ **to** y **do** $x := 0$ **end**

Συντακτικά Διαγράμματα της LOOP I

- **Variable:**

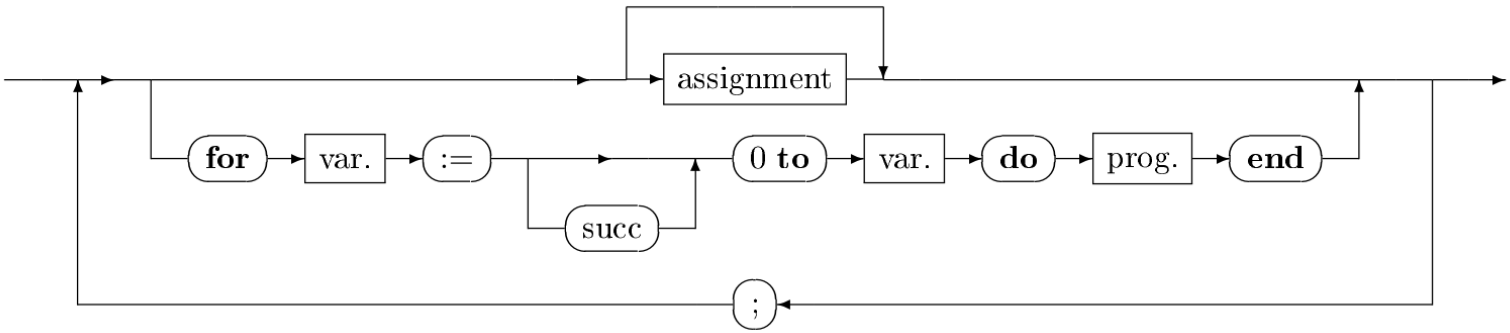


- **Assignment:**



Συντακτικά Διαγράμματα της LOOP II

- **Program:**



Επαγωγή και Αναδρομή

Η **επαγωγή** είναι:

- **Μέθοδος απόδειξης**
  - $A(0)$: Βάση
  - $\forall n : A(n) \rightarrow A(n+1)$: Επαγωγικό βήμα
- Τρόπος ορισμού **επαγωγικού πεδίου**
  - *Αρχικά στοιχεία* (initial objects)
  - Πράξεις για *κλείσιμο* (closure operations)

Για κάθε επαγωγικό πεδίο μπορούμε να αποδεικνύουμε ιδιότητες των αντικειμένων χρησιμοποιώντας τη μέθοδο της επαγωγής.

Επαγωγή Ι

Παραδείγματα:

1. $\mathbb{N}$ = το σύνολο των **φυσικών** αριθμών
   - Αρχικό στοιχείο: 0
   - Πράξη για κλείσιμο: successor

   $\mathbb{N} = \{0\} \cup \{n+1 \mid n \in \mathbb{N}\}$
2. Σ^* = το σύνολο όλων των **strings** του αλφάβητου Σ:
   - αρχικά στοιχεία: ε, α για κάθε $\alpha \in \Sigma$
   - πράξεις για κλείσιμο: concatenation
3. Σύνολο **θεωρημάτων**:
   - αρχικά στοιχεία: αξιώματα
   - πράξεις για κλείσιμο: συμπερασματικοί κανόνες (inference rules)
4. Σύνολο **όρων** σε μία μαθηματική θεωρία:
   - αρχικά στοιχεία: σταθερές και μεταβλητές
   - πράξεις για κλείσιμο: σύνθεση (composition) συναρτησιακών συμβόλων

Επαγωγή II

5. **Σύντακτικά ορθοί τύποι** (καλώς, ορισμένοι, wff) σε μία θεωρία μόνο με το κατηγορηματικό σύμβολο "$=$":
   - αρχικά στοιχεία: τύποι της μορφής όρος$_1$ $=$ όρος$_2$
   - πράξεις για κλείσιμο: της μορφής $\neg\Phi, \Phi \wedge \Psi, \Phi \vee \Psi, \Phi \rightarrow \Psi, \Phi \leftrightarrow \Psi, \exists x\Phi, \forall x\Phi$, όπου Φ, Ψ είναι τύποι και x είναι μεταβλητή.
6. Σύνολο **προγραμμάτων LOOP**:
   - αρχικά στοιχεία: της μορφής $x := 0, x := y, x := y + 1, x := y \dot{-} 1$ και το κενό πρόγραμμα.
   - πράξεις για κλείσιμο: "$;$" (παράθεση), "**for**" (βρόχος)
7. $S_1 = \{0\} \cup \{n + 2 \mid n \in S_1\}$
8. $S_2 = \{3, 5\} \cup \{2n + m \mid m, n \in S_2 \wedge n < m\}$
9. $S_3 = \{0\} \cup \{n + 2 \mid n \in S_3\} \cup \{n + 5 \mid n \in S_3\}$

Αναδρομή Ι

Ορισμός

Αναδρομικό πεδίο = **Επαγωγικό πεδίο** + **Unique parsing**

Παραδείγματα:

1. $\mathbb{N}$
2. Το Σ^* με τον παραπάνω ορισμό *δεν* είναι αναδρομικό.

$$abb = \text{conc}(ab, b) = \text{conc}(a, bb)$$

Όμως το Σ^* δεν είναι *ουσιωδώς διφορούμενο* (inherently ambiguous). Με τον παρακάτω ορισμό το Σ^* είναι αναδρομικό:

$$\Sigma^* = \{\varepsilon\} \cup \{wa |\ w \in \Sigma^* \wedge a \in \Sigma\}$$

3. Το S_1 είναι αναδρομικό ενώ το S_3 δεν είναι. π.χ. $10 = 0+2+2+2+2 = 0+5+5$.

Αναδρομή ΙΙ

4. $A = \{a, ab, ba\}$. Το A^* είναι επαγωγικό πεδίο αλλά ουσιωδώς διφορούμενο (μη αναδρομικό). π.χ. $aba = \text{conc}(a, ba) = \text{conc}(ab, a)$.

Σε αναδρομικά πεδία μπορούμε να ορίζουμε μονοσήμαντα συναρτήσεις χρησιμοποιώντας τη μέθοδο της αναδρομής:

1. **Παραγοντικό**: $\begin{cases} 0! = 1 \\ (n+1)! = n! \cdot (n+1) \end{cases}$
2. Ακολουθία **Fibonacci**: $\begin{cases} f(0) = 1 \\ f(1) = 1 \\ f(n+2) = f(n) + f(n+1) \end{cases}$

Αναδρομή III

3. Συνάρτηση παρόμοια με αυτή του Ackermann:

$$
\begin{aligned}
f(x,y,0) &= y+1 \\
f(x,y,1) &= x+y = x+1+1+\ldots+1 \text{ (y φορές)} \\
f(x,y,2) &= x\cdot y = x+x+x+\ldots+x \text{ (y φορές)} \\
f(x,y,3) &= x^y = x\cdot x\cdot x\cdot\ldots\cdot x \text{ (y φορές)} \\
f(x,y,4) &= x^{x^{x^{\cdot^{\cdot^{\cdot^{x}}}}}} \text{ (y φορές)} \\
&\ldots
\end{aligned}
$$

4. $\begin{cases} x+0=x \\ x+Sy=S(x+y) \end{cases}$ ή $\begin{cases} \mathsf{add}(x,0)=x \\ \mathsf{add}(x,Sy)=S(\mathsf{add}(x,y)) \end{cases}$

Αναδρομή IV

5. $\begin{cases} x * 0 = 0 \\ x * Sy = x + x * y \end{cases}$
6. $\begin{cases} x \dot{-} 0 = x \\ x \dot{-} Sy = P(x \dot{-} y) \end{cases}$, όπου $P(n) = n \dot{-} 1$
7. $\begin{cases} sg(0) = 0 \\ sg(Sx) = 1 \end{cases}$ ή $sg(x) = x \dot{-} P(x)$
8. $\begin{cases} \overline{sg}(0) = 1 \\ \overline{sg}(Sx) = 0 \end{cases}$ ή $\overline{sg}(x) = 1 \dot{-} x$

Πρωταρχική αναδρομή Ι

Σχήμα **πρωταρχικής αναδρομής** (scheme of primitive recursion):

$$\begin{cases} f(x,0) = g(x) \\ f(x,Sy) = h(x,y,f(x,y)) \end{cases}$$

Ορισμός

Η κλάση των **primitive recursive functions** (πρωταρχικά αναδρομικών συναρτήσεων) $\mathcal{P}$ είναι η μικρότερη κλάση συναρτήσεων που:

- περιέχει τις εξής αρχικές συναρτήσεις: S, P, Z, U_i^n (για όλα τα $n, i : 1 \leqslant i \leqslant n$) και
- είναι κλειστή ως προς τα σχήματα σύνθεσης και πρωταρχικής αναδρομής.

Πρωταρχική αναδρομή II

Ορισμός

Η κλάση των **primitive recursive functions** (πρωταρχικά αναδρομικών συναρτήσεων) $\mathcal{P}$ είναι η μικρότερη κλάση συναρτήσεων που:

- περιέχει τις εξής αρχικές συναρτήσεις: S, P, Z, U_i^n (για όλα τα $n, i : 1 \leqslant i \leqslant n$) και
- είναι κλειστή ως προς τα σχήματα σύνθεσης και πρωταρχικής αναδρομής.

Διευκρινίσεις:

- $S(x) = x + 1$ (επόμενο)
- $P(x) = x \dot{-} 1$ (προηγούμενο)
- $Z(x) = 0$ (μηδενική)
- $U_i^n(x_1, \ldots, x_n) = x_i,\ 1 \leqslant i \leqslant n$ (προβολές)
- Σύνθεση: π.χ.: $f(x, y) = g(h_1(x, y), h_2(x, y), h_3(x, y))$
- Πρωταρχική αναδρομή: π.χ.:

$$\begin{cases} f(0) = c \\ f(Sy) = h(y, f(y)) \end{cases}$$

LOOP-υπολογισιμότητα

Ορισμός

Μια συνάρτηση $f\colon \mathbb{N}^n \to \mathbb{N}$ λέγεται **LOOP-υπολογίσιμη** (LOOP-computable) εάν υπάρχει ένα LOOP_{n+m} πρόγραμμα

$$\pi(x_1, \ldots, x_n, \ldots, x_{n+m})$$

και ένα $i \leqslant m + n$ έτσι ώστε για κάθε $a_1, a_2, \ldots, a_n \in \mathbb{N}$:

$$f(a_1, a_2, \ldots, a_n) = S_{d,i}(\pi)[a_1, \ldots, a_n, 0, \ldots, 0]\,.$$

Πρωταρχικές αναδρομικές συναρτήσεις Ι

Ορισμός

Η κλάση $\mathcal{P}$ των **πρωταρχικών αναδρομικών** συναρτήσεων είναι η μικρότερη κλάση συναρτήσεων που:

1. περιέχει τις εξής **αρχικές συναρτήσεις**: S, P, Z, U_i^n (για όλα τα n και $i \leqslant n$) και
2. είναι κλειστή ως προς τα σχήματα της **σύνθεσης** και της **πρωταρχικής αναδρομής**.

Πρωταρχικές αναδρομικές συναρτήσεις II

Εξηγήσεις:
$S(x) = x + 1$, $P(x + 1) = x$, $P(0) = 0$, $Z(x) = 0$,
$U_i^n(x_1, x_2, \ldots, x_n) = x_i$, $1 \leqslant i \leqslant n$

Σύνθεση: $f(x) = h(g(x))$
γενικά $f(x_1, x_2, \ldots, x_n) = h(g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n))$

Πρωταρχική Αναδρομή: $\begin{cases} f(0) = C \\ f(Sy) = h(y, f(y)) \end{cases}$

Γενικά:

$$\begin{cases} f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n) \\ f(x_1, x_2, \ldots, x_n, Sy) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)) \end{cases}$$

Πρωταρχικές αναδρομικές συναρτήσεις III

Παραδείγματα:

1. $\begin{cases} add(x,0) = U_1^1(x) \\ add(x,Sy) = h(x,y,add(x,y)) \end{cases}$ όπου $h(x,y,z) = S(U_3^3(x,y,z))$
2. $\begin{cases} mult(x,0) = Z(x) \\ mult(x,Sy) = h(x,y,mult(x,y)) \end{cases}$
   όπου $h(x,y,z) = add(U_1^3(x,y,z), U_3^3(x,y,z))$
3. $mult2(x) = mult(S(S(Z(x))), U_1^1(x))$
4. $\begin{cases} pow2(0) = 1 \\ pow2(Sy) = h(y,pow2(y)) \end{cases}$ όπου $h(y,z) = mult2(U_2^2(y,z))$
5. Αν η $g(x,y)$ είναι πρωταρχικά αναδρομική, τότε είναι και η f:
   $f(x) = g(x,x) = g(U_1^1(x), U_1^1(x))$ (ταυτοποίηση ορισμάτων)
6. Αν η $g(x,y)$ είναι πρωταρχικά αναδρομική, τότε είναι και η f:
   $f(x,y) = g(y,x) = g(U_2^2(x,y), U_1^2(x,y))$ (εναλλαγή ορισμάτων)
7. $abs(x-y) = (x \dot{-} y) + (y \dot{-} x)$

Πρωταρχικές αναδρομικές συναρτήσεις IV

8. προσημοσυναρτήσεις $(sg, \overline{sg})$
9. $eq(x, y) = \overline{sg}(abs(x - y))$

Παρατήρηση

Οι sg, $\overline{sg}$ και eq είναι χαρακτηριστικές συναρτήσεις.

Ορισμός

Μια σχέση $R \subseteq \mathbb{N}^n$ είναι πρωταρχική αναδρομική, αν η χαρακτηριστική συνάρτηση χ_R είναι πρωταρχική αναδρομική, όπου

$$\chi_R(x_1, \ldots, x_n) = \begin{cases} 1, & \text{αν } (x_1, \ldots, x_n) \in R \\ 0, & \text{αλλιώς} \end{cases}$$

Άλλα είδη αναδρομής Ι

Ακολουθία Fibonacci:

$$\begin{cases} f(0) = 1 \\ f(1) = 1 \\ f(SSn) = f(n) + f(Sn) \end{cases}$$

Αυτό το σχήμα είναι αναδρομή αλλά δεν είναι πρωταρχική αναδρομή!

Αμοιβαία (Mutual) Πρωταρχική Αναδρομή

$$\begin{cases} f_1(0) = 1 \\ f_1(Sn) = f_1(n) + f_2(n) \end{cases} \qquad \begin{cases} f_2(0) = 0 \\ f_2(Sn) = f_1(n) \end{cases}$$

$$f_1 : 1, 1, 2, 3, 5, 8, \ldots \qquad f_2 : 0, 1, 1, 2, 3, 5, \ldots$$

Άλλα είδη αναδρομής II

Γενικά: για $j = 1, \ldots, m$:

$$\begin{cases} f_j(x_1, x_2, \ldots, x_n, 0) & = g_j(x_1, \ldots, x_n) \\ f_j(x_1, x_2, \ldots, x_n, Sy) & = h_j(x_1, \ldots, x_n, y, f_1(x_1, x_2, \ldots, x_n, y), \ldots \\ & \qquad , f_m(x_1, x_2, \ldots, x_n, y)) \end{cases}$$

Λήμμα

Αν όλες οι συναρτήσεις g_j και h_j είναι πρωταρχικές αναδρομικές, τότε και οι συναρτήσεις f_j, όπως ορίζονται από το παραπάνω σχήμα, είναι επίσης πρωταρχικές αναδρομικές.

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων Ι

Θεώρημα

Κάθε πρωταρχική αναδρομική συνάρτηση είναι LOOP-υπολογίσιμη.

Απόδειξη: Με επαγωγή στη δομή της συνεπαγωγικής ακολουθίας των πρωταρχικών αναδρομικών συναρτήσεων.

Παρατήρηση

Η επαγωγική απόδειξη ιδιότητας σε επαγωγικό πεδίο συνεπάγεται:

1. απόδειξη της ιδιότητας για τα αρχικά στοιχεία
2. απόδειξη της ιδιότητας για "νέα" στοιχεία που δημιουργούνται με πράξεις κλεισίματος και με "παλιά" στοιχεία για τα οποία υποθέτουμε ότι *έχουν* την ιδιότητα.

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων II

1. **Αρχικές συναρτήσεις**

αρχικές συναρτήσεις	Loop-πρόγραμμα	μεταβλητή εξόδου
S	$x := \text{succ } x$	x
P	$x := \text{pred } x$	x
Z	$x := 0$	x
U_i^n	κενό πρόγραμμα	x_i

2. **Σχήματα κλεισίματος**
   1. **Σύνθεση**: Έστω ότι δίνονται τα $m+1$ προγράμματα για:
      - $y_1 := g_1(x_1, x_2, \ldots, x_n)$
      - $\ldots$
      - $y_m := g_m(x_1, x_2, \ldots, x_n)$ και
      - $z := h(y_1, y_2, \ldots, y_n)$.

      Τότε η f που ορίζεται με σύνθεση από τις g_1, g_2, ..., g_m και h, με το σχήμα δηλαδή:

      $$f(x_1, x_2, \ldots, x_n) = h(g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n))$$

      μπορεί να υπολογιστεί με το εξής πρόγραμμα:
      "$y_1 := g_1(x_1, x_2, \ldots, x_n)$"; $\ldots$; "$y_m := g_m(x_1, x_2, \ldots, x_n)$"; "$z := h(y_1, y_2, \ldots, y_m)$"

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων III

2. **Πρωταρχική Αναδρομή**: Έστω ότι δίνονται προγράμματα για "$y := g(x_1, \ldots, x_n)$" και "$y := h(x_1, \ldots, x_n, u, y)$".
Τότε η f που ορίζεται με πρωταρχική αναδρομή από τις συναρτήσεις g και h, με το σχήμα δηλαδή:

$$\begin{cases} f(x_1, x_2, \ldots, x_n, 0) = g(x_1, \ldots, x_n) \\ f(x_1, x_2, \ldots, x_n, Sy) = h(x_1, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)) \end{cases}$$

μπορεί να υπολογιστεί με το εξής πρόγραμμα:
$z := x_{n+1}$; "$y := g(x_1, x_2, \ldots, x_n)$"; **for** $u := 1$ **to** z **do** "$y := h(x_1, x_2, \ldots, x_n, u - 1, y)$" **end**

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων Ι

Θεώρημα

Κάθε LOOP-υπολογίσιμη συνάρτηση είναι πρωταρχική αναδρομική.

Απόδειξη: Με επαγωγή στη δομή των προγραμμάτων LOOP:

1. **Αρχικά προγράμματα (αναθέσεις)**

Αρχικό πρόγρ. (αναθέσεις)	Πρωτ. Αναδρ. Συναρτ.
$y := x_i$ (μετ. εξόδου y)	$U_i^n(x_1, x_2, \ldots, x_n)$
$y := \text{succ}\, x_i$	$S(U_i^n(x_1, x_2, \ldots, x_n))$
$y := \text{pred}\, x_i$	$P(U_i^n(x_1, x_2, \ldots, x_n))$
$y := 0$	$Z(U_i^n(x_1, x_2, \ldots, x_n))$
κενό π (μετ. εξόδου x_j)	$U_j^n(x_1, x_2, ..., x_n)$

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων II

2. **Δομές Ελέγχου ροής προγράμματος**
   1. **Παράθεση (;):** Έστω ότι το π_1 υπολογίζει τις πρωταρχικές αναδρομικές συναρτήσεις g_1 (μεταβλητή εξόδου: x_1), g_2 (μεταβλητή εξόδου: x_2), ..., g_n (μεταβλητή εξόδου: x_n) και ότι το π_2 υπολογίζει την πρωταρχική αναδρομική συνάρτηση h (μεταβλητή εξόδου: x_j).
   Τότε το πρόγραμμα $\pi_1; \pi_2$ υπολογίζει:
   $$h_j(g_1(x_1, \ldots, x_n), \ldots, g_n(x_1, \ldots, x_n)) = f(x_1, x_2, \ldots, x_n)$$
   που είναι πρωταρχική αναδρομική λόγω του κλεισίματος ως προς σύνθεση συναρτήσεων.
   2. **βρόχος for:** Έστω ότι το π υπολογίζει τις πρωταρχικές αναδρομικές συναρτήσεις h_1 (μεταβλητή εξόδου: x_1), h_2 (μεταβλητή εξόδου: x_2), ..., h_n (μεταβλητή εξόδου: x_n) και χωρίς βλάβη της γενικότητας ας υποθέσουμε ότι ούτε η μεταβλητή ελέγχου w ούτε το άνω όριο z δεν εμφανίζονται στο π (αλλιώς μετατρέψτε το π σε ισοδύναμο· δες εξήγηση παρακάτω).
   Τότε το πρόγραμμα **for** $w := 1$ **to** z **do** π **end** υπολογίζει τις συναρτήσεις f_j $(1 \leqslant j \leqslant n)$:
   $$\begin{cases} f_j(x_1, x_2, \ldots, x_n, 0) = U_j^n(x_1, \ldots, x_n) \\ f_j(x_1, x_2, \ldots, x_n, Sz) = h_j(f_1(x_1, \ldots, x_n, z), \ldots, f_n(x_1, x_2, \ldots, x_n, z)) \end{cases}$$
   που είναι πρωταρχικές αναδρομές λόγω του κλεισίματος ως προς αμοιβαία πρωταρχική αναδρομή.

Ισοδυναμία πρωταρχικών αναδρομικών και LOOP συναρτήσεων III

Εξήγηση: Αν η μεταβλητή ελέγχου (έστω x_k) και το άνω όριο (έστω x_m) εμφανίζονται στο π, μετατρέπουμε το πρόγραμμα **for** $x_k := 1$ **to** x_m **do** π **end** στο ακόλουθο ισοδύναμο πρόγραμμα π' (ισοδύναμο πρόγραμμα ως προς τις μεταβλητές x_1, x_2, ..., x_n, του π): $x_k := 1$; $z := x_m$; **for** $w := 1$ **to** z **do** (* w, z νέες μεταβλητές *) π; $x_k := x_k + 1$ end

Βασιζόμενοι τώρα στις προηγούμενες αποδείξεις για την παράθεση προγραμμάτων και το βρόχο **for**, μπορούμε εύκολα να δείξουμε ότι κάθε μια από τις f_j $(1 \leqslant j \leqslant n)$ τις οποίες υπολογίζει το π' είναι πρωταρχική αναδρομή. Επομένως, και κάθε f_j του ισοδύναμου προγράμματος **for** $x_k := 1$ **to** x_m **do** π **end** είναι πρωταρχική αναδρομική.

Υπολογισιμότητα

Θεώρημα

Υπάρχουν "υπολογίσιμες" συναρτήσεις που δεν είναι πρωταρχικές αναδρομικές.

Απόδειξη: Διαγωνιοποίηση.

- Μηχανιστική απαρίθμηση πρωταρχικών αναδρομικών συναρτήσεων: φ_0, φ_1, φ_2, ...
- Ορίζουμε: $f(x) = \varphi_x(x) + 1$
  Η f είναι "υπολογίσιμη".
- Έστω ότι η f είναι πρωταρχική αναδρομική, άρα εμφανίζεται στην παραπάνω μηχανιστική απαρίθμηση. Έστω π.χ. ότι ένας δείκτης της f είναι y, δηλαδή $\varphi_y = f$. Εφάρμοσε την f σε όρισμα y:

$$\varphi_y(y) = f(y) = \varphi_y(y) + 1 \qquad \text{Άτοπο}$$

□

Υπολογισιμότητα

Παράδειγμα μιας **ολικής** (ορισμένης για όλους τους φυσικούς αριθμούς) υπολογίσιμης συνάρτησης που όμως δεν είναι πρωταρχική αναδρομική είναι η εξής συνάρτηση f:

$$\begin{cases} f(x,y,0) = Sy \\ f(x,0,1) = x, \quad f(x,0,2) = 0, \quad f(x,0,SSSn) = 1 \\ f(x,Sy,Sn) = f(x,f(x,y,Sn),n) \end{cases}$$

Η παραπάνω συνάρτηση f συγγενεύει με την περίφημη συνάρτηση του Ackermann A:

$$\begin{cases} A(0,n) = n+1 \\ A(m,0) = A(m-1,1) \\ A(m,n) = A(m-1,A(m,n-1)) \end{cases}$$

Ανάγκη να επεκταθούμε απο ολικές σε μερικές συναρτήσεις

Ο τρόπος να αποφύγουμε την αντίφαση ($\varphi_x(x) = \varphi_x(x) + 1$) της διαγωνιοποίησης είναι να επιτρέψουμε **μερικές** (partial) συναρτήσεις, δηλαδή συναρτήσεις που δεν είναι κατά ανάγκη ορισμένες για όλους τους φυσικούς αριθμούς $\mathbb{N}$.

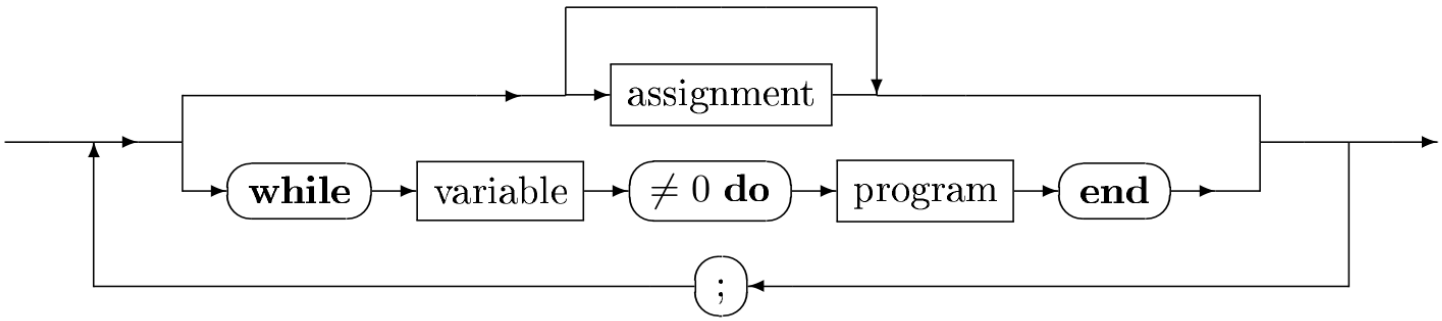
Συνήθως χρησιμοποιούμε τους ακόλουθους συμβολισμούς όταν η συνάρτηση f δεν είναι ορισμένη για το όρισμα x:

> *Η f* **αποκλίνει** *(diverges) για το x, $f(x)\uparrow$, ή ακόμα το πρόγραμμα για την f δεν σταματάει.*

Τα σχήματα σύνθεσης και πρωταρχικής αναδρομής γενικεύονται κατα προφανή τρόπο.

Προγράμματα WHILE και μερικές αναδρομικές συναρτήσεις I

- Οι assignments ακριβώς ίδιες όπως στη γλώσσα LOOP.
- program:



Προγράμματα WHILE και μερικές αναδρομικές συναρτήσεις II

Όπως είναι γνωστό, είναι δυνατόν η εκτέλεση ενός προγράμματος WHILE να μην σταματάει ποτέ.

Παράδειγμα

Η $\sqrt{x} : \mathbb{N} \to \mathbb{N}$, ως μερική συνάρτηση, ορίζεται (σταματάει) μόνο αν το x είναι τέλειο τετράγωνο:
$y := 0$; $z := \mathrm{abs}(x - y^2)$; **while** $z \neq 0$ **do** $y := y + 1$; $z := \mathrm{abs}(x - y^2)$ **end**

Σημασιολογία για προγράμματα WHILE και ακολούθως η έννοια της WHILE-υπολογίσιμης συνάρτησης μπορούν να οριστούν με παρόμοιο τρόπο, όπως και για τα LOOP-προγράμματα. Η κλάση των WHILE-υπολογίσιμων συναρτήσεων συμπεριλαμβάνει και όλες τις LOOP-υπολογίσιμες συναρτήσεις.

μ-σχήμα

Θα εισαγάγουμε τώρα ένα νέο σχήμα, το **μ-σχήμα** ή σχήμα **απεριόριστης ελαχιστοποίησης** (unbounded minimization).

Παράδειγμα

$\sqrt{x} = \mu y[\text{abs}(x - y^2) = 0]$, δηλαδή το μικρότερο y, ώστε $\text{abs}(x - y^2) = 0$, αν υπάρχει τέτοιο y, ειδάλλως η τιμή δεν είναι ορισμένη.

$$\textit{Γενικώς: } f(x_1, \ldots, x_n) = \mu y[h(x_1, \ldots, x_n, y) = 0].$$

Η f μπορεί να μην είναι ορισμένη για δύο λόγους: είτε η h δεν είναι ποτέ $= 0$, είτε η h δεν είναι κάπου ορισμένη πριν να βρεθεί ένα y για το οποίο $h = 0$.

Μερικές αναδρομικές συναρτήσεις

Ορισμός

Η κλάση $\mathcal{PR}$ των **μερικών αναδρομικών συναρτήσεων** (partial recursive functions) είναι η μικρότερη κλάση που:

α) περιλαμβάνει τις **αρχικές συναρτήσεις**: S, P, Z, U_i^n

β) είναι κλειστή ως προς τη **σύνθεση**, την **πρωταρχική αναδρομή** και το μ-**σχήμα**.

Μερικές αναδρομικές συναρτήσεις Ι

Θεώρημα

Μια μερική συνάρτηση είναι WHILE-υπολογίσιμη ανν είναι μερική αναδρομική.

Απόδειξη (σκελετός):
Με επαγωγή, παρομοίως με τις προηγούμενες αποδείξεις της ισοδυναμίας των LOOP-υπολογίσιμων και των πρωταρχικών αναδρομικών συναρτήσεων.

Πρόσθετες ιδέες:

$\Leftarrow$: $f(x_1, \ldots, x_n) = \mu z[h(x_1, \ldots, x_n, z) = 0]$
μπορεί να υπολογιστεί με το πρόγραμμα:
$z := 0$; "$y := h(x_1, \ldots, x_n, z)$"; **while** $y \neq 0$ **do** $z := \mathrm{succ}\, z$; "$y := h(x_1, \ldots, x_n, z)$" **end**

Μερικές αναδρομικές συναρτήσεις II

$\Rightarrow$: **while** $x_k \neq 0$ **do** π **end**
υπολογίζει την $f_i(x_1, \ldots, x_n, v(x_1, \ldots, x_n))$ [σύνθεση], όπου για $1 \leqslant i \leqslant n$:

$$\begin{cases} f_i(x_1, \ldots, x_n, 0) = U_i^n(x_1, \ldots, x_n) [\text{αμοιβαία πρωταρχική αναδρομή}] \\ f_i(x_1, \ldots, x_n, Sz) = h_i(f_1(x_1, \ldots, x_n, z), \ldots, f_n(x_1, \ldots, x_n, z)) \end{cases}$$

$$[z = \text{αριθμός επαναλήψεων βρόχου}]$$

και $v(x_1, \ldots, x_n) = \mu z[f_k(x_1, \ldots, x_n, z) = 0]$

Αναδρομικές Συναρτήσεις και Σχέσεις Ι

Ορισμός

Η ολική συνάρτηση $h(\vec{x}, y)$ είναι **κανονική** (regular):

$$\forall \vec{x}\, \exists y\, h(\vec{x}, y) = 0$$

Παρατήρηση

Η $f(\vec{x}) = \mu y[h(\vec{x}, y) = 0]$ είναι ορισμένη για κάθε $\vec{x}$ για μια κανονική συνάρτηση h.

Ορισμός

$\mathcal{R}$: η μικρότερη κλάση συναρτήσεων που

α) περιλαμβάνει τις S, P, Z, U_i^n, και

β) είναι κλειστή ως προς τη σύνθεση, την πρωταρχική αναδρομή και το μ-σχήμα (εφαρμοζόμενο μόνο) σε κανονικές συναρτήσεις.

Αναδρομικές Συναρτήσεις και Σχέσεις II

Θεώρημα

Η $\mathcal{R}$ περιλαμβάνει ακριβώς εκείνες τις συναρτήσεις της $\mathcal{PR}$ που είναι ολικές.

Απόδειξη.

Έπεται από το **Θεώρημα Κανονικής Μορφής Kleene** (Kleene Normal Form Theorem) □

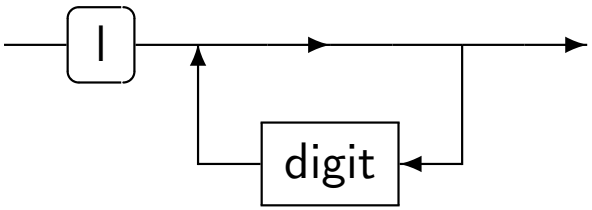
Ορισμός

Μια **σχέση** (ή ένα **σύνολο**) είναι **αναδρομική** αν η χαρακτηριστική της (ή του) συνάρτηση είναι (ολική) αναδρομική.

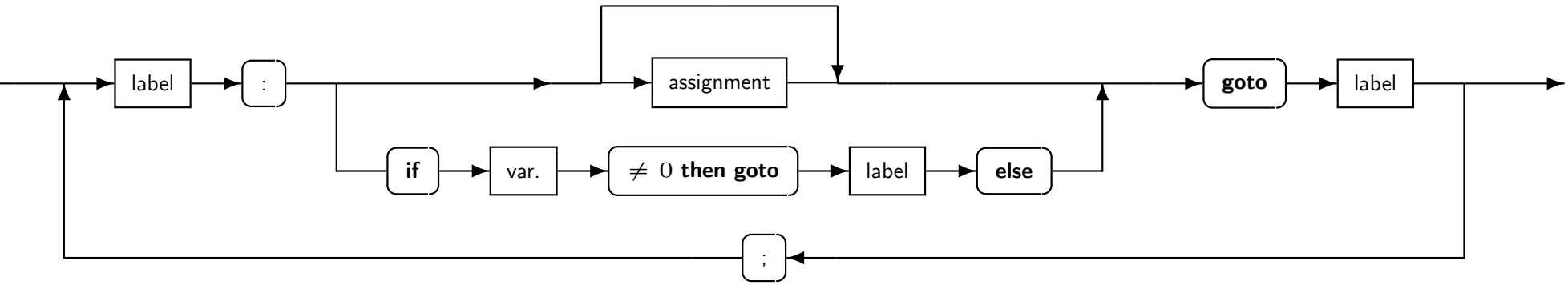
Η γλώσσα GOTO I

Μία περιορισμένη γλώσσα GOTO:

- μεταβλητές και εντολές ανάθεσης όπως στην WHILE.
- label



- program



Η γλώσσα GOTO II

Παρατηρήσεις:

- Οι εντολές των προγραμμάτων GOTO είναι αριθμημένες με διαδοχικές labels αρχίζοντας από το l_0.
- Η υπολογιστική εκτέλεση ενός GOTO προγράμματος σταματάει όταν εκτελεστεί ένα **goto** l_i και καμία εντολή δεν είναι αριθμημένη με label l_i.

Άλλα *ισοδύναμα* υπολογιστικά μοντέλα αποτελούν π.χ. μία γλώσσα assembly, ή μία γλώσσα συναρτησιακή όπως η LISP.

Όλα αυτά τα μοντέλα είναι **μηχανιστικά ισοδύναμα**. Μερικά κοινά χαρακτηριστικά αυτών των μοντέλων είναι π.χ.:

- **ντετερμινιστική** υπολογιστική ακολουθία σε **διακριτά βήματα**.
- **πεπερασμένο σύνολο εντολών** που μπορούν να εκτελεστούν από κάποιον επεξεργαστή (άνθρωπο ή μη).

Μηχανές Turing (T.M) I

Οι βασικές λειτουργίες μιας **TM** είναι:

- Διάβασε το περιεχόμενο του τρέχοντος κουτιού
- Γράψε 1 ή 0 στο τρέχον κουτάκι
- Κάνε τρέχον το αμέσως αριστερότερο ή το αμέσως δεξιότερο κουτάκι

Η TM έχει ένα πεπερασμένο αριθμό εσωτερικών **καταστάσεων (internal states)**:

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

Πρόγραμμα μιας TM: είναι ένα σύνολο από **τετράδες** της μορφής $\langle q_i, e, d, q_j \rangle$ όπου $q_i, q_j \in Q, e \in \Sigma$, $d \in A = \Sigma \bigcup \{L, R\}$ με τον εξής συναρτησιακό (ντετερμινιστικό) περιορισμό: Για κάθε $\langle q_i, e \rangle$ υπάρχει το πολύ ένα $\langle d, q_j \rangle$ έτσι ώστε η τετράδα $\langle q_i, e, d, q_j \rangle$ να ανήκει στο πρόγραμμα, δηλαδή πρόκειται για μια **συνάρτηση μετάβασης** (*transition function*) $\delta : Q \times \Sigma \rightarrow A \times Q$.

Παραδείγματα TM I

Παράδειγμα:

$\Sigma = \{0, 1\}$, $Q = \{q_0, q_1\}$
Πρόγραμμα: $\{\langle q_0, 1, 0, q_1\rangle, \langle q_1, 0, R, q_0\rangle\}$
Είσοδος: πεπερασμένος αριθμός διαδοχικών "1", όλα τα υπόλοιπα κουτάκια περιέχουν "0".
Αρχική κατάσταση: q_0
Αρχική θέση κεφαλής (τρέχον κουτάκι): το αριστερότερο "1"

Παραδείγματα TM II

Παράδειγμα υπολογιστικής ακολουθίας, δηλαδή νόμιμης ακολουθίας από στιγμιότυπα:

$$\ldots\ 0\ 0\ \underset{\uparrow}{1}\ 1\ 1\ 0\ 0$$
$$\qquad\qquad q_0$$

$$\ldots\ 0\ 0\ \underset{\uparrow}{0}\ 1\ 1\ 0\ 0$$
$$\qquad\qquad q_1$$

$$\ldots\ 0\ 0\ 0\ \underset{\uparrow}{1}\ 1\ 0\ 0$$
$$\qquad\qquad\quad q_0$$

$$\ldots\ 0\ 0\ 0\ q_1\ 0\ 1\ 0\ 0$$

$$\ldots\ 0\ 0\ 0\ 0\ q_0\ 1\ 0\ 0$$

$$\ldots\ 0\ 0\ 0\ 0\ q_1\ 0\ 0\ 0$$

$$\ldots\ 0\ 0\ 0\ 0\ 0\ q_0\ 0\ 0\ 0 \qquad \textit{halt}$$

Παραδείγματα TM III

Σε κάθε TM μπορούμε να αντιστοιχίσουμε μια μερική συνάρτηση από το $\mathbb{N}$ στο $\mathbb{N}$. Η είσοδος $n \in \mathbb{N}$ παριστάνεται με $\underline{n+1}$ διαδοχικά 1 (έτσι ο αριθμός 0 παριστάνεται με 1). Σαν αρχικό στιγμιότυπο έχουμε την κεφαλή (τρέχον κουτάκι) να δείχνει στο αριστερότερο 1 και να βρίσκεται στην κατάσταση q_0. Σαν έξοδο λαμβάνουμε το συνολικό αριθμό από 1 που βρίσκεται στην ταινία, όταν και εάν η μηχανή σταματήσει.

Παραδείγματα TM IV

Παράδειγμα

Να κατασκευαστεί μια TM που υπολογίζει το $2 * x$.

Ιδέα:

	...	0	0	~~1~~	~~1~~	1	1	0		
						↑			1	1

Η TM θα εργάζεται σύμφωνα με το παρακάτω πρόγραμμα (το οποίο γράφεται πάντα πρώτα σε άτυπη γλώσσα υψηλού επιπέδου):

αρχικοποίηση; (διαγραφή του πρώτου 1)
while *είσοδος<>0* **do**
  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

Παραδείγματα TM V

$< q_0$	1	0	$q_1 >$	
$< q_1$	0	R	$q_2 >$	
$< q_2$	1	0	$q_3 >$	| halt για $< q_2 \;\; 0 >$
$< q_3$	0	R	$q_4 >$	
$< q_4$	1	R	$q_4 >$	
$< q_4$	0	R	$q_5 >$	
$< q_5$	1	R	$q_5 >$	
$< q_5$	0	1	$q_6 >$	
$< q_6$	1	R	$q_6 >$	
$< q_6$	0	1	$q_7 >$	
$< q_7$	1	L	$q_7 >$	
$< q_7$	0	L	$q_8 >$	
$< q_8$	1	L	$q_8 >$	
$< q_8$	0	R	$q_2 >$	

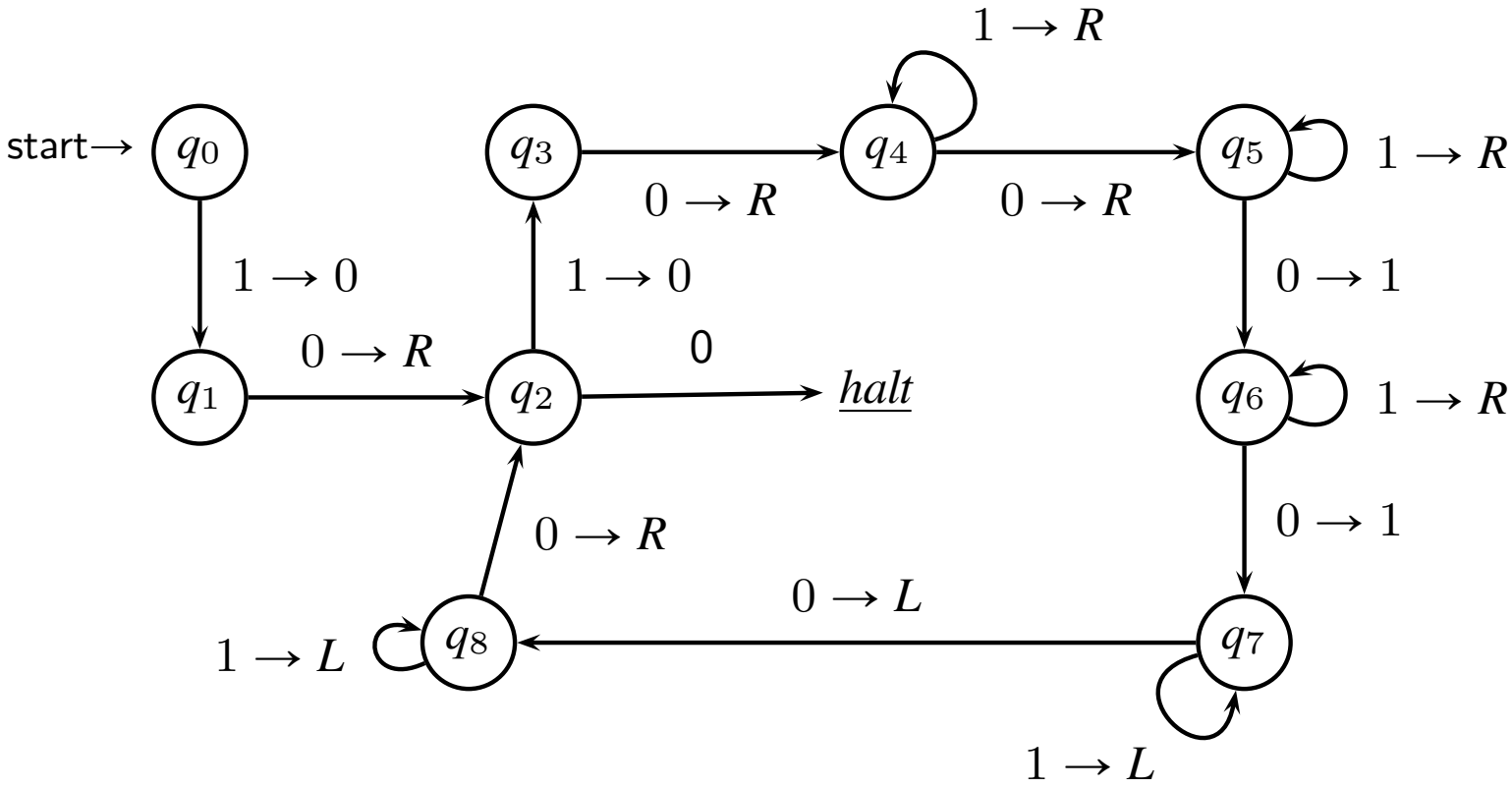
Πίνακας: Πρόγραμμα TM παραδείγματος σε μορφή τετράδων

Παραδείγματα TM VI

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
0		R/q_2		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

Πίνακας: TM σε μορφή πίνακα

Παραδείγματα TM VII



Σχήμα: TM σε μορφή διαγράμματος καταστάσεων

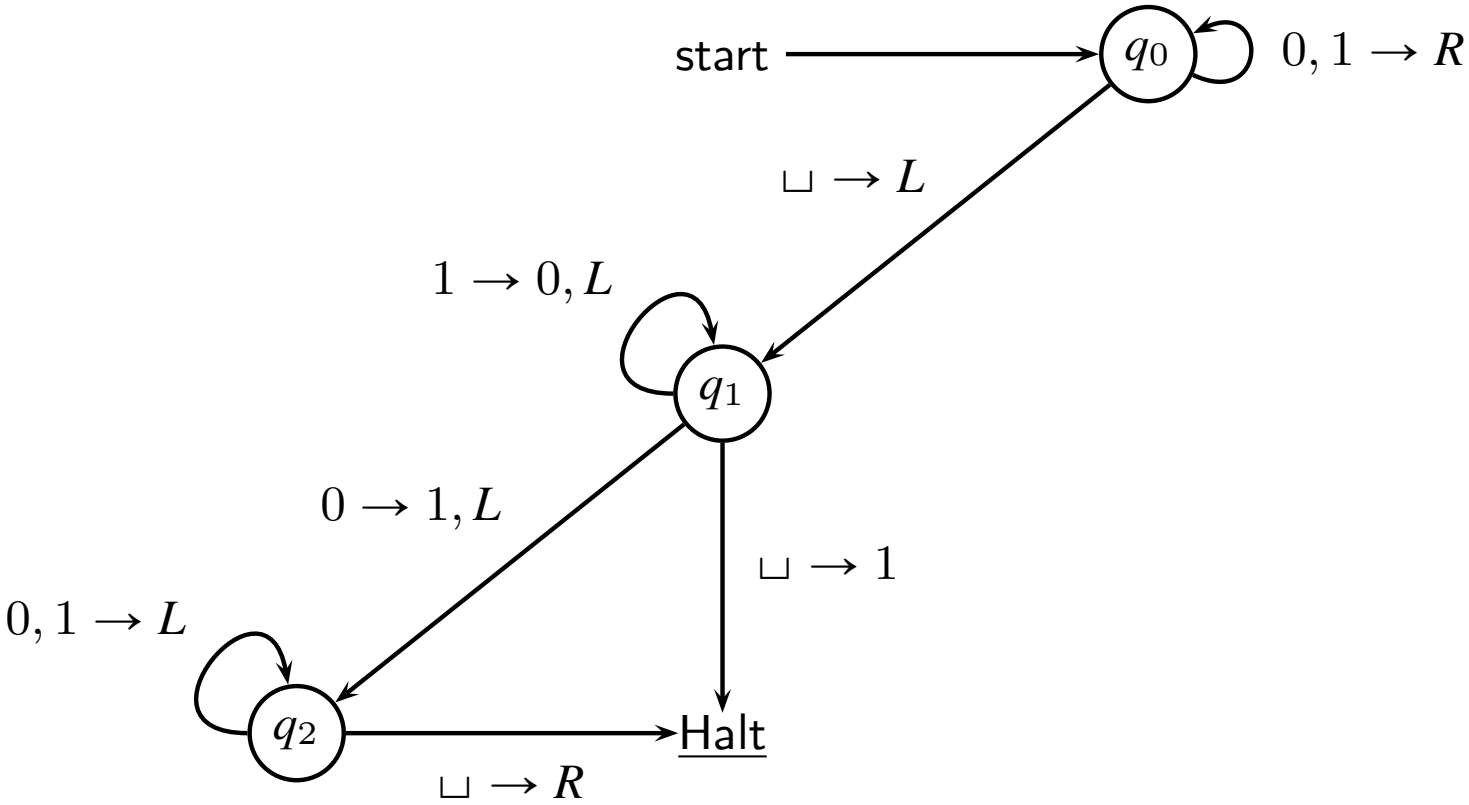
Παραδείγματα TM I

Στο επόμενο παράδειγμα το αλφάβητο είναι $\Sigma = \{0, 1, \sqcup\}$.
Η είσοδος και η έξοδος κωδικοποιούνται στο **δυαδικό σύστημα**.
Παράδειγμα: Να κατασκευαστεί μια TM που υπολογίζει το $x + 1$.

Η TM θα εργάζεται σύμφωνα με το παρακάτω πρόγραμμα:

Κάνε τρέχον το κουτάκι με τελευταίο σύμβολο της εισόδου x;
repeat
  Αν τρέχον κουτάκι έχει $\sqcup$, γράψε 1 και σταμάτα;
  Αν τρέχον κουτάκι έχει 1, γράψε 0, κάνε τρέχον το αμέσως αριστερότερο κουτάκι, και κρατούμενο := 1;
  Αν τρέχον κουτάκι έχει 0, γράψε 1, κάνε τρέχον το αμέσως αριστερότερο κουτάκι, και κρατούμενο := 0;
until *κρατούμενο = 0;*
Κάνε τρέχον το κουτάκι με πρώτο σύμβολο του $x + 1$ και σταμάτα;

Παραδείγματα TM II



Σχήμα: TM σε μορφή διαγράμματος καταστάσεων

Παραδείγματα TM III

Παράδειγμα λειτουργίας με είσοδο $x = 1011$:

$$
\begin{array}{ccccccccc}
(q_0, \underline{1}011) & \vdash & (q_0, 1\underline{0}11) & \vdash & (q_0, 10\underline{1}1) & \vdash & (q_0, 101\underline{1}) & \vdash \\
(q_0, 1011\underline{\sqcup}) & \vdash & (q_1, 101\underline{1}) & \vdash & (q_1, 10\underline{1}0) & \vdash & (q_1, 1\underline{0}00) & \vdash \\
(q_2, \underline{1}100) & \vdash & (q_2, \underline{\sqcup}1100) & \vdash & (\mathsf{HALT}, \underline{1}100) & & &
\end{array}
$$

Παραδείγματα TM IV

$< q_0$	0	q_0	0	R $>$
$< q_0$	1	q_0	1	R $>$
$< q_0$	$\sqcup$	q_1	$\sqcup$	L $>$
$< q_1$	0	q_2	1	L $>$
$< q_1$	1	q_1	0	L $>$
$< q_1$	$\sqcup$	Halt	1	S $>$
$< q_2$	0	q_2	0	L $>$
$< q_2$	1	q_2	1	L $>$
$< q_2$	$\sqcup$	Halt	$\sqcup$	R $>$

Πίνακας: Πρόγραμμα TM

	0	1	$\sqcup$
q_0	(q_0, 0, R)	(q_0, 1, R)	(q_1, $\sqcup$, L)
q_1	(q_2, 1, L)	(q_1, 0, L)	(Halt, 1, S)
q_2	(q_2, 0, L)	(q_2, 1, L)	(Halt, $\sqcup$, R)

Πίνακας: TM σε μορφή πίνακα

Αντιστοιχία προγραμμάτων WHILE και TM

WHILE - πρόγραμμα	αντίστοιχη TM
succ(x)	κενή TM
pred(x)	$\{q_0 1 0 q_1,\ q_1 0 R q_2,\ q_2 1 0 q_2\}$
zero(x)	$\{q_0 1 0 q_1,\ q_1 0 R q_0\}$
;	"ένωση" των δύο TM μέ αναπροσαρμογή των ονομάτων των καταστάσεων ώστε να είναι διαφορετικές και αναπροσαρμογή της παραστασης εξόδου της πρώτης TM που θα χρησημοποιηθεί ώς είσοδος για τη δέυτερη TM.
while	παρόμοια (βλ. επίσης το παράδειγμα παραπάνω)

Ντετερμινισμός και μη I

Ντετερμινιστική μηχανή Turing (DTM):
$\delta : Q \times \Sigma \to Q \times A$, όπου $A = \Sigma \cup \{L, R\}$

Μη-Ντετερμινιστική μηχανή Turing (NTM):
$\delta : Q \times \Sigma \to \text{Pow}(Q \times A)$, όπου $A = \Sigma \cup \{L, R\}$

Η τριάδα που αποτελείται από τα **περιεχόμενα** της ταινίας της TM, την **θέση της κεφαλής** πάνω στην ταινία και την **τρέχουσα κατάσταση** της TM ονομάζεται **configuration** (στιγμιαία περιγραφή) της TM.

Ντετερμινισμός και μη II

- Σε μία **DTM** ο υπολογισμός είναι μία **γραμμική ακολουθία από configurations** (σε κάθε βήμα υπάρχει ακριβώς **μία** επόμενη νόμιμη configuration).
- Σε μία **NTM**, όμως, ο υπολογισμός περιγράφεται με ένα **δένδρο από configurations**, αφού σε κάθε βήμα είναι δυνατό να προκύψουν περισσότερες από μία επόμενες νόμιμες configurations.

Θα λέμε ότι μία συμβολοσειρά γίνεται **αποδεκτή** από μία NTM, αν γίνεται αποδεκτή **τουλάχιστον από ένα** υπολογιστικό μονοπάτι του δένδρου.

Θεώρημα

Για κάθε μη-ντετερμινιστική μηχανή Turing (NTM) υπάρχει **ισοδύναμη** *ντετερμινιστική μηχανή Turing (DTM).*

Παραλλαγές ΤΜ

Παραλλαγές Μηχανών Turing που έχουν την ίδια υπολογιστική δυνατότητα, όχι όμως και αποδοτικότητα (*efficiency*) είναι:

- πολλές ταινίες, μνήμη πλέγματος (grid memory), μνήμη περισσοτέρων διαστάσεων
- μεγαλύτερο Σ
- πολλές παράλληλες κεφαλές
- μη ντετερμινιστικές μεταβάσεις
- μίας κατευθύνσεως, απείρου μήκους ταινία
- εγγραφή και κίνηση της κεφαλής σε κάθε βήμα

Υπολογιστικά Μοντέλα Ι

Μερικά υπολογιστικά μοντέλα είναι τα εξής:

- προγράμματα Pascal
- προγράμματα Pascal χωρίς αναδρομή (αφαίρεση αναδρομής με χρήση στοίβας)
- προγράμματα Pascal χωρίς αναδρομή και χωρίς άλλους τύπους δεδομένων εκτός από τους φυσικούς αριθμούς (επιτυγχάνεται με κωδικοποιήσεις)
- προγράμματα WHILE (μόνη δομή ελέγχου το WHILE)
- προγράμματα GOTO και IF
- Assembler-like RAM (random access machine), URM (universal register machine)
- SRM (single register machine) ένας καταχωρητής
- Μηχανή Turing (πρόσβαση μόνο σε ένα κουτάκι "cell" της ταινίας κάθε φορά)
- παραλλαγές από μηχανές Turing
- Thue: κανόνες επανεγγραφής (re-writing rules)
- Post: κανονικά συστήματα (normal systems)
- Church: λογισμός λ (λ-calculus)

Υπολογιστικά Μοντέλα II

- Curry: συνδυαστική λογική (combinatory logic)
- Markov: M. αλγόριθμοι
- Kleene: γενικά αναδρομικά σχήματα (general recursive schemes)
- Shepherdson-Sturgis, Elgott: URM, SRM, RAM, RASP
- Σχήματα McCarthy (if ... then ... else ... $\Rightarrow$ LISP)

Υπολογιστικά Μοντέλα III

Τα χαρακτηριστικά των παραπάνω μοντέλων είναι:

- ντετερμινιστική πολυπλοκότητα σε διακριτά βήματα
- πεπερασμένο σύνολο εντολών που εκτελούνται από επεξεργαστή
- απεριόριστη μνήμη

Θεώρημα

f είναι TM υπολογίσιμη ανν

- *f είναι WHILE-υπολογίσιμη*
- *f είναι GOTO-υπολογίσιμη*
- *f είναι PASCAL-υπολογίσιμη*
- *f είναι μερικά αναδρομική (partial recursive)*

Αυτόματο με δύο στοίβες

Αν επεκτείνουμε το αυτόματο στοίβας (PDA) προσθέτοντάς του μία επιπλέον στοίβα, έχουμε το αποκαλούμενο 2-PDA. Αποδεικνύεται ότι τα 2-PDA έχουν υπολογιστική ισχύ ίση με αυτήν των TM.

Ιδέα εξομοίωσης μίας TM από ένα 2-PDA: Στην μία στοίβα του 2-PDA τοποθετούμε τα σύμβολα της ταινίας που βρίσκονται αριστερά της κεφαλής της TM και στην άλλη στοίβα του 2-PDA τα σύμβολα που βρίσκονται δεξιά της κεφαλής (τα σύμβολα που είναι πλησιέστερα στην κεφαλή θα βρίσκονται προς την κορυφή της κάθε στοίβας).

Ιδιότητες κλεισίματος r.e. γλωσσών

Πρόταση

Η κλάση των r.e. γλωσσών είναι κλειστή ως προς ένωση, παράθεση (concatenation), άστρο του Kleene ($*$) και τομή.

Πρόταση

Η κλάση των r.e. γλωσσών δεν είναι κλειστή ως προς συμπλήρωμα.

Recursive and Recursively Enumerable sets I

Ορισμός

Το σύνολο A είναι **αναδρομικό** ανν η χαρακτηριστική του συνάρτηση X_A είναι αναδρομική.

Το σύνολο A είναι **αναδρομικά αριθμήσιμο** (recursively enumerable) ανν $A = \varnothing$ ή $A =$ πεδίο τιμών της f $(A = \text{ran}(f))$ για κάποια αναδρομική συνάρτηση f.

Εξηγήσεις:

- Το σύνολο A είναι αναδρομικό (αποκρίσιμο) αν υπάρχει πρόγραμμα που μπορεί να αποφαίνεται για το ερώτημα: "δεδομένου του αριθμού n, είναι το n στοιχείο του συνόλου A;".
- Το σύνολο A είναι αναδρομικά αριθμήσιμο (καταγράψιμο, r.e.) αν υπάρχει πρόγραμμα που (χωρίς είσοδο) δίνει στην έξοδο όλα τα στοιχεία του συνόλου A (σε οποιαδήποτε σειρά).

Recursive and Recursively Enumerable sets II

Παράδειγμα

- Το $K = \{x \mid \varphi_x(x) \downarrow\}$ δεν είναι αναδρομικό (μη επιλυσιμότητα του προβλήματος τερματισμού).
- Το K είναι r.e. (Μπορούμε να κατασκευάσουμε πρόγραμμα που ξεκινά διαδοχικά και τρέχει παράλληλα με τη μέθοδο "dovetaling" όλα τα προγράμματα **while** και όταν κάποιο από αυτά σταματάει βγάζει στην έξοδο το δείκτη του).
- Το $\overline{K}$ δεν είναι ούτε r.e. (γιατί αν και το K και το $\overline{K}$ ήταν r.e., τότε το K θα ήταν αναδρομικό: λεπτομέρειες αργότερα).

Recursive and Recursively Enumerable sets III

Θεώρημα

1. *Το A είναι r.e.*
2. *αν $A = \mathrm{dom}(f)$ (πεδίο ορισμού), όπου $f \in \mathcal{PR}$*
3. *αν $A = \mathrm{ran}(f)$, όπου $f \in \mathcal{PR}$*
4. *αν $A = \varnothing$ ή $A = \mathrm{ran}(f)$, όπου $f \in \mathcal{P}$*
5. *αν $A = \{x \mid \exists y\, R(x, y)\}$, όπου R αναδρομική*
6. *αν το A είναι **while**-καταγράψιμο*
7. *αν το A είναι Turing-καταγράψιμο*
8. *αν το A είναι (**while**- ή) Turing-αποδεκτό (acceptable), δηλ. μπορεί να αναγνωριστεί από (πρόγραμμα **while** ή από) μηχανή Turing (η μηχανή Turing (πρόγραμμα **while**)) σταματά για κάθε $x \in A$ και δε σταματά για $x \notin A$)*
9. *αν $= graph(f)$, όπου $f \in \mathcal{PR}$*

Recursive and Recursively Enumerable sets IV

Παρατηρήσεις:

- Από το πρώτο *ανν* του θεωρήματος και από μια μηχανιστική απαρίθμηση $\{\varphi_i\}$ όλων των μερικών αναδρομικών συναρτήσεων μπορούμε να λάβουμε μια μηχανιστική απαρίθμηση $\{W_i\}$ όλων των r.e. συνόλων (= πεδίων ορισμού των $\{\varphi_i\}$).

$$W_i = \{x \mid \varphi_i(x) \downarrow\} = \mathrm{dom}(\varphi_i)$$

- $K = \{x \mid \varphi_x(x) \downarrow\} = \{x \mid x \in W_x\}$

Recursive and Recursively Enumerable sets V

Θεώρημα

Το A είναι αναδρομικό ανν το A και το $\overline{A}$ είναι r.e..

Απόδειξη.

"$\Rightarrow$" : Έστω A αναδρομικό, δηλαδή υπάρχει πρόγραμμα π που αποφαίνεται για το A. Κατασκευάζουμε πρόγραμμα που εκτελεί το π επανειλημμένα για να αποφανθεί για όλες τις εισόδους: 0, 1, 2, ... Κατ' αυτόν τον τρόπο μπορούν να απαριθμηθούν το A και το $\overline{A}$.

"$\Leftarrow$" : Έστω A και $\overline{A}$ r.e., δηλ. υπάρχουν δύο προγράμματα π_1, και π_2 που απαριθμούν (καταγράφουν) το A και το $\overline{A}$ αντιστοίχως. Κατασκευάζουμε πρόγραμμα που ελέγχει αν $n \in A$ τρέχοντας εν παραλλήλω το π_1 και το π_2. Το n θα εμφανιστεί κάποτε είτε στην έξοδο του π_1 είτε στην έξοδο του π_2. 'Ετσι τότε θα αποφασίσει το πρόγραμμα αν $n \in A$ ή όχι (φυσικά δεν είναι γνωστό πόσος χρόνος θα χρειαστεί για να απαντηθεί το ερώτημα: "$n \in A$;").

□

Recursive and Recursively Enumerable sets VI

Θεώρημα

Το A είναι αναδρομικό αν $A = \text{ran}(f)$, *όπου f γνησίως αύξουσα αναδρομική συνάρτηση ή το A είναι πεπερασμένο σύνολο.*

Παρατήρηση

Η κλάση των **αναδρομικών σχέσεων** είναι **κλειστή** ως προς:

- όλες τις **λογικές πράξεις** (ένωση, τομή, συμπλήρωμα) και ως προς
- **φραγμένους ποσοδείκτες** (bounded quantification, $\forall\ <, \exists\ <$).

Η κλάση (RE) των **recursively enumerable σχέσεων** είναι **κλειστή** ως προς:

- **ένωση και τομή** και όχι όμως και ως προς συμπλήρωμα.
- **φραγμένο καθολικό ποσοδείκτη** (bounded universal quantification: $\forall\ <$) και **μη φραγμένο υπαρξιακό ποσοδείκτη** (unbounded existential quantification: $\exists$).

Recursive and Recursively Enumerable sets VII

Ορμώμενοι από τις αναγωγές που κάναμε του HP, για παράδειγμα, σε άλλα προβλήματα, εισάγουμε τώρα την έννοια της *αναγωγιμότητας (reducibility)*.

Ορισμός

$A \leqslant_m B$: (A είναι (πολλά-ένα) αναγώγιμο στο B):
$\exists g \in R \; \forall n : (n \in A \leftrightarrow g(n) \in B)$.
Αν επιπλέον η g είναι 1-1, τότε γράφουμε $A \leqslant_1 B$ (το A είναι ένα προς ένα αναγώγιμο στο B).
Αν η συνάρτηση g είναι πολυωνυμικής πολυπλοκότητας μιλάμε για αναγωγή πολυωνυμικού χρόνου και συμβολίζουμε $A \leqslant_m^p B$ (αναγωγή κατά Karp).

Θεώρημα

α) $\leqslant_m$ *είναι ανακλαστική και μεταβατική.*

β) $A \leqslant_m B \Rightarrow \overline{A} \leqslant_m \overline{B}$

γ) $A \leqslant_m B \wedge B$ *αναδρομικό* $\Rightarrow$ *αναδρομικό*
[ή : $A \leqslant_m B \wedge A$ *μη επιλύσιμο* $\Rightarrow B$ *μη επιλύσιμο]*

δ) $A \leqslant_m B \wedge B$ *r.e.* $\Rightarrow A$ *r.e.*

Recursive and Recursively Enumerable sets VIII

Ορισμός

$A \equiv_m B : \; A \leqslant_m B \wedge B \leqslant_m A$

Παρατήρηση

Η $\equiv_m$ είναι σχέση ισοδυναμίας και διαβάζεται "έχουν τον ίδιο *βαθμό (μη) επιλυσιμότητας*" (have the same degree of unsolvability).
Έστω C μια κλάση από σύνολα (π.χ. RE η κλάση των r.e. συνόλων).

Ορισμός

B είναι δύσκολο (hard) στην C ως προς $\leqslant_m$: $\forall A \in C : A \leqslant_m B$

Ορισμός

B είναι πλήρες (complete) στην C ως προς $\leqslant_m$: $B \in C$ και είναι δύσκολο στην C ως προς $\leqslant_m$.

Recursive and Recursively Enumerable sets IX

Παρατήρηση:

Π.χ. αν $C = NP$ έχουμε προβλήματα (ή σύνολα) *πλήρη* στην NP (NP είναι η κλάση προβλημάτων υπολογιστών μη ντετερμινιστικά σε πολυωνυμικό χρόνο) [ως προς αναγωγή πολυωνυμικού χρόνου].

Η χρησιμότητα της έννοιας του NP-πλήρους προβλήματος φαίνεται και από το εξής: Αν ένα NP-πλήρες πρόβλημα A αποδειχθεί ότι επιλύεται με πολυωνυμικού χρόνου ντετερμινιστικό αλγόριθμο, τότε κάθε NP πρόβλημα θα επιλύεται επίσης με πολυωνυμικό (ντετερμινιστικό) αλγόριθμο, αφού ανάγεται πολυωνυμικά στο A.

Recursive and Recursively Enumerable sets X

Θεώρημα

α)

$$\left.\begin{array}{l} \textit{Το } B \textit{ είναι πλήρες στην } RE \\ B \leqslant_m C \\ \textit{Το } C \textit{ είναι r.e.} \end{array}\right\} \Rightarrow C \textit{ πλήρες στην } RE.$$

β) A και B είναι πλήρη στην $RE \Rightarrow A \equiv_m B$

γ) Το K είναι πλήρες στην RE.

Απόδειξη: Άσκηση ...

Μαντεία Ι

Στη συνέχεια θα παρουσιάσουμε προγράμματα ή μηχανές Turing με **μαντεία**.

Επιτρέπουμε στα προγράμματα (ή μηχανές) να εισέρχονται σε κατάσταση ερώτησης (οσοδήποτε συχνά), ενώ υπολογίζουν και να συμβουλεύονται ένα *μαντείο* για το A, δηλαδή μία συσκευή που μπορεί να αποφασίζει την ιδιότητα "στοιχείο του" για το A ($x \in A$ ή όχι;). Αν το μαντείο αποκρίνεται για ένα αναδρομικό σύνολο A, τότε το πρόγραμμα μας (ή TM) υπολογίζει ακριβώς όλες τις υπολογίσιμες συναρτήσεις. Αν όμως το μαντείο αποκρίνεται για άλλα μη επιλύσιμα σύνολα (π.χ. K), τότε το πρόγραμμα μας υπολογίζει συναρτήσεις που είναι *υπολογίσιμες με χρήση μαντείου για το σύνολο A* ή *απλά A-υπολογίσιμες*.

Ορισμός

$\mathcal{PR}^A$ είναι η μικρότερη κλάση συναρτήσεων

1. περιλαμβάνει τις S, P, Z, U_i^n και χ_A
2. είναι κλειστή ως προς σύνθεση, πρωταρχική αναδρομή και μ-σχήμα.

Μαντεία II

Παρόμοια, μπορούμε να επεκτείνουμε την έννοια σε πολλά μαντεία: *(μερική) $\mathcal{C}$-αναδρομική*, όπου $\mathcal{C}$ είναι μία κλάση από συναρτήσεις και/ή σύνολα.

Μαντεία III

Ορισμός

$A \leqslant_T B$ (A ανάγεται κατά Turing στο B):
A είναι B-αναδρομική.

Παρατήρηση: Παρόμοιες ιδιότητες και έννοιες όπως για $\leqslant_m$

- $\leqslant_T$ είναι ανακλαστική και μεταβατική.
- $\equiv_T$: είναι η προκύπτουσα αντίστοιχα σχέση ισοδυναμίας, η Turing ισοδυναμία· Turing βαθμός μη επιλυσιμότητας: $d(A) =$ κλάση ισοδυναμίας του A.
- $d(\varnothing)$: μικρότερος βαθμός, βαθμός των αναδρομικών συνόλων.
- Δεν υπάρχει μεγαλύτερος βαθμός: $\forall A : d(A) < d(A^K)$
- $d(K)$ είναι μεγιστικός βαθμός μεταξύ των αναδρομικά αριθμήσιμων συνόλων.
- Υπάρχουν αναδρομικά αριθμήσιμα σύνολα , τέτοια ώστε δεν ισχύει ούτε $A \leqslant_T B$ ούτε $\leqslant_T A$.
- $\leqslant_T$-hard, $\leqslant_T$-complete. ($\leqslant_T$-δύσκολο, $\leqslant_T$-πλήρες.)
- K είναι $\leqslant_T$-complete στο RE (αναδρομικά αριθμήσιμα σύνολα)

Μαντεία IV

Ορισμός

Μία σχέση R είναι αριθμητική αν ορίζεται στην αριθμητική (δηλαδή την σχεσιακή δομή: $\underline{\mathbb{N}} = \langle \mathbb{N}; <; S; +; *; 0 \rangle$.

Θεώρημα (Gödel)

R είναι r.e. (αναδρομικά αριθμήσιμη) $\Longrightarrow$ R είναι αριθμητική.

Απόδειξη.

Επαγωγή στο $\mathcal{PR}$. □

Η αριθμητική ιεραρχία Ι

$\Sigma_1^0 = \text{RE} = \{S \mid S(\underline{x}) \leftrightarrow \exists \underline{y}, R(\underline{x}, \underline{y}),\ R \text{ αναδρομική}\}$

$\Pi_1^0 = \text{co-RE} = \text{συμπληρώματα συνόλων του RE} = \{S \mid S(\underline{x}) \leftrightarrow \forall \underline{y} R(\underline{x}, \underline{y}),\ R \text{ αναδρομική}\}$

και γενικά:

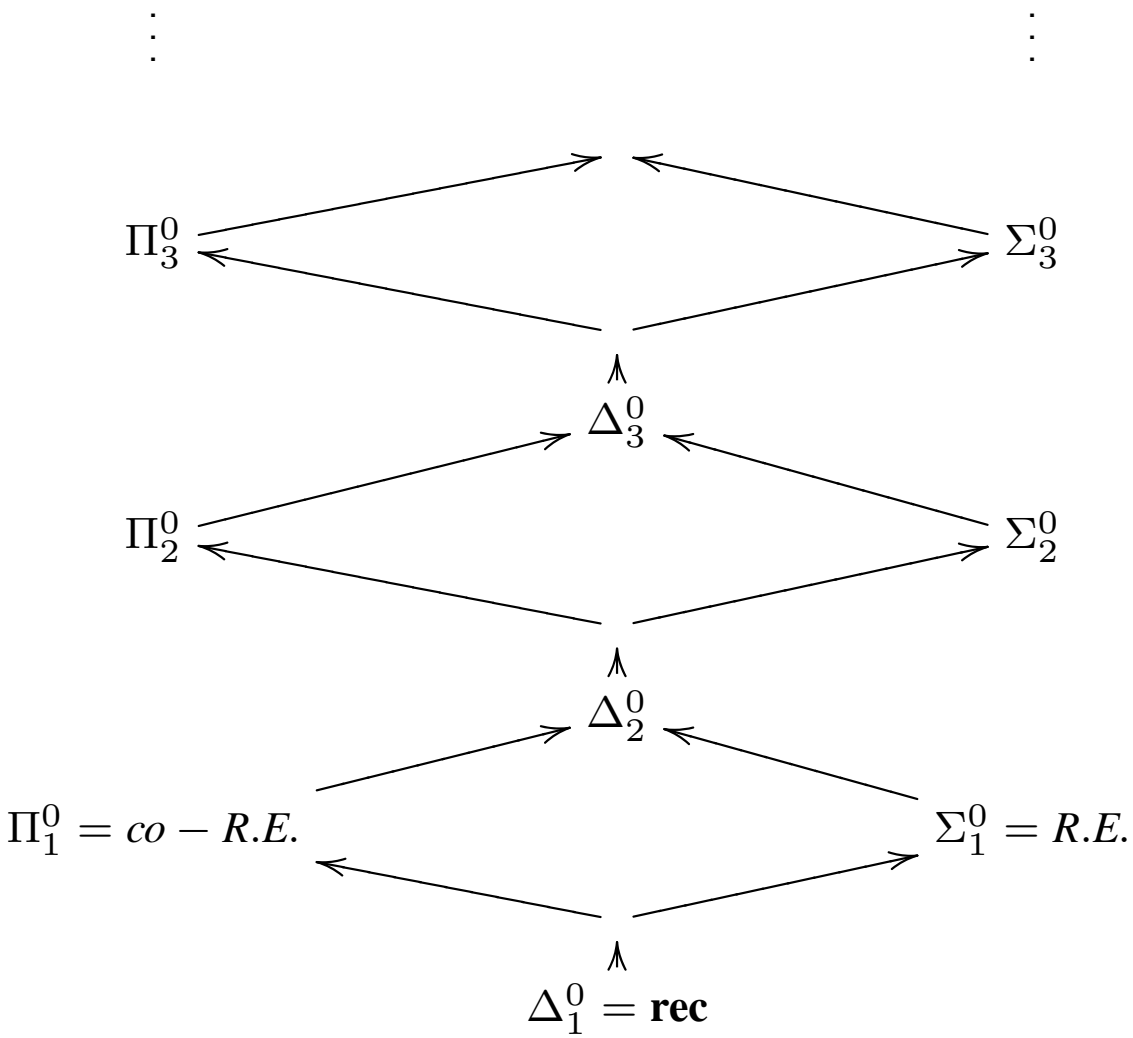
$\Sigma_{n+1}^0 = \Pi_n^0\text{-RE} = \{S \mid S(\underline{x}) \leftrightarrow \exists \underline{y}, R(\underline{x}, \underline{y}), R \in \Pi_n^0\}$

$\Pi_{n+1}^0 = \text{co-}\Sigma_{n+1}^0 = \text{συμπληρώματα συνόλων του } \Sigma_{n+1}^0$

$\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$

Παράδειγμα: $\forall x_1 \forall x_2 \exists x_3 \exists x_4 \exists x_5 \forall x_6 \exists x_7 \forall x_8 \forall x_9 R(\underline{x})$ στο Π_5^0 αν R: αναδρομική.

Η αριθμητική ιεραρχία II



Σχήμα: Αριθμητική ιεραρχία

Η αριθμητική ιεραρχία III

Θεώρημα

Οι εγκλεισμοί στην αριθμητική ιεραρχία είναι γνήσιοι.

Θεώρημα

Μία σχέση R είναι αριθμητική ανν υπάρχει k τ.ω. $R \in \Sigma_k^0$.

Αποκρίσιμα και μη προβλήματα Ι

Πρόταση

Έστω G γραμματική (τύπου 0). Τα παρακάτω προβλήματα δεν είναι αποκρίσιμα:

- $w \in L(G)$;
- $L(G_1) = L(G_2)$; *(ισοδυναμία δύο γενικών γραμματικών)*
- $L(G) = \varnothing$;
- $L(G) = \Sigma^*$;
- $L(G)$ *κανονική* ;
- $L(G)$ *c.f.* ;
- $L(G)$ *c.s.* ;
- $L(G)$ *πεπερασμένη* ;

Αποκρίσιμα και μη προβλήματα II

Πρόταση

Για ένα LBA M είναι αποκρίσιμο το πρόβλημα:

- $w \in L(M)$;

και μη αποκρίσιμα τα προβλήματα:

- $L(M_1) = L(M_2)$; *(ισοδυναμία δύο LBA)*
- $L(M) = \varnothing$;
- $L(M) = \Sigma^*$;
- $L(M)$ *κανονική* ;
- $L(M)$ *c.f.* ;

Αποκρίσιμα και μη προβλήματα III

Πρόταση

Για μια context free γλώσσα L είναι αποκρίσιμα τα προβλήματα:

- $w \in L(G)$;
- $L(G) = \varnothing$;

και μη αποκρίσιμα τα προβλήματα:

- $L(G_1) = L(G_2)$; *(ισοδυναμία δύο c.f. γραμματικών)*
- $L(G) = \Sigma^*$;
- $L(G)$ *κανονική* ;

Αποκρίσιμα και μη προβλήματα IV

Επίσης, έχει νόημα να αναζητήσουμε αν η τομή και το συμπλήρωμα γλωσσών ιδίου τύπου παραμένει γλώσσα του ιδίου τύπου.

Για την τομή, η προκύπτουσα γλώσσα παραμένει του ιδίου τύπου αν τέμνουμε γλώσσες του ιδίου τύπου 0, 1, 3 (γενικές, c.s., regular). Για τις γλώσσες τύπου 2, δηλαδή τις c.f., η τομή δεν είναι πάντοτε c.f. γλώσσα, αλλά και επιπλέον το πρόβλημα δεν είναι καν αποκρίσιμο.

Για το συμπλήρωμα, η προκύπτουσα γλώσσα παραμένει του ιδίου τύπου αν θεωρήσουμε συμπλήρωμα γλώσσας τύπου 1 και 3 (c.s. και regular). Για τις γλώσσες τύπου 0 και 2 (γενικές και c.f.) το πρόβλημα δεν είναι καν αποκρίσιμο.

Η μη αποκρισιμότητα πολλών από τα παραπάνω προβλήματα αποδεικνύεται με αναγωγή από το επίσης μη αποκρίσιμο πρόβλημα PCP (Post's Correspondence Problem). Πριν ορίσουμε το πρόβλημα, δίνουμε τις έννοιες του συστήματος αντιστοίχησης και του ταιριάσματος.

Αποκρίσιμα και μη προβλήματα V

Ορισμός

Ένα *σύστημα αντιστοίχησης* είναι ένα πεπερασμένο σύνολο P αποτελούμενο από διατεταγμένα ζεύγη μη κενών συμβολοσειρών επί ενός αλφαβήτου :

$$\{(l_1, r_1), (l_2, r_2), \ldots, (l_n, r_n)\}.$$

Ορισμός

Ένα *ταίριασμα* (match) είναι μία ακολουθία ζευγών από το P: $(l_{i_1}, r_{i_1}), \ldots, (l_{i_k}, r_{i_k})$ τέτοια ώστε $l_{i_1} \cdots l_{i_k} = r_{i_1} \cdots r_{i_k}$.

Παράδειγμα

Αν $P = \{(abc, c), (ca, a), (a, ab), (b, ca), (aa, bb)\}$, ένα ταίριασμα είναι το $(a, ab)(b, ca)(ca, a)(a, ab)(abc, c)$.

Ορισμός (PCP: Post's Correspondence Problem)

Υπάρχει ταίριασμα για ένα δεδομένο σύστημα αντιστοίχησης;

Αποκρίσιμα και μη προβλήματα VI

Πρόταση

*Το **PCP** είναι μη αποκρίσιμο.*

Ένας άλλος φορμαλισμός που δίνει μη επιλύσιμα προβλήματα είναι τα συστήματα Thue:

Ορισμός

Ένα *σύστημα Thue* $T = (\Sigma, P)$ αποτελείται από ένα αλφάβητο Σ και ένα σύνολο P κανόνων παραγωγής της μορφής $l \to r$, όπου l, r συμβολοσειρές επί του Σ.
Όπως και στις γραμματικές: $u_1 l u_2 \Rightarrow u_1 r u_2$ αν $(l \to r) \in P$.

Παράδειγμα

Έστω το σύστημα Thue: $T = (\{a, b\}, \{ab \to aa, a \to bb\})$. Τότε, ισχύει:
$abba \Rightarrow abbbb \Rightarrow aabbb \Rightarrow abbbbb$.

Θεωρούμε το συμμετρικό μεταβατικό ανακλαστικό κλείσιμο της σχέσης "$\Rightarrow$" το οποίο και συμβολίζουμε με "$\overset{*}{\Leftrightarrow}$". Το παρακάτω πρόβλημα είναι ενδιαφέρον:

Αποκρίσιμα και μη προβλήματα VII

Ορισμός (Word problem)

Δίνονται x, $y \in \Sigma^*$ και σύστημα Thue. Ισχύει $x \overset{*}{\Leftrightarrow} y$ στο σύστημα Thue;

Για το σύστημα Thue του προηγούμενου παραδείγματος το word problem έχει θετική απάντηση για τα παρακάτω ζευγάρια (x, y) (γιατί;):

$$(abba, ababb), \quad (bba, bba), \quad (bbbb, bbb)$$

και αρνητική απάντηση για το (ba, aaa) (γιατί;).

Τα συστήματα Thue είναι σαν τις γραμματικές χωρίς όμως διαφοροποίηση τερματικών και μη τερματικών συμβόλων και χωρίς αρχικό σύμβολο. Συνήθως, απαιτούμε επιπλέον σε ένα σύστημα Thue η σχέση $\longrightarrow$ να είναι συμμετρική, δηλαδή όποτε υπάρχει ο κανόνας παραγωγής $l \longrightarrow r$ στο P να υπάρχει και ο $r \longrightarrow l$. Τότε το word problem διατυπώνεται απλούστερα ως: "Για δεδομένα x, y, ισχύει $x \overset{*}{\Rightarrow} y$;".

Τα συστήματα Thue ως συστήματα υπολογισμού έχουν ισοδύναμη υπολογιστική ισχύ με τις μηχανές Turing. Μάλιστα, έχουμε το παρακάτω αποτέλεσμα μη αποκρισιμότητας:

Πρόταση

Το word problem είναι μη αποκρίσιμο.

Περιεχόμενα

1 Πεπερασμένα Αυτόματα και Κανονικά Σύνολα

- Παραλλαγές, επεκτάσεις και εφαρμογές FA/REGEXP
- Ιδιότητες κανονικών συνόλων
- Αλγεβρική περιγραφή κανονικών συνόλων. Ελαχιστοποίηση DFA

2 Τυπικές Γλώσσες

- Τυπικές Γραμματικές
- Απλοποίηση c.f. γραμματικών
- Αυτόματα στοίβας (pushdown automata)
- Ιδιότητες c.f. γλωσσών

3 Μοντέλα Υπολογισμού

- Ιστορία - Εισαγωγή
- LOOP: Μια απλή γλώσσα προγραμματισμού
- LOOP-υπολογίσιμες και πρωταρχικές αναδρομικές συναρτήσεις

Κλάσεις Πολυπλοκότητας Ι

- **Θεωρία Υπολογισμού**: Μας ενδιαφέρει μόνον αν ένα πρόβλημα είναι **υπολογίσιμο ή όχι**.
- **Θεωρία Πολυπλοκότητας**: Θεωρούμε μόνο υπολογίσιμα προβλήματα και προσπαθούμε να δούμε αν μπορούν να επιλυθούν με **περιορισμούς στους διαθέσιμους υπολογιστικούς πόρους**, όπως ο χρόνος υπολογισμού, ο επιπλέον χώρος μνήμης που απαιτείται για ενδιάμεσα αποτελέσματα κατά την επίλυση, και άλλοι.
- Αυτοί οι *περιορισμοί*, καθώς και άλλα χαρακτηριστικά των υπολογισμών ορίζουν **κλάσεις πολυπλοκότητας** μέσα στις οποίες τοποθετούμε τα διάφορα προβλήματα.

Παρά τις συνεχείς και μακροχρόνιες προσπάθειες πολλών επιστημόνων, υπάρχουν αρκετά **ανοιχτά ερωτήματα**. Π.χ., υπάρχουν προβλήματα για τα οποία, αν και ανήκουν στο NP, δεν έχει βρεθεί πολυωνυμικός αλγόριθμος, αλλά ούτε απόδειξη ότι είναι NP-complete.

Κλάσεις Πολυπλοκότητας (Τι δεν ήξερε ο Karp το 1972)

- **GRAPH ISOMORPHISM** (το πιο γνωστό ανοιχτό πρόβλημα): Δεδομένων δύο γράφων είναι ισομορφικοί; (παράβαλε με το SUBGRAPH ISOMORPHISM το οποίο είναι γνωστό ότι είναι NP-complete)
- **LINEAR PROGRAMMING** (παρέμενε για χρόνια ανοιχτό): δεδομένου ενός συστήματος γραμμικών εξισώσεων και ανισοτήτων και μιας γραμμικής αντικειμενικής συνάρτησης (μεγιστοποίηση ή ελαχιστοποίηση) να ευρεθεί μια βέλτιστη εφικτή λύση;
  - **Μέθοδος Simplex** (Dantzig): Στη χειρότερη περίπτωση χρειαζόταν εκθετικό χρόνο.
  - **Ελλειψοειδής μέθοδος** (Khachiyan): Ο πρώτος πολυωνυμικός αλγόριθμος για τον γραμμικό προγραμματισμό. Δεν είχε μεγάλο πρακτικό ενδιαφέρον.
  - **Αλγόριθμος Karmarkar**: Πολυωνυμικός αλγόριθμος που είχε και πρακτικά αποτελέσματα καλύτερα από τη μέθοδο Simplex.
- **PRIMALITY**: Δίνεται ένας ακέραιος. Είναι πρώτος ή όχι;
  Πρόσφατα (2002 από τους Agrawal, Kayal, Saxena — AKS) απεδείχθη και ότι το πρόβλημα αυτό, που παρέμενε για αρκετό καιρό ανοικτό, είναι στο P.

Βασικοί Ορισμοί Ι

Ορισμός

Στην κλάση TIME($t(n)$) (ή DTIME($t(n)$)) ανήκουν τα προβλήματα που μπορούν να επιλυθούν από **ντετερμινιστική** μηχανή Turing σε χρόνο $t(n)$.

Ορισμός

Στην κλάση NTIME($t(n)$) ανήκουν τα προβλήματα που μπορούν να επιλυθούν από **μη ντετερμινιστική** μηχανή Turing σε χρόνο $t(n)$.

Ορισμός

Στην κλάση SPACE($s(n)$) (ή DSPACE($s(n)$)) ανήκουν τα προβλήματα που μπορούν να επιλυθούν από **ντετερμινιστική** μηχανή Turing χρησιμοποιώντας επιπλέον χώρο $s(n)$.

Ορισμός

Στην κλάση NSPACE($s(n)$) ανήκουν τα προβλήματα που μπορούν να επιλυθούν από **μη ντετερμινιστική** μηχανή Turing χρησιμοποιώντας επιπλέον χώρο $s(n)$.

Βασικοί Ορισμοί II

Με βάση τα παραπάνω, ορίζουμε:

- $\mathrm{P} = \mathrm{PTIME} = \bigcup_{i \geqslant 1} \mathrm{DTIME}(n^i)$
- $\mathrm{NP} = \mathrm{NPTIME} = \bigcup_{i \geqslant 1} \mathrm{NTIME}(n^i)$
- $\mathrm{PSPACE} = \bigcup_{i \geqslant 1} \mathrm{DSPACE}(n^i)$
- $\mathrm{NPSPACE} = \bigcup_{i \geqslant 1} \mathrm{NSPACE}(n^i)$
- $\mathrm{L} = \mathrm{DSPACE}(\log n)$
- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$
- $\mathrm{EXP} = \bigcup_{i \geqslant 1} \mathrm{DTIME}(2^{n^i})$
- $\mathrm{EXPSPACE} = \bigcup_{i \geqslant 1} \mathrm{DSPACE}(2^{n^i})$

Παρατήρηση

Μία συνάρτηση f ονομάζεται συνάρτηση πολυπλοκότητας (constructible) αν πρέπει να υπάρχει μία TM τέτοια ώστε: $\forall$ input x με $|x| = n$, αποδέχεται το input σε χρόνο $O(n + f(n))$ (time-constructible) ή working space $O(f(n))$ (space-constructible).

Βασικοί Ορισμοί III

Αν η f είναι μία συνάρτηση πολυπλοκότητας τότε ισχύουν:

- $\text{DSPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$
- $\text{DTIME}(f(n)) \subseteq \text{NTIME}(f(n))$

διότι κάθε ντετερμινιστική μηχανή Turing μπορεί να θεωρηθεί ως μη ντετερμινιστική με μία μόνο επιλογή σε κάθε βήμα.

- $\text{DTIME}(f(n)) \subseteq \text{DSPACE}(f(n))$
- $\text{NTIME}(f(n)) \subseteq \text{DSPACE}(f(n))$

διότι σε χρόνο $f(n)$ δεν μπορεί να εξεταστεί χώρος (αριθμός θέσεων στην ταινία της $T.M.$) παραπάνω από $f(n)$.

Αν $f(n) > \log n$ τότε:

- $\text{DSPACE}(f(n)) \subseteq \text{DTIME}(c^{f(n)})$

Βασικοί Ορισμοί IV

- $\mathrm{NTIME}(f(n)) \subseteq \mathrm{DTIME}(c^{f(n)})$
- $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{DTIME}(k^{f(n)})$

Το παρακάτω θεώρημα οφείλεται στον Savitch (1970):

Θεώρημα

Αν $f(n) \geqslant \log n$ *τότε* $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{DSPACE}(f^2(n))$.

Άμεσα από το θεώρημα του Savitch προκύπτει ότι $\mathrm{PSPACE} = \mathrm{NPSPACE}$.

Από τις παραπάνω σχέσεις προκύπτει η εξής ιεραρχία:

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE}$$

Γνωρίζουμε ότι $\mathrm{L} \neq \mathrm{PSPACE}$ και $\mathrm{NL} \neq \mathrm{PSPACE}$ (αυτό προκύπτει από το θεώρημα ιεραρχίας για χωρικές κλάσεις πολυπλοκότητας, που αναφέρεται παρακάτω).

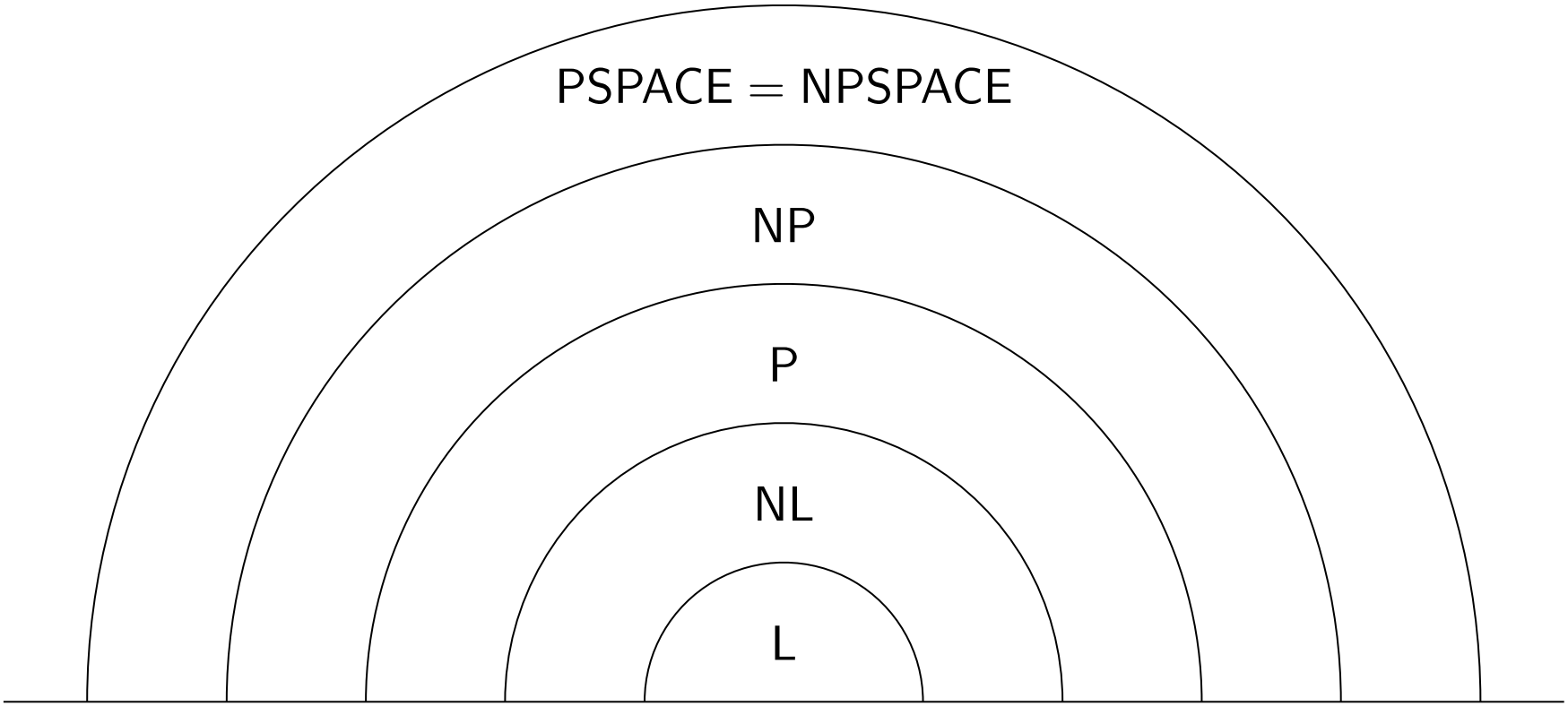
Βασικοί Ορισμοί V

Ανοιχτά παραμένουν τα προβλήματα:

$$\mathrm{L} \supseteq \mathrm{NL} \supseteq \mathrm{P} \supseteq \mathrm{NP} \supseteq \mathrm{PSPACE}$$

Βασικοί Ορισμοί VI

Ο κόσμος μοιάζει, ως τώρα, να είναι όπως στο παρακάτω σχήμα.



Σχήμα: Κλάσεις πολυπλοκότητας

Βασικοί Ορισμοί VII

Επίσης, πρέπει να αναφέρουμε ότι οι παραπάνω κλάσεις πολυπλοκότητας αφορούν προβλήματα **απόφασης**. Μπορούμε επίσης να ορίσουμε κλάσεις πολυπλοκότητας για μηχανές Turing που υπολογίζουν **συναρτήσεις**. Ένα χαρακτηριστικό παράδειγμα είναι η παρακάτω κλάση:

Ορισμός

FP = το σύνολο των συναρτήσεων που υπολογίζεται από ντετερμινιστική μηχανή Turing σε πολυωνυμικό χρόνο.

Η κλάση FP θα φανεί χρήσιμη παρακάτω στον ορισμό των αναγωγών, αφού περιλαμβάνει τις "εύκολα" υπολογιζόμενες συναρτήσεις.

Μία άλλη επίσης χρήσιμη κλάση πολυπλοκότητας που αφορά συναρτήσεις είναι η εξής:

Ορισμός

FL = το σύνολο των συναρτήσεων που υπολογίζεται από ντετερμινιστική μηχανή Turing σε λογαριθμικό χώρο.

Θεωρήματα ιεραρχίας Ι

Ισχύουν τα παρακάτω θεωρήματα για το μοντέλο της ντετερμινιστικής μηχανής Turing με τρεις ταινίες (θεωρούμε πάντοτε συναρτήσεις πολυπλοκότητας t_1, t_2, s_1, s_2):

Θεώρημα (Fürer, 1982)

Έστω $t_2(n) > n$. Τότε υπάρχει γλώσσα που γίνεται αποδεκτή σε χρόνο t_2, αλλά όχι σε χρόνο t_1 για οποιοδήποτε $t_1 = o(t_2(n))$.

Θεώρημα (Hartmanis, Lewis, Stearns, 1965)

Έστω $s_2(n) > \log n$. Τότε υπάρχει γλώσσα που γίνεται αποδεκτή σε χώρο s_2, αλλά όχι σε χώρο s_1 για οποιοδήποτε $s_1 = o(s_2(n))$.

Οι τεχνικές αποδείξεις παραλείπονται.

Παρόμοια θεωρήματα ισχύουν και για μη ντετερμινιστικές μηχανές Turing. Οι αποδείξεις μάλιστα είναι πιο εύκολες.

Θεωρήματα ιεραρχίας II

Η εμμονή μας σε constructible συναρτήσεις πολυπλοκότητας οφείλεται στο γεγονός ότι αν επιτρέψουμε οποιαδήποτε συνάρτηση στην θέση των $t(n)$, $s(n)$, τότε προκύπτουν διάφορα παθολογικά φαινόμενα, όπως το παρακάτω:

Θεώρημα (Gap theorem)

Υπάρχει αναδρομική συνάρτηση $t(n)$, τέτοια ώστε $\mathrm{TIME}(t(n)) = \mathrm{TIME}(2^{t(n)})$.

Συμπληρωματικές κλάσεις πολυπλοκότητας Ι

Ορισμός

Έστω γλώσσα L. Ως γνωστόν, το συμπλήρωμα της γλώσσας συμβολίζεται και ορίζεται ως εξής: $\overline{L} = \{x \mid x \notin L\}$. Τώρα, για μία κλάση γλωσσών $\mathcal{C}$, ορίζουμε (με την βοήθεια του συμπληρώματος):

$$\text{co}\mathcal{C} = \{\overline{L} \mid L \in \mathcal{C}\}.$$

Παράδειγμα: η κλάση coNP αποτελείται από τις γλώσσες που είναι συμπληρώματα γλωσσών στο NP. Ένα πρόβλημα που ανήκει στην κλάση coNP είναι το $\overline{\text{SAT}}$ ή το στενά συσχετιζόμενο με αυτό πρόβλημα της ταυτολογίας, αν δηλαδή ένας λογικός τύπος που δίνεται είναι ταυτολογία.

Συμπληρωματικές κλάσεις πολυπλοκότητας II

Έχει ενδιαφέρον να δούμε ποιες κλάσεις πολυπλοκότητας είναι **κλειστές ως προς συμπλήρωμα**, δηλαδή για ποιες κλάσεις $\mathcal{C}$ ισχύει $\mathcal{C} = co\mathcal{C}$.

Γενικά, οι ντετερμινιστικές κλάσεις πολυπλοκότητας (είτε χρονικές, είτε χωρικές) είναι κλειστές ως προς συμπλήρωμα, δηλαδή, οι $DTIME(t(n))$ και $DSPACE(s(n))$ είναι κλειστές ως προς συμπλήρωμα.

Αν θεωρήσουμε μη ντετερμινισμό, το πρόβλημα είναι ανοιχτό στην περίπτωση της χρονικής πολυπλοκότητας. Για παράδειγμα δεν γνωρίζουμε αν $\text{coNP} \neq \text{NP}$. Μάλιστα, το τελευταίο συνδέεται και με το πρόβλημα αν $\text{P} \neq \text{NP}$, αφού προφανώς αν $\text{coNP} \neq \text{NP}$, τότε $\text{P} \neq \text{NP}$.

Συμπληρωματικές κλάσεις πολυπλοκότητας III

Θεώρημα (Immerman-Szelepcsényi)

Η κλάση $\mathrm{NSPACE}(s(n))$ *είναι κλειστή ως προς συμπλήρωμα.*

Για $s(n) = n$ έχουμε την κλάση προβλημάτων που επιλύονται από μηχανή Turing που χρησιμοποιεί γραμμικό χώρο, αλλιώς γνωστό και ως LBA (linearly bounded automaton), οπότε το παραπάνω θεώρημα έλυσε και ένα, για πολλά χρόνια, ανοικτό πρόβλημα, αν δηλαδή η κλάση των LBA (ή ισοδύναμα η κλάση των context sensitive γλωσσών, από ένα αποτέλεσμα του Kuroda, του 1964) είναι κλειστή ως προς συμπλήρωμα.

Αναγωγές Ι

Η έννοια της **αναγωγής** σε πολυωνυμικό χρόνο πρέπει να συνδέει μεταξύ τους προβλήματα με υπολογιστικά "εύκολο" τρόπο. Θεωρούμε εύκολες συναρτήσεις (και προβλήματα) που υπολογίζονται σε πολυωνυμικό χρόνο.

Θα θέλαμε:

- Αν οι συναρτήσεις f και g είναι "εύκολες", τότε και η σύνθεσή τους $f \circ g$ είναι "εύκολη".
- Αν η f είναι υπολογίσιμη σε χρόνο $O(n^2)$, τότε θεωρείται εύκολη.

Άρα θεωρούμε εύκολα προβλήματα (και συναρτήσεις) αυτά που υπολογίζονται σε πολυωνυμικό χρόνο (έστω και σε $O(n^{1000})$). Για τους παραπάνω λόγους, ορίζουμε την αναγωγή κατά Karp:

Ορισμός (Αναγωγή κατά Karp)

$$A \leqslant_m^P B: \quad \exists f \in \mathrm{FP}, \forall x (x \in A \iff f(x) \in B)$$

Αναγωγές II

Υπάρχουν και άλλες χρήσιμες αναγωγές, όπως η λεγόμενη log-space, που χρησιμοποιεί λογαριθμικό χώρο, και η οποία είναι χρήσιμη για αναγωγές προβλημάτων σε μικρότερες κλάσεις πολυπλοκότητας, όπως η P:

Ορισμός (Log-space Αναγωγή)

$$A \leqslant_m^L B: \quad \exists f \in \mathrm{FL}, \forall x (x \in A \iff f(x) \in B)$$

Ισχύει: $A \leqslant_m^L B \implies A \leqslant_m^P B$, αλλά όχι το αντίστροφο.

Μία επιθυμητή ιδιότητα μίας αναγωγής είναι να είναι κλειστή ως προς διάφορες κλάσεις γλωσσών:

Ορισμός

Λέμε ότι μία κλάση γλωσσών C είναι **κλειστή** ως προς μία αναγωγή $\leqslant$ αν

$$A \leqslant B \wedge B \in C \implies A \in C.$$

Αναγωγές III

Μερικές από τις κλάσεις πολυπλοκότητας που είναι κλειστές ως προς την αναγωγή κατά Karp ($\leqslant_m^P$) είναι οι εξής: P, PSPACE, EXP, EXPSPACE (βλέπε παραπάνω για τους ορισμούς τους).

Ορισμός (Hardness)

Λέμε ότι A είναι C-hard (C-δύσκολο), ως προς την $\leqslant$, αν:

$$\forall B \in C : B \leqslant A.$$

Η έννοια της hardness δίνει ένα κάτω όριο για την πολυπλοκότητα ενός προβλήματος, δεδομένου ότι το πρόβλημα A είναι τουλάχιστον τόσο δύσκολο όσο οποιοδήποτε πρόβλημα μίας κλάσης C.

Ορισμός (Completeness)

Λέμε ότι A είναι C-complete (C-πλήρες), ως προς την $\leqslant$, αν:

$$A \text{ είναι } C - \text{hard ως προς } \leqslant \quad \wedge \quad A \in C.$$

Αναγωγές IV

Παρακάτω δίνουμε πλήρη προβλήματα για μερικές από τις σημαντικότερες κλάσεις πολυπλοκότητας:

- NL: το πρόβλημα Reachability (log-space αναγωγές).
- P: Circuit-Value και Linear Programming (πάλι με log-space αναγωγές).
- NP: το 3SAT.
- PSPACE: το QBF (Quantified Boolean Formula satisfiability problem).
- EXP: το $n \times n$ Go.
- EXPSPACE: το RegExp$(\cup, \cdot, *, ^2)$, που είναι το πρόβλημα ελέγχου ισοδυναμίας regular expressions, που χρησιμοποιούν τους τελεστές $\cup$ (ένωση), $\cdot$ (παράθεση), $*$ (άστρο του Kleene) και ^{2}, όπου $\alpha^2 = \alpha \cdot \alpha$.

Παράμετροι για ορισμό κλάσεων πολυπλοκότητας I

- **Concrete Complexity**: Θεωρούμε κάποιο *συγκεκριμένο* μοντέλο υπολογισμού, κάποιο *συγκεκριμένο* πρόβλημα και κάποιον *συγκεκριμένο* αλγόριθμο για το πρόβλημα σε αυτό το μοντέλο. Έτσι καθορίζουμε την ακριβή πολυπλοκότητα του αλγορίθμου (οι σταθερές φυσικά δεν παίζουν ρόλο).
- **Abstract** ή αλλιώς **structural complexity**: Θεωρούμε κλάσεις πολυπλοκότητας με διάφορες *υπολογιστικές παραμέτρους* και συγκρίνουμε τις κλάσεις μεταξύ τους (ως προς εγκλεισμό, διαχωρισμό κ.τ.λ.). Χρήσιμο για τις συγκρίσεις είναι να βρούμε αναγωγές και προβλήματα που είναι πλήρη σε αυτές τις κλάσεις ως προς αυτές τις αναγωγές.

Παράμετροι για ορισμό κλάσεων πολυπλοκότητας II

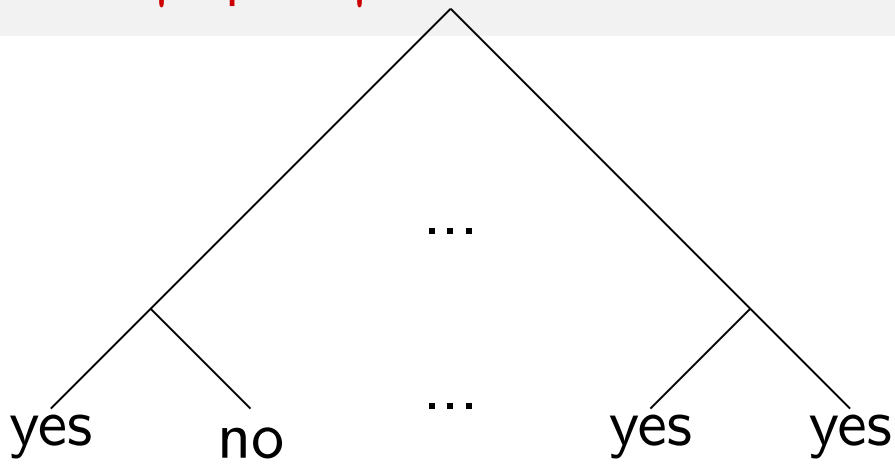
Αναφέρουμε επιγραμματικά μερικές παραμέτρους με τις οποίες ορίζονται κλάσεις πολυπλοκότητας:

- **μοντέλο υπολογισμού**: Μηχανή Turing (TM), Random Access Machine (RAM), πεπερασμένο αυτόματο, Linearly Bounded Automaton (LBA), Παράλληλη RAM (PRAM), μονότονα κυκλώματα (monotone circuits)·
- **μέθοδος λειτουργίας/αποδοχής**: ντετερμινιστική, μη ντετερμινιστική, πιθανοτική, εναλλασσόμενη (alternating), παράλληλη·
- **είδος μοντέλου/λειτουργίας**: αποφασιστής (decider), αποδέκτης (acceptor), γεννήτρια (generator), μετατροπέας (transducer)·
- **αγαθά**: αριθμός βημάτων, αριθμός συγκρίσεων, αριθμός πολλαπλασιασμών, χρόνος, χώρος μνήμης, πλήθος επεξεργαστών, αριθμός εναλλαγών στο υπολογιστικό δένδρο, μέγεθος (size) κυκλώματος, βάθος (depth) κυκλώματος·
- **άλλα εργαλεία**: τυχαιότητα (randomness), μαντεία (oracles), διαλογική αλληλεπίδραση (interactivity), υπόσχεση (promise), τελεστές (operators)·
- **φράγματα (bounds) ως προς το μήκος της εισόδου**: για παράδειγμα $O(n^3)$ ή πολυωνυμικό, time/space $(t(n), s(n))$ tradeoff (αντιστάθμισμα), Probabilistic Checkable Proofs: PCP$(r(n), q(n))$ (με χρήση $r(n)$ τυχαίων bits και $q(n)$ queries, ερωτήσεων, στην απόδειξη).

Μοντέλα δένδρων υπολογισμού για TM I

- Για να μελετήσουμε την συμπεριφορά μηχανών Turing, θα κωδικοποιήσουμε τους υπολογισμούς μίας μηχανής Turing με ένα **δέντρο υπολογισμού**.
- Ο υπολογισμός ξεκινά στην **ρίζα** του δένδρου.
- Θεωρούμε ότι αν σε κάποιο σημείο του υπολογισμού έχουμε μία μη ντετερμινιστική επιλογή τότε έχουμε μία **διακλάδωση** στο δένδρο.
- Στα **φύλλα** της μηχανής TM έχουμε τις απαντήσεις της μηχανής Turing. Κάθε μονοπάτι από την ρίζα του δένδρου μέχρι κάποιο φύλλο επομένως κωδικοποιεί έναν πιθανό υπολογισμό
- Χωρίς βλάβη της γενικότητας, υποθέτουμε ότι το δένδρο είναι δυαδικό, πλήρες και γεμάτο. όλα τα φύλλα του είναι στο ίδιο επίπεδο.

Μοντέλα δένδρων υπολογισμού για TM II



Σχήμα: Μοντέλο δένδρων υπολογισμού

Επίσης, έχει ενδιαφέρον το μήκος του υπολογιστικού μονοπατιού από την ρίζα μέχρι το φύλλο να έχει **πολυωνυμικό μήκος** ως προς το μήκος της εισόδου (να αντιστοιχεί δηλαδή το κάθε μονοπάτι σε κάποιον «εύκολο», δηλαδή πολυωνυμικό, υπολογισμό).

Θεωρώντας το παραπάνω μοντέλο, θα ορίσουμε μερικές από τις γνωστές κλάσεις υπολογισμού, καθώς και μερικές καινούριες. Πιο συγκεκριμένα, Θα χρησιμοποιήσουμε ποσοδείκτες $(\exists, \forall)$ στα μονοπάτια. Επειδή εννοείται πάντοτε ο περιορισμός του μήκους των μονοπατιών, θα γράφουμε π.χ. $\exists y$, αντί για $\exists y\colon |y| \leqslant p(|x|)$, όπου y: μεταβλητή για τα μονοπάτια, x: μεταβλητή για την είσοδο, p: πολυώνυμο.

Μοντέλα δένδρων υπολογισμού για TM III

Για παράδειγμα, η κλάση P μπορεί να περιγραφεί ως εξής:

$$L \in \mathrm{P} \iff \exists R \in \mathrm{P}: \begin{cases} x \in L & \implies \forall y R(x,y) \\ x \notin L & \implies \forall y \neg R(x,y) \end{cases}$$

Περισσότερο ενδιαφέρον έχει η κλάση NP που μπορεί να περιγραφεί ως εξής:

$$L \in \mathrm{NP} \iff \exists R \in \mathrm{P}: \begin{cases} x \in L & \implies \exists y R(x,y) \\ x \notin L & \implies \forall y \neg R(x,y) \end{cases}$$

Δηλαδή, αν $x \in L$ υπάρχει τουλάχιστον ένας υπολογισμός που αποδέχεται, ενώ αν $x \notin L$ κανένας υπολογισμός δεν αποδέχεται.

Μοντέλα δένδρων υπολογισμού για TM IV

Παρομοίως, η κλάση coNP περιγράφεται ως εξής:

$$L \in \text{coNP} \iff \exists R \in \text{P}: \begin{cases} x \in L & \implies \forall y R(x,y) \\ x \notin L & \implies \exists y \neg R(x,y) \end{cases}$$

Παρατηρούμε ότι οι ποσοδείκτες που χρησιμοποιούνται και αντιστοιχούν στο «$x \in L$» και στο «$x \notin L$» καθορίζουν πλήρως την αντίστοιχη κλάση πολυπλοκότητας. Έτσι, εισάγουμε τον παρακάτω συμβολισμό:

$$\text{P} = (\forall, \forall), \quad \text{NP} = (\exists, \forall), \quad \text{coNP} = (\forall, \exists).$$

Τυχαιότητα (Randomness) I

- Χρησιμοποιώντας το μοντέλο **δένδρων υπολογισμού**, θα ορίσουμε κλάσεις πολυπλοκότητας που βασίζονται στις **πιθανότητες**, με βάση τυχαίες επιλογές.
- Αυτή η προσέγγιση είναι πολύ χρήσιμη από πρακτική άποψη, αφού σε πολλές εφαρμογές, είναι ικανοποιητικός ένας αλγόριθμος ο οποίος κάνοντας κάποιες τυχαίες επιλογές, δίνει στις περισσότερες των περιπτώσεων το σωστό αποτέλεσμα.
- Ένας **πιθανοκρατικός αλγόριθμος** είναι συνήθως πιο απλός στην διατύπωσή του και στην πράξη πιο αποδοτικός από έναν αντίστοιχο ντετερμινιστικό που επιλύει το ίδιο πρόβλημα. Για παράδειγμα, απλοί πιθανοκρατικοί αλγόριθμοι για τον έλεγχο αν ένας αριθμός είναι πρώτος υπάρχουν από την δεκαετία του 1970 και χρησιμοποιούνται στην πράξη έναντι πιο περίπλοκων ντετερμινιστικών τύπου AKS.
- Στα πλαίσια του μοντέλου δένδρων υπολογισμού, θα θεωρήσουμε ότι η επιλογή σε κάθε κόμβο του δένδρου γίνεται τυχαία με πιθανότητα $1/2$ για κάθε παιδί του κόμβου. Για να δείξουμε ότι η *«συντριπτική» πλειοψηφία* των υπολογισμών δίνει το σωστό αποτέλεσμα, εισάγουμε έναν νέο ποσοδείκτη, τον $\exists^+$.

Τυχαιότητα (Randomness) II

- Με την βοήθεια του $\exists^+$, ορίζουμε την κλάση BPP, από το **Bounded error Probabilistic Polynomial**:

Ορισμός ($\mathrm{BPP} = (\exists^+, \exists^+)$)

$$L \in \mathrm{BPP} \iff \exists R \in \mathrm{P}: \begin{cases} x \in L & \implies \exists^+ y R(x,y) \\ x \notin L & \implies \exists^+ y \neg R(x,y) \end{cases}$$

Τυχαιότητα (Randomness) III

- Με άλλα λόγια, σε ένα δέντρο για την κλάση BPP έχουμε την «συντριπτική» πλειοψηφία των φύλλων να δίνει το σωστό αποτέλεσμα. Στον παραπάνω ορισμό, δεν έχει μεγάλη σημασία ο ακριβής ορισμός της «συντριπτικής» πλειοψηφίας, αλλά πρέπει να είναι οπωσδήποτε *φραγμένος* (εξ ου και το 'bounded' του BPP) πάνω από το $1/2$. Το ποσοστό της πλειοψηφίας μπορεί να είναι, ενδεικτικά, μεγαλύτερο από $1/2 + \varepsilon$, $1/2 + 1/p(|x|)$, $2/3$, 99%, $1 - 2^{-p(|x|)}$ (όπου $p(|x|) > 1$) ($2^{-p|(x|)}$ καλείται αμελητέα ποσότητα). Αυτή η δυνατότητα επιλογής υπάρχει, επειδή με πολυωνυμικές επαναλήψεις του αντίστοιχου αλγορίθμου, είναι δυνατόν να αυξήσουμε την πιθανότητα επιτυχίας, όσο θέλουμε. Αλγόριθμοι BPP ονομάζονται **Monte Carlo** ή αλλιώς two-sided error, επειδή ανεξάρτητα από το αποτέλεσμα (ναι ή όχι), υπάρχει κάποια πιθανότητα λάθους. Είναι προφανές ότι η κλάση BPP είναι κλειστή ως προς συμπλήρωμα.

Τυχαιότητα (Randomness) IV

- Ας θεωρήσουμε τώρα αλγορίθμους οι οποίοι κάνουν λάθος μόνον για την μία απάντηση (one sided error). Έτσι, προκύπτει η κλάση RP (**Randomized Polynomial**):

Ορισμός ($\mathrm{RP} = (\exists^+, \forall)$)

$$L \in \mathrm{RP} \iff \exists R \in \mathrm{P}: \begin{cases} x \in L & \implies \exists^+ y R(x,y) \\ x \notin L & \implies \forall y \neg R(x,y) \end{cases}$$

Σε αυτήν την κλάση, αν ο αντίστοιχος RP αλγόριθμος δώσει απάντηση «ναι» (δηλαδή το κατηγόρημα R υπολογιστεί αληθές), είμαστε σίγουροι ότι $x \in L$. Αντίθετα, η απάντηση «όχι» του RP αλγορίθμου δεν είναι «σίγουρη».

Προφανώς, ισχύουν: $\mathrm{RP} \subseteq \mathrm{BPP}$, $\mathrm{coRP} \subseteq \mathrm{BPP}$, αλλά δεν γνωρίζουμε αν $\mathrm{RP} = \mathrm{coRP}$.

Τυχαιότητα (Randomness) V

- Μία άλλη πολύ χρήσιμη κλάση, είναι αυτή που ορίζεται με τομή των RP και coRP, η $\mathrm{ZPP} = \mathrm{RP} \cap \mathrm{coRP}$. Η ονομασία προέρχεται από το **Zero error Probabilistic Polynomial**, γιατί μπορεί εύκολα να δειχτεί ότι ένα πρόβλημα είναι στο ZPP αν υπάρχει πιθανοκρατικός αλγόριθμος ο οποίος τρέχει σε αναμενόμενο πολυωνυμικό χρόνο και δίνει πάντοτε σωστή απάντηση. Πράγματι, αν ένα πρόβλημα είναι στο ZPP, σημαίνει ότι έχουμε ένα RP και έναν coRP αλγόριθμο για αυτό, οπότε αρκεί να τρέχουμε εναλλακτικά τους δύο αλγορίθμους, μέχρι ο ένας να δώσει την «σίγουρή» του απάντηση. Βέβαια, μπορεί να χρειαστεί να τρέχουμε εναλλακτικά τους δύο αλγορίθμους για πάντα, αλλά με μεγάλη πιθανότητα θα έχουμε μία «σίγουρη» απάντηση, μετά από μερικές επαναλήψεις. Εναλλακτικά, μπορούμε να πούμε ότι ένας ZPP αλγόριθμος έχει τρεις εξόδους: «ναι», «όχι» (για τις «σίγουρες» απαντήσεις), και «δεν ξέρω» (για τις όχι «σίγουρες»).

  Οι αλγόριθμοι στο ZPP ονομάζονται **Las Vegas**.

Τυχαιότητα (Randomness) VI

- Δεδομένου ότι υπάρχουν αρκετοί πιθανοκρατικοί αλγόριθμοι ευρείας χρήσης για πρακτικά προβλήματα, πολλοί τοποθετούν τους εφικτούς (feasible) υπολογισμούς πάνω από το P, στις πιθανοτικές κλάσεις BPP, RP, ZPP.

  Πάντως, δεν γνωρίζουμε αν υπάρχουν πλήρη προβλήματα για τις κλάσεις που ορίστηκαν παραπάνω (BPP, RP, ZPP).
- Αν τώρα το ποσοστό λάθους ενός πιθανοκρατικού αλγορίθμου δεν φραχθεί μακριά από το $1/2$, τότε έχουμε απλώς την βεβαιότητα ότι στο μοντέλο δένδρων υπολογισμού παραπάνω από τα μισά υπολογιστικά μονοπάτια δίνουν την σωστή απάντηση. Για να δηλώσουμε το παραπάνω χρησιμοποιούμε τον ποσοδείκτη $\exists_{1/2}$. Για unbounded two-sided error, έχουμε την κλάση PP (Probabilistic Polynomial):

Ορισμός ($\mathrm{PP} = (\exists_{1/2}, \exists_{1/2})$)

$$L \in \mathrm{PP} \iff \exists R \in \mathrm{P}: \begin{cases} x \in L & \implies \exists_{1/2} y R(x,y) \\ x \notin L & \implies \exists_{1/2} y \neg R(x,y) \end{cases}$$

Τυχαιότητα (Randomness) VII

- Λόγω της έλλειψης φράγματος για την πιθανότητα λάθους, δεν μπορούμε να χρησιμοποιήσουμε την τεχνική της επανάληψης για να βελτιώσουμε την πιθανότητα σωστού αποτελέσματος από έναν PP αλγόριθμο. Μία άλλη ένδειξη για το ανέφικτο της κλάσης PP σε σχέση με τις BPP, RP, ZPP, προκύπτει από το παρακάτω αποτέλεσμα:

Πρόταση

$\mathrm{NP} \subseteq \mathrm{PP}$.

- Πρέπει επίσης να σημειώσουμε ότι δεν λάβαμε καθόλου υπ' όψιν μας, ως υπολογιστικό πόρο, των αριθμό των τυχαίων bits που χρησιμοποιεί ένας πιθανοκρατικός αλγόριθμος. Στην πράξη, κάθε «τυχαίο» bit που χρειαζόμαστε δεν είναι χωρίς τίμημα, αφού το λαμβάνουμε από κάποια γεννήτρια *ψευδοτυχαίων* bits.
- Τέλος, αναφέρουμε και την κλάση RL (**Randomized Logspace**) που περιέχει τα προβλήματα που έχουν one-sided error αλγόριθμο που χρησιμοποιεί λογαριθμικό χώρο και πολυωνυμικό ως προς το μήκος της εισόδου αριθμό τυχαίων bits.

Πολυωνυμική Ιεραρχία Ι

Αντίστοιχα με προηγούμενο κεφάλαιο, όπου ορίσαμε μαντεία και την αριθμητική ιεραρχία, θα ορίσουμε την **πολυωνυμική ιεραρχία**, η οποία έχει παρόμοια δομή, αλλά βρίσκεται πολύ χαμηλότερα από άποψη υπολογιστικής πολυπλοκότητας. Υπενθυμίζουμε την έννοια του υπολογισμού με μαντείο: Ένας αλγόριθμος χρησιμοποιεί ένα μαντείο για το πρόβλημα Π, αν έχει την δυνατότητα κατά την διάρκεια του υπολογισμού, να ρωτάει το μαντείο για κάποιο στιγμιότυπο x του προβλήματος Π, αν $x \in \Pi$, και το μαντείο να του απαντά άμεσα με ένα «ναι» ή με ένα «όχι». Όσο δύσκολο και να είναι το πρόβλημα Π, ο αλγόριθμος δεν σπαταλά επιπλέον υπολογιστικούς πόρους.

Πολυωνυμική Ιεραρχία II

Ορισμός (Κλάσεις με μαντεία)

- $\mathcal{C}^{\Pi}$: η κλάση των προβλημάτων τα οποία λύνονται με αλγόριθμο στην κλάση $\mathcal{C}$ ο οποίος χρησιμοποιεί μαντείο για το πρόβλημα Π
- $\mathcal{C}^{\mathcal{C}_o} = \bigcup_{\Pi \in \mathcal{C}_o} \mathcal{C}^{\Pi}$

Για παράδειγμα, η κλάση $\mathrm{P}^{\mathrm{SAT}}$ αποτελείται από τα προβλήματα που λύνονται με ντετερμινιστικό πολυωνυμικό αλγόριθμο ο οποίος χρησιμοποιεί μαντείο για το πρόβλημα SAT. Άλλη περιγραφή της ίδιας κλάσης είναι: P^{NP} (αφού το SAT είναι NP-πλήρες).

Πολυωνυμική Ιεραρχία III

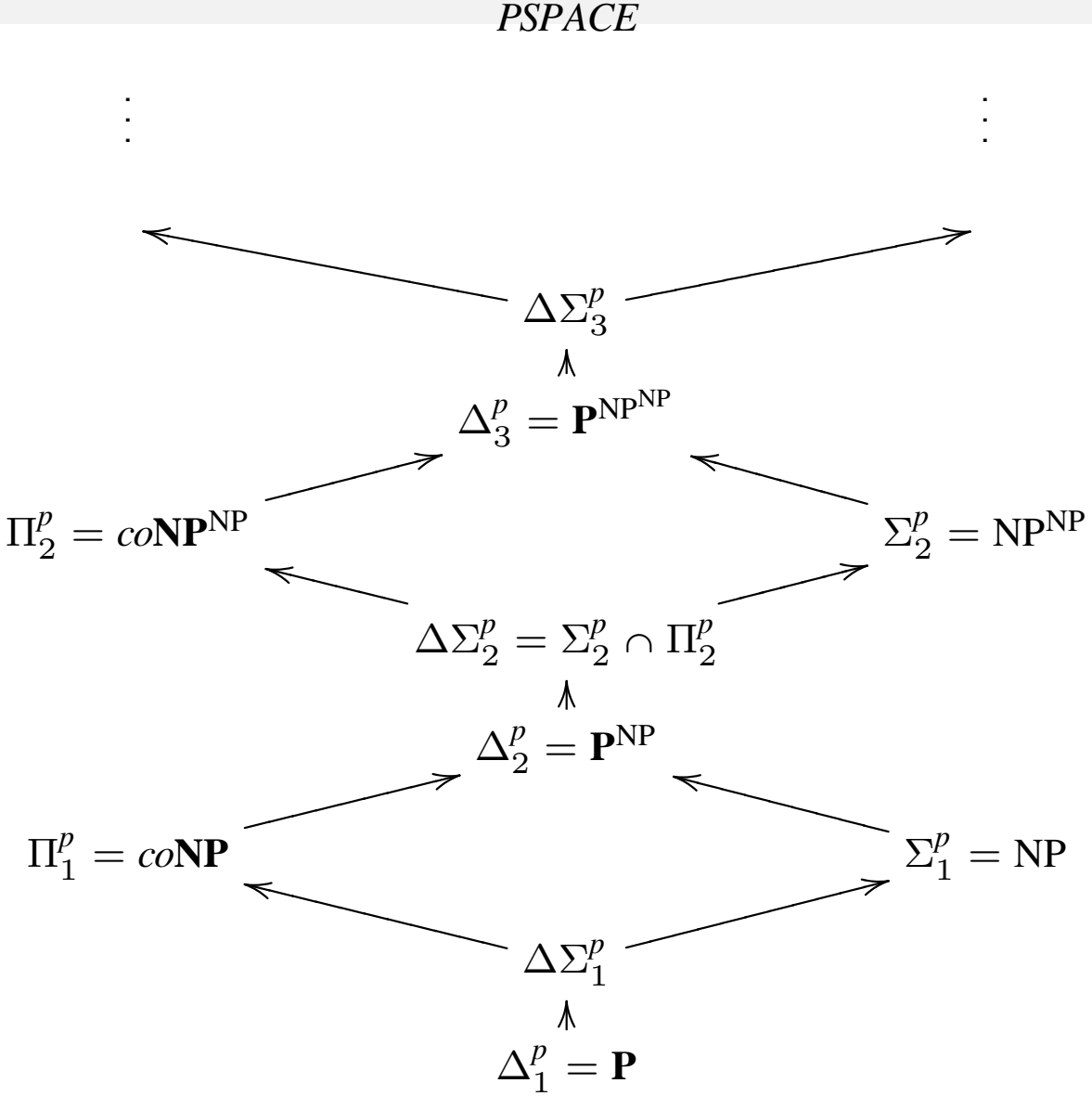
Ορισμός

$(k \geqslant 0)$

- $\Sigma_0^p = \Pi_0^p = \Delta_0^p = \mathrm{P}$
- $\Sigma_{k+1}^p = \mathrm{NP}^{\Sigma_k^p}$, $\Pi_{k+1}^p = \mathrm{co}\Sigma_{k+1}^p$, $\Delta_{k+1}^p = \mathrm{P}^{\Sigma_k^p}$, $\Delta\Sigma_k^p = \Sigma_k^p \cap \Pi_k^p$
- Πολυωνυμική ιεραρχία: $\mathrm{PH} = \bigcup_{k\in\mathbb{N}} \Sigma_k^p$

Ισχύουν τα παρακάτω: $\Sigma_1^p = \mathrm{NP}$, $\Pi_1^p = \mathrm{coNP}$ και για κάθε $k \geqslant 0$: $\Sigma_k^p \subseteq \Sigma_{k+1}^p$ και $\Pi_k^p \subseteq \Sigma_{k+1}^p$. Αν και δεν έχει αποδειχθεί το αυστηρό των παραπάνω εγκλεισμών (όπως στην αριθμητική ιεραρχία), εν τούτοις πιστεύουμε ότι η ιεραρχία είναι *αυστηρή* (strict). Αν η PH δεν είναι αυστηρή, τότε θα υπάρχει κάποιο k για το οποίο $\mathrm{PH} = \Sigma_k^p$, οπότε λέμε ότι *η πολυωνυμική ιεραρχία καταρρέει στο k-οστό επίπεδο* (collapses at the k-th level).

Πολυωνυμική Ιεραρχία IV



Σχήμα: Πολυωνυμική ιεραρχία

Πολυωνυμική Ιεραρχία — Εναλλαγή Ποσοδεικτών Ι

Ένας εναλλακτικός τρόπος ορισμού της πολυωνυμικής ιεραρχίας είναι με την βοήθεια εναλλαγής ποσοδεικτών ($\exists$ και $\forall$). Επισημαίνουμε ότι, σε κάθε περίπτωση, οι ποσοδείκτες αναφέρονται σε αντικείμενα το μέγεθος των οποίων είναι φραγμένο από κάποιο πολυώνυμο p ως προς το μήκος της εισόδου.

Πρόταση

$L \in \Sigma_k^p$ ανν υπάρχει κατηγόρημα R υπολογιζόμενο σε πολυωνυμικό χρόνο και πολυώνυμο p που φράσσει το μέγεθος των αντικειμένων των ποσοδεικτών, τέτοια ώστε:

$$x \in L \iff \exists y_1 \forall y_2 \ldots Q y_k \; R(x, y_1, y_2, \ldots, y_k),$$

όπου $Q = \begin{cases} \exists, & k \text{ περιττό} \\ \forall, & k \text{ άρτιο} \end{cases}$.

Πολυωνυμική Ιεραρχία — Εναλλαγή Ποσοδεικτών

Παρομοίως, για την κλάση Π_k^p μόνο που τώρα η ακολουθία των ποσοδεικτών αρχίζει από $\forall$:

Πρόταση

$L \in \Pi_k^p$ ανν υπάρχει κατηγόρημα R υπολογιζόμενο σε πολυωνυμικό χρόνο και πολυώνυμο p που φράσσει το μέγεθος των αντικειμένων των ποσοδεικτών, τέτοια ώστε:

$$x \in L \iff \forall y_1 \exists y_2 \ldots Q y_k \; R(x, y_1, y_2, \ldots, y_k),$$

όπου $Q = \begin{cases} \forall, & k \text{ περιττό} \\ \exists, & k \text{ άρτιο} \end{cases}$.

Πολυωνυμική Ιεραρχία — Alternating TM I

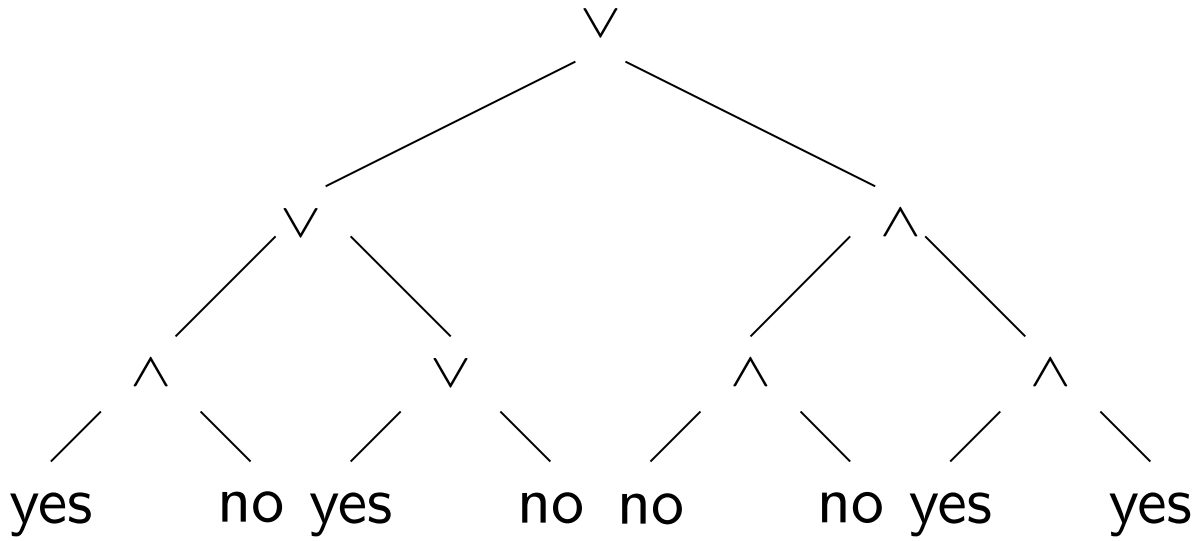
Η εναλλαγή των ποσοδεικτών στην πολυωνυμική ιεραρχία δίνει το έναυσμα για τον ορισμό της μηχανής Turing με εναλλασσόμενη λειτουργία. Αν θεωρήσουμε την δενδρική αναπαράσταση των υπολογισμών μίας NP μηχανής Turing, η μηχανή απαντά ναι, αν υπάρχει ένα τουλάχιστον φύλλο που λέει «ναι». Μπορούμε να θεωρήσουμε ότι κάθε κόμβος στο δένδρο υπολογίζει την διάζευξη ($\vee$) των αποτελεσμάτων από τα παιδιά του και την προωθεί στον γονέα του (τα φύλλα απλώς προωθούν προς τα πάνω), μέχρι το αποτέλεσμα να φτάσει στην ρίζα. Αντίστοιχα, μία coNP μηχανή Turing αποδέχεται όταν στην δενδρική αναπαράσταση όλα τα φύλλα λένε «ναι», οπότε μπορούμε να θεωρήσουμε ότι ο κάθε κόμβος συλλέγει τα αποτέλεσμα των παιδιών του και προωθεί στον γονέα του την σύζευξη ($\wedge$) των αποτελεσμάτων από τα παιδιά του, πάλι μέχρι το σωστό αποτέλεσμα να φτάσει στην ρίζα. Λέμε ότι όλοι οι κόμβοι στο δένδρο μίας NP μηχανής είναι τύπου $\vee$, ή $\exists$, ή υπαρξιακού. Λέμε ότι όλοι οι κόμβοι στο δένδρο μίας coNP μηχανής είναι τύπου $\wedge$, ή $\forall$, ή καθολικού.

Πολυωνυμική Ιεραρχία — Alternating TM II

Μία **εναλλασσόμενη μηχανή Turing** είναι μία μηχανή Turing στην οποία το αντίστοιχο υπολογιστικό δένδρο έχει εσωτερικούς κόμβους τύπου $\vee$ ή $\wedge$. Σημασία έχει το **πλήθος εναλλαγών τύπου**. Το μέγιστο **πλήθος εναλλαγών** (σε ένα μονοπάτι), που πιθανώς να είναι φραγμένο, αποτελεί μέτρο των δυνατοτήτων μίας τέτοιας μηχανής.

Για παράδειγμα, το υπολογιστικό δένδρο του παρακάτω σχήματος έχει πλήθος εναλλαγών τύπου ίσο με 2.

Πολυωνυμική Ιεραρχία — Alternating TM III



Σχήμα: Υπολογιστικό δένδρο με εναλλαγές

Πολυωνυμική Ιεραρχία — Alternating TM IV

Μπορεί να δειχθεί ότι η πολυωνυμική ιεραρχία είναι ακριβώς η κλάση των γλωσσών που γίνεται αποδεκτή από μηχανές Turing που έχουν φραγμένο πλήθος εναλλαγών. Πιο συγκεκριμένα:

- $L \in \Sigma_k^p$ ανν η L γίνεται αποδεκτή από μηχανή Turing που έχει το πολύ k εναλλαγές τύπου και αρχίζει με τύπο $\vee$.
- $L \in \Pi_k^p$ ανν η L γίνεται αποδεκτή από μηχανή Turing που έχει το πολύ k εναλλαγές τύπου και αρχίζει με τύπο $\wedge$.

Παραλληλοποιήσιμα προβλήματα Ι

Για να μελετήσουμε παράλληλους υπολογισμούς θα εισάγουμε ένα νέο μοντέλο υπολογισμού, το **κύκλωμα**.

Ένα κύκλωμα είναι ένας κατευθυνόμενος ακυκλικός γράφος, στον οποίο έχουμε ένα σύνολο κόμβων **εισόδου** και έναν κόμβο που είναι η **έξοδος**. Θεωρούμε ότι οι είσοδοι στο κύκλωμα είναι αληθοτιμές (ή αλλιώς οι τιμές 0 και 1) και κάθε εσωτερικός κόμβος αντιστοιχεί σε μία λογική συνάρτηση (ή αλλιώς πύλη — gate) με πλήθος εισόδων, όσες οι ακμές που καταλήγουν σε αυτόν. Αν ένα κύκλωμα C έχει n εισόδους του ενός bit: x_1, x_2, ..., x_n, τότε για κάθε $x \in \{0,1\}^n$, υπολογίζει μία μοναδική τιμή στην έξοδο, την $C(x)$. Αν $C(x) = 1$, λέμε ότι το κύκλωμα C αποδέχεται την είσοδο των n bit: x.

Παραλληλοποιήσιμα προβλήματα II

Η αναντιστοιχία του παραπάνω ορισμού αποδοχής, σε σχέση με την αποδοχή, ας πούμε, σε μία μηχανή Turing, είναι ότι ένα κύκλωμα, λόγω της αμετάβλητης φύσης του, μπορεί να απαντά για εισόδους μήκους ακριβώς n, ενώ μία μηχανή Turing (ή ένας αλγόριθμος γενικότερα) δέχεται εισόδους οποιουδήποτε μεγέθους. Για τον λόγο αυτό θα θεωρούμε μία οικογένεια κυκλωμάτων $\{C_1, C_2, \dots\}$, όπου κάθε C_n έχει n κόμβους εισόδου. Η **γλώσσα** που αποδέχεται μία οικογένεια κυκλωμάτων είναι η

$$L(C) = \{x \mid C_{|x|}(x) = 1\}.$$

Το πρόβλημα είναι ότι οι οικογένειες κυκλωμάτων (σε αντίθεση με τις μηχανές Turing) δεν είναι αριθμήσιμες.

Παραλληλοποιήσιμα προβλήματα III

Για να ξεπεράσουμε την παραπάνω δυσκολία, θα περιοριστούμε σε **ομοιόμορφες οικογένειες κυκλωμάτων** (uniform circuit families). Για αυτές υπάρχει αλγόριθμος και μάλιστα αποδοτικός ο οποίος δεδομένου n κατασκευάζει την αναπαράσταση του κυκλώματος C_n της οικογένειας. Μία επιλογή είναι οι P-ομοιόμορφες οικογένειες, που χρησιμοποιούν πολυωνυμικό αλγόριθμο κατασκευής. Επειδή όμως τα κυκλώματα χρησιμοποιούνται συνήθως για ορισμό κλάσεων χαμηλότερα από το P, θα χρησιμοποιήσουμε μία άλλη πιο περιορισμένη έννοια ομοιομορφίας:

Ορισμός

Μία οικογένεια κυκλωμάτων είναι $\mathrm{DLOGTIME}$-ομοιόμορφη αν υπάρχει μηχανή Turing (με δυνατότητα τυχαίας προσπέλασης στην ταινία εισόδου) που απαντά τις παρακάτω ερωτήσεις σε χρόνο $O(\log n)$:

- Υπάρχει σύνδεση από τον κόμβο u στον κόμβο v στο C_n;
- Τι είδους πύλη έχει ο κόμβος u;

Παραλληλοποιήσιμα προβλήματα IV

Το *μέγεθος* (size) ενός κυκλώματος είναι το πλήθος των κόμβων που περιέχει ο αντίστοιχος γράφος. Το μέγεθος αποτελεί μέτρο του κόστους κατασκευής του κυκλώματος και συνήθως δεν θεωρείται περισσότερο από πολυωνυμικό ως προς το μήκος της εισόδου. Το μέγεθος, όμως, δεν αποτελεί και πολύ καλό μέτρο του χρόνου υπολογισμού σε ένα κύκλωμα, επειδή εν γένει πολλές λογικές πύλες λειτουργούν παράλληλα. Οι πύλες που πρέπει να περιμένουν διαδοχικά ενδιάμεσα αποτελέσματα είναι αυτές που βρίσκονται σε κάθε μονοπάτι από μία είσοδο στην έξοδο. Για τον λόγο αυτό, πιο σημαντικό είναι το *βάθος* (depth) ενός κυκλώματος, που ορίζεται ως το μήκος του μακρύτερου μονοπατιού από είσοδο στην έξοδο.

Παραλληλοποιήσιμα προβλήματα V

Επίσης, σημαντικό είναι το είδος των λογικών πυλών που χρησιμοποιούνται σε κάθε κύκλωμα. Πιο συγκεκριμένα, θεωρούμε τα παρακάτω είδη λογικών πυλών:

1. Λογικές πύλες με περιορισμένο αριθμό εισόδων (bounded fan-in), καθώς και εναδικές πύλες $\neg$. Αρκεί να θεωρήσουμε δυαδικές πύλες $\wedge$ και $\vee$ (μαζί με τις εναδικές πύλες $\neg$).
2. Λογικές πύλες $\wedge$ και $\vee$ με απεριόριστο αριθμό εισόδων (unbounded fan-in), καθώς και εναδικές πύλες $\neg$.
3. Πύλες κατωφλίου (threshold) με απεριόριστο αριθμό εισόδων, καθώς και εναδικές πύλες $\neg$. Αρκεί, αντί για γενικές πύλες κατωφλίου, να χρησιμοποιήσουμε την πύλη πλειοψηφίας (majority gate), που δίνει στην έξοδο 1 αν και μόνον αν τουλάχιστον $r/2$ από τις r εισόδους της είναι 1.

Παραλληλοποιήσιμα προβλήματα VI

Με βάση τα παραπάνω μπορούμε να ορίσουμε τις παρακάτω κλάσεις:

Ορισμός

$(k \geqslant 0)$:

1. NC^k: η κλάση των γλωσσών που γίνονται αποδεκτές από DLOGTIME-ομοιόμορφες οικογένειες κυκλωμάτων πολυωνυμικού μεγέθους και βάθους $O(\log^k n)$, με χρήση πυλών του 1ου είδους (bounded fan-in).
2. AC^k: η κλάση των γλωσσών που γίνονται αποδεκτές από DLOGTIME-ομοιόμορφες οικογένειες κυκλωμάτων πολυωνυμικού μεγέθους και βάθους $O(\log^k n)$, με χρήση πυλών του 2ου είδους (unbounded fan-in).
3. TC^k: η κλάση των γλωσσών που γίνονται αποδεκτές από DLOGTIME-ομοιόμορφες οικογένειες κυκλωμάτων πολυωνυμικού μεγέθους και βάθους $O(\log^k n)$, με χρήση πυλών του 3ου είδους (threshold gates).
4. SC^k: η κλάση των γλωσσών που γίνονται αποδεκτές από DTM σε πολυωνυμικό χρόνο και σε $O(\log^k n)$ χώρο.

Παραλληλοποιήσιμα προβλήματα VII

Επίσης, ορίζεται $\mathrm{NC} = \bigcup_{k\in\mathbb{N}} \mathrm{NC}^k$. Η τελευταία κλάση λέγεται και Nick's Class, από τον Nicholas Pippenger, που ήταν από τους πρώτους που μελέτησε τέτοια κυκλώματα. Στην πραγματικότητα, πολλά άλλα μοντέλα παράλληλου υπολογισμού (π.χ. PRAM), εκτός από τα κυκλώματα, μπορούν να χρησιμοποιηθούν για τον ορισμό της κλάσης NC, κάτι που αποτελεί ένδειξη για την ευρωστία της κλάσης και την στενή σχέση της με τα παραλληλοποιήσιμα προβλήματα.

Το «A» στην AC^k οφείλεται στην εναλλαγή (alternation), αφού αποδεικνύεται ότι η κλάση AC^k, για $k \geqslant 1$, είναι ακριβώς οι γλώσσες που γίνονται αποδεκτές από εναλλασσόμενη μηχανή Turing που χρησιμοποιεί $O(\log n)$ χώρο και κάνει το πολύ $O(\log^k n)$ εναλλαγές. Το «T» στην TC^k προέρχεται από το threshold. Η ονομασία SC, "Steve's Class", προέρχεται από τον Steve Cook.

Παραλληλοποιήσιμα προβλήματα VIII

Πιο συγκεκριμένα οι κλάσεις σχετίζονται μεταξύ τους ως εξής:

Θεώρημα

Για κάθε $k \geqslant 0$, $\mathrm{NC}^k \subseteq \mathrm{AC}^k \subseteq \mathrm{TC}^k \subseteq \mathrm{NC}^{k+1}$.

Σε σχέση με άλλες γνωστές κλάσεις, ισχύει:

Θεώρημα

$\mathrm{Regular} \subseteq \mathrm{NC}^1 \subseteq \mathrm{L} = \mathrm{SC}^1 \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1$.
$\mathrm{Regular} \subset \mathrm{CF} \subset \mathrm{AC}^1$.

Δηλαδή το πρόβλημα του ελέγχου αν μία συμβολοσειρά παράγεται από δεδομένη γλώσσα χωρίς συμφραζόμενα (context free) ανήκει στην κλάση NC^2.

Προσεγγιστικοί Αλγόριθμοι Ι

Γνωρίζουμε ότι τα NP-δύσκολα προβλήματα δεν μπορούμε να τα λύσουμε:

1. Ακριβώς
2. Για όλα τα στιγμιότυπα
3. Σε πολυωνυμικό χρόνο

Προσεγγιστικοί Αλγόριθμοι II

- Αν αγνοήσουμε την συνθήκη (1), τότε έχουμε **Προσεγγιστικούς Αλγορίθμους**.
- Αν αγνοήσουμε την συνθήκη (2), τότε **μπορούμε να βρούμε μεγάλες υποκλάσεις στιγμιοτύπων του προβλήματος, στις οποίες το πρόβλημα να λύνεται σε πολυωνυμικό χρόνο**, αλλά επίσης είναι δυνατόν να αποφασίσουμε ντετερμινιστικά σε πολυωνυμικό χρόνο εάν η είσοδος ανήκει στην εν λόγω υποκλάση.
  - Ψευδοπολυωνυμικοί, Ισχυρά Πολυωνυμικοί
  - Παραμετροποίηση (πχ VERTEXCOVER(n, k))
    Παραμετρική Πολυπλοκότητα ($2^k n^c$, n^k κλπ)
- Αν αγνοήσουμε την συνθήκη (3), τότε κατατάσσουμε υπερπολυωνυμικές λύσεις με μεγαλύτερη ευκρίνεια.
  $1.003^n \leqslant 1.5^n \leqslant 2^n \leqslant 5^n \leqslant n! \leqslant n^n$
  $n^{\log\log n} \leqslant n^{\log n} \leqslant n^{\log^{13} n} \leqslant n^n$.

Προσεγγιστικοί Αλγόριθμοι III

Ένα πρόβλημα βελτιστοποίησης είναι: (I, S, v, goal).

- I: τα στιγμιότυπα του προβλήματος·
- S: μία συνάρτηση που αντιστοιχίζει σε κάθε στιγμιότυπο τις εφικτές λύσεις·
- v: η αντικειμενική συνάρτηση· αντιστοιχίζει σε κάθε εφικτή λύση, έναν θετικό ακέραιο·
- goal: min ή max, για πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης της αντικειμενικής συνάρτησης, αντίστοιχα.

Η τιμή της αντικειμενικής συνάρτησης για την βέλτιστη λύση με είσοδο x συμβολίζεται με $\text{OPT}(x)$ και είναι ίση με $\text{goal}\{v(y) \mid y \in S(x)\}$.

Προσεγγιστικοί Αλγόριθμοι IV

Επίσης, ορίζουμε για κάθε πρόβλημα βελτιστοποίησης το *αντίστοιχο* (underlying) πρόβλημα απόφασης ως εξής:

Δίδεται επιπλέον της εισόδου x ένα φράγμα k.
Ερώτηση: είναι $\mathrm{OPT}(x) \geqslant k$;
(για πρόβλημα μεγιστοποίησης – ανάλογα για πρόβλημα ελαχιστοποίησης.)

Παράδειγμα

Στο πρόβλημα MAX-CLIQUE το στιγμιότυπο είναι ένας γράφος x, οι εφικτές λύσεις είναι όλοι οι πλήρεις υπογράφοι του x (κλίκες), η αντικειμενική συνάρτηση είναι το πλήθος των κόμβων της κλίκας και goal = max. Το αντίστοιχο πρόβλημα απόφασης είναι το γνωστό CLIQUE.

Προσεγγιστικοί Αλγόριθμοι V

Ορίζουμε τις παρακάτω βασικές κλάσεις πολυπλοκότητας για προβλήματα βελτιστοποίησης:

Ορισμός

NPO: η κλάση των προβλημάτων βελτιστοποίησης, για τα οποία το αντίστοιχο πρόβλημα απόφασης είναι στο NP (με την προϋπόθεση ότι υπάρχουν εφικτές λύσεις για κάθε στιγμιότυπο).

Ορισμός

PO: η κλάση των προβλημάτων βελτιστοποίησης, για τα οποία το αντίστοιχο πρόβλημα απόφασης είναι στο P.

Προσεγγιστικοί Αλγόριθμοι VI

Πολλά προβλήματα βελτιστοποίησης είναι NP-δύσκολα. Για αυτό αναζητούμε προσεγγιστικούς πολυωνυμικούς αλγόριθμους που επιλύουν τέτοια προβλήματα.

Ορισμός

Ένας πολυωνυμικός αλγόριθμος M είναι ρ-προσεγγιστικός για ένα πρόβλημα μεγιστοποίησης αν για κάθε $x \in I$ επιστρέφει μια λύση $M(x) \in S(x)$ τέτοια ώστε:

$$\frac{v(M(x))}{\mathrm{OPT}(x)} \leqslant \rho.$$

Αντίστοιχα ορίζεται ρ-προσεγγιστικός αλγόριθμος για πρόβλημα ελαχιστοποίησης.

Προσεγγιστικοί Αλγόριθμοι VII

Οι πιο γνωστές υποκλάσεις της NPO, εκτός της PO, είναι οι εξής:

- poly-APX: περιέχει προβλήματα για τα οποία υπάρχει $p(n)$-προσεγγιστικός αλγόριθμος για κάποιο πολυώνυμο p (όπου n είναι το μήκος της εισόδου: $n = |x|$).
- log-APX: περιέχει προβλήματα για τα οποία υπάρχει $\log n$-προσεγγιστικός αλγόριθμος (όπου n είναι το μήκος της εισόδου: $n = |x|$).
- APX: περιέχει προβλήματα για τα οποία υπάρχει ρ-προσεγγιστικός αλγόριθμος για κάποια σταθερά $\rho > 0$.
- PTAS: περιέχει προβλήματα για τα οποία υπάρχει *πολυωνυμικού χρόνου προσεγγιστικό σχήμα*, δηλαδή $(1+\varepsilon)$-προσεγγιστικός αλγόριθμος *για κάθε* σταθερά $\varepsilon > 0$.
- FPTAS: περιέχει προβλήματα για τα οποία υπάρχει *πλήρως πολυωνυμικού χρόνου προσεγγιστικό σχήμα*, δηλαδή $(1+\varepsilon)$-προσεγγιστικός αλγόριθμος για κάθε σταθερά $\varepsilon > 0$, που επιπλέον ο χρόνος που χρειάζεται είναι πολυωνυμικός και ως προς το $1/\varepsilon$.

Προσεγγιστικοί Αλγόριθμοι VIII