Εισαγωγή στην Επιστήμη των Υπολογιστών

4ο εξάμηνο Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

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

2η ενότητα: Αυτόματα και Τυπικές Γραμματικές

Στάθης Ζάχος
Άρης Παγουρτζής

Επιμέλεια: Πάνος Χείλαρης, Βαγγέλης Μπαμπάς, Γεωργία Καούρη

Αυτόματα

Στο κεφάλαιο αυτό θα επιδιώξουμε να ταξινομήσουμε τα υπολογιστά σύνολα (γλώσσες) ανάλογα με το είδος του **αυτομάτου** (*αφηρημένης υπολογιστικής συσκευής*) που μπορεί να τα αναγνωρίσει.

Γενικά μπορούμε να κατατάξουμε τα αυτόματα ανάλογα με διάφορα κριτήρια όπως, π.χ., ως προς

- **αιτιότητα** (ντετερμινιστικά, μη ντετερμινιστικά, πιθανοτικά)
- **μέγεθος** (συγκεκριμένο, αυξανόμενο, άπειρο)
- **είσοδο/έξοδο** (διακριτές δηλ. ψηφιακά, συνεχείς δηλ. αναλογικά)
- **γενικό ρολόι** σε περίπτωση παραλληλίας (συγχρονισμένα, ασύγχρονα)
- **αριθμό καταστάσεων** (πεπερασμένο, άπειρο)
- **λειτουργία** (αναγνωριστικά (*acceptors*), γεννητικά (*generators*), υπολογιστικά (*transducers*)).

Αυτόματα

Στις πιο απλές συσκευές, που ονομάζονται **μηχανισμοί**, η συνάρτηση μετάβασης (*transition function*) δ έχει ως όρισμα μία κατάσταση q_i και ως τιμή μια άλλη κατάσταση q_j. Δεν υπάρχει δηλαδή είσοδος και έξοδος. Υπολογιστική ακολουθία σε αυτήν την περίπτωση είναι μια ακολουθία από καταστάσεις:

$$q_i \to q_j \to q_k \to q_l \cdots$$

Αν τώρα η συσκευή διαβάζει είσοδο, σύμβολο προς σύμβολο, από αριστερά προς τα δεξιά και ανάλογα αλλάζει καταστάσεις, τότε έχουμε ένα **αναγνωριστή πεπερασμένων καταστάσεων** (*FSA: finite state acceptor*) για το οποίον η συνάρτηση μετάβασης είναι της μορφής

$$\delta\colon (q_i, a) \to q_j,$$

όπου $a \in \Sigma$ και το Σ το αλφάβητο εισόδου.

Αυτόματα

Αν προσθέσουμε έξοδο σε ένα *FSA*, δηλαδή $\delta\colon (q_i, a) \to (q_j, b)$, όπου $b \in \Delta$ και Δ το αλφάβητο εξόδου, τότε έχουμε τη λεγόμενη μηχανή πεπερασμένων καταστάσεων (*FSM: finite state machine*). Οι FSA και FSM εμφανίζονται συχνά ως χρήσιμα εργαλεία στο λογικό, ψηφιακό σχεδιασμό. Προσθέτοντας στο αυτόματό μας μνήμη υπό μορφή στοίβας (*stack*) έχουμε πολύ περισσότερες ικανότητες: το αυτόματο ονομάζεται τότε **αυτόματο στοίβας** (*PDA: push-down automaton*). Αν αντί στοίβας έχουμε απεριόριστη δυνατότητα μνήμης υπό μορφή ταινίας, τότε έχουμε τη γνωστή μας **μηχανή Turing** (*TM*). **Γραμμικά περιορισμένο αυτόματο** (*LBA: linearly bounded automaton*) είναι μια μηχανή Turing που όμως μπορεί να χρησιμοποιήσει ταινία που το μήκος της είναι μια γραμμική συνάρτηση του μήκους της εισόδου.

Τυπικές γλώσσες

Οι γλώσσες προγραμματισμού είναι προφανώς **τυπικές γλώσσες** (*formal languages*) που έχουν αυστηρό συντακτικό ορισμό. Οι τυπικές γλώσσες μπορούν να ταξινομηθούν ανάλογα με τις τυπικές γραμματικές που τις παράγουν.Στη θεωρία τυπικών γλωσσών οι πρωταρχικές έννοιες είναι τα **σύμβολα** (ως αντικείμενα) και η **παράθεση** (ως πράξη).

Μια λέξη ή πρόταση ή συμβολοσειρά ή string είναι μια πεπερασμένου μήκους ακολουθία συμβόλων. Ένα **αλφάβητο** Σ είναι ένα πεπερασμένο σύνολο συμβόλων. Το μήκος του string w συμβολίζεται με $|w|$. Το κενό string συμβολίζεται με ε. Η παράθεση των strings x και y συμβολίζεται με xy. Άλλοι χρήσιμοι όροι είναι **πρόθεμα** (*prefix*), **υποσυμβολοσειρά** (*substring*), **υπακολουθία** (*subsequence*), **κατάληξη** (*suffix*), **αντίστροφη** (*reversal*),**παλινδρομική** ή **καρκινική** (*palindrome*).

Τυπικές γλώσσες

Ισχύουν $x\varepsilon = \varepsilon x = x$ για όλα τα strings x και $|\varepsilon| = 0$. Το string x^k μπορεί να οριστεί με πρωταρχική αναδρομή:

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

Αν Σ είναι ένα αλφάβητο τότε Σ^* είναι το σύνολο όλων των strings από το Σ. Μια γλώσσα L από το Σ δεν είναι παρά κάποιο υποσύνολο του Σ^*.

Τυπικές γλώσσες

Παραδείγματα (με αλφάβητο $\Sigma = \{a, b\}$):

- $L_1 = \{w \in \Sigma^* \mid w \text{ αρχίζει με } a\}$
- $L_2 = \{w \in \Sigma^* \mid w \text{ περιέχει ζυγό αριθμό από } a\}$
- $L_3 = \{w \in \Sigma^* \mid w \text{ είναι παλινδρομική }\}$

Τυπικές γραμματικές

Ορισμός 3.2.1. Μια *τυπική γραμματική* G αποτελείται από:

- ένα αλφάβητο V από *μη τερματικά* σύμβολα (μεταβλητές),
- ένα αλφάβητο T από *τερματικά* σύμβολα (σταθερές), τ.ω. $V \cap T = \emptyset$,
- ένα πεπερασμένο σύνολο P από *κανόνες παραγωγής*, δηλαδή διατεταγμένα ζεύγη (α, β) όπου $\alpha, \beta \in (V \cup T)^*$ και $\alpha \neq \varepsilon$
(Σύμβαση: γράφουμε $\alpha \to \beta$ αντί για (α, β)),
- ένα *αρχικό σύμβολο* (ή αξίωμα) $S \in V$.

Σύμβαση για τη χρήση γραμμάτων:

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

Τυπικές γραμματικές

Σύντμηση: Γράφουμε $\alpha \to \beta \mid \gamma \mid \delta$ ως ένα κανόνα στο P αντί για τους τρεις κανόνες $\alpha \to \beta$, $\alpha \to \gamma$, $\alpha \to \delta$ στο P.

Ορισμός 3.2.2.

- Λέμε ότι το $\gamma_1\alpha\gamma_2$ *παράγει* το $\gamma_1\beta\gamma_2$ και το 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 $\alpha \overset{*}{\Rightarrow} \beta$ (με λόγια: «το α παράγει το β») σημαίνει ότι υπάρχει μια ακολουθία: $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \alpha_k \Rightarrow \beta$.
- *Ως γλώσσα που παράγεται από τη γραμματική* G ορίζουμε την $L(G) := \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$.
- Δύο γραμματικές G_1, G_2 ονομάζονται ισοδύναμες αν $L(G_1) = L(G_2)$.

Τυπικές γραμματικές

Παράδειγμα 3.2.3. Έστω η γραμματική

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

Μια πιθανή ακολουθία παραγωγής είναι η:
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
Η γλώσσα που παράγεται από τη γραμματική είναι η $L(G) = \{a^n b^n \mid n \in N\}$

Ιεραρχία Γραμματικών Chomsky

Ο Noam Chomsky (1956) ταξινόμησε τις τυπικές γραμματικές σε μια ιεραρχία σύμφωνα με τον τύπο των κανόνων παραγωγής τους:

τύπου 0: *γενικές* γραμματικές (general, phrase structure, semi-Thue). Μορφή κανόνων παραγωγής: $\alpha \to \beta$, $\alpha \neq \varepsilon$.

τύπου 1: γραμματικές *με συμφραζόμενα* ή μονοτονικές (context sensitive, monotonic). Μορφή: $\alpha \to \beta$, όπου $|\alpha| \leq |\beta|$ (μπορεί επιπλέον να επιτρέπεται $S \to \varepsilon$)

τύπου 2: γραμματικές *χωρίς συμφραζόμενα* (context free). Μορφή: $A \to \alpha$, όπου $A \in V$

τύπου 3: *κανονικές* γραμματικές (regular). Η μορφή των κανόνων παραγωγής τους είναι *δεξιογραμμική*: $A \to w$, $A \to wB$ ή *αριστερογραμμική*: $A \to w$, $A \to Bw$, όπου $w \in T^*$, $A, B \in V$.

Ιεραρχία Γραμματικών Chomsky

Όπως θα δούμε στη συνέχεια, αυτή είναι μια γνήσια ιεράρχηση, δηλαδή ισχύει

τύπου 3 $\subset$ τύπου 2 $\subset$ τύπου 1 $\subset$ τύπου 0

Οι γλώσσες τύπου 0, 1, 2, 3 μπορούν να αναγνωριστούν από αυτόματα που έχουμε ήδη συζητήσει: από μηχανές Turing (Turing Machines - TM), γραμμικά περιορισμένα αυτόματα (linearly bounded automata - LBA), αυτόματα στοίβας (push down automata - PDA) και αναγνωριστές πεπερασμένων καταστάσεων (finite state acceptors - FSA), αντίστοιχα.

Ορισμός DFA

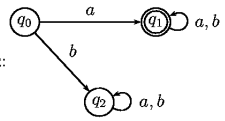
Ορισμός 3.3.1. Τυπικά ένα DFA είναι μία πεντάδα $M = (Q, \Sigma, \delta, q_0, F)$, όπου:

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

Παράδειγμα DFA

Παράδειγμα 3.3.2. $L_1 = \{w \in \Sigma^* \mid w \text{ αρχίζει από } a\}$

Διάγραμμα:



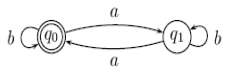
Πίνακας:

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

Χρησιμοποιούμε έναν επιπλέον κύκλο για να δείξουμε τις τελικές καταστάσεις.

Παράδειγμα γλώσσας με DFA και γλώσσας χωρίς DFA

Παράδειγμα 4.3.3. $L_2 = \{w \in \Sigma^* \mid w \text{ περιέχει άρτιο πλήθος από } a\}$



	a	b
(q_0)	q_1	q_0
q_1	q_0	q_1

Παράδειγμα 4.3.4. $L_3' = \{w \in \Sigma^* \mid w \text{ παλίνδρομη αρτίου μήκους}\}$, δηλαδή $L_3' = \{ww^R \mid w \in \Sigma^*, w^R = \text{αντίστροφη της } w\}$. Δεν υπάρχει DFA που αποδέχεται την L_3'.

Επέκταση ορισμού DFA

Ορισμός 3.3.5. $\tilde{\delta} : Q \times \Sigma^* \to Q$ όπου

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

Ο πιο πάνω ορισμός είναι αναδρομικός, ή, πιο συγκεκριμένα, είναι ορισμός σύμφωνα με το σχήμα της *πρωταρχικής αναδρομής*. Παρατηρούμε ότι

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

Δηλαδή, για σύμβολα $a \in \Sigma$ η $\tilde{\delta}$ ταυτίζεται με τη δ. Αυτό συμβολίζεται ως

$$\tilde{\delta} \restriction_{Q \times \Sigma} = \delta$$

Γλώσσα αποδεκτή από DFA

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

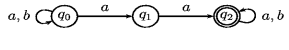
Άσκηση: Δείξτε ότι $\tilde{\delta}(q, uv) = \tilde{\delta}(\tilde{\delta}(q, u), v)$, όπου $u, v \in \Sigma^*$.

Μη ντετερμινιστικά πεπερασμένα αυτόματα (NFA)

- NFA: *μη ντετερμινιστικό πεπερασμένο αυτόματο*. Σε κάθε μετάβαση υπάρχει επιλογή της επόμενης κατάστασης από ένα σύνολο πιθανών νομίμων καταστάσεων.
- NFA_ε: *μη ντετερμινιστικό πεπερασμένο αυτόματο με ε-κινήσεις*. Το πεπερασμένο αυτόματο ενδέχεται να αλλάζει την κατάστασή του χωρίς να μετακινείται η κεφαλή στην ταινία εισόδου.

Παράδειγμα NFA

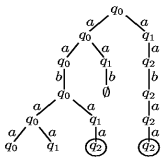
Παράδειγμα 3.3.7. NFA για $L_4 := \{w \in \Sigma^* \mid w \text{ περιέχει δύο συνεχόμενα } a\}$



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

Ένας υπολογισμός σε ένα NFA δεν είναι απλώς μία γραμμική (νόμιμη) ακολουθία καταστάσεων, αλλά ένα *υπολογιστικό δένδρο* (κάθε κλάδος είναι μία νόμιμη ακολουθία καταστάσεων).

Το δένδρο υπολογισμού για το παραπάνω παράδειγμα για είσοδο *aabaa*:



Η συμβολοσειρά *aabaa* γίνεται αποδεκτή, επειδή υπάρχει τουλάχιστον ένα νόμιμο μονοπάτι που την αποδέχεται.

Τυπικός ορισμός NFA

Στα μη ντετερμινιστικά αυτόματα, για κάθε είσοδο και κατάσταση, μπορεί να υπάρχει καμμία, μία ή πολλές πιθανές επόμενες καταστάσεις. Αυτό εκφράζεται στον ορισμό ενός NFA από το γεγονός ότι η συνάρτηση μετάβασης δ έχει ως πεδίο τιμών το *δυναμοσύνολο* του Q ($\text{Pow}(Q)$).

- Σ: ένα πεπερασμένο αλφάβητο εισόδου,
- $\delta : Q \times \Sigma \to \text{Pow}(Q)$: η συνάρτηση μετάβασης
- $q_0 \in Q$: η αρχική κατάσταση και
- $F \subseteq Q$: το σύνολο των τελικών καταστάσεων.

Γλώσσα αποδεκτή από NFA

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

Ισοδυναμία DFA και NFA

Ισοδυναμία DFA και NFA. Όπως φαίνεται από τον ορισμό της δ ενός NFA, ένα DFA είναι μια «υποπερίπτωση» ενός NFA. Παρ' όλα αυτά, τα NFA δεν μας παρέχουν περισσότερες δυνατότητες υπολογισμού από ότι τα DFA. Αυτό αποδεικνύει το παρακάτω θεώρημα:

Θεώρημα 3.3.13. *(Rabin - Scott)*

Έστω M ένα NFA. Τότε $\exists$ DFA $M' : L(M) = L(M')$

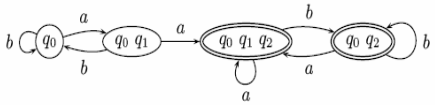
Ισοδυναμία DFA και NFA

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

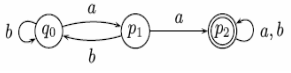
Μένει να αποδειχθεί ότι $L(M) = L(M')$ με την βοήθεια του ισχυρισμού: $\tilde{\delta}'(q_0', w) = \tilde{\delta}(q_0, w)$. (Αφήνεται ως άσκηση· χρησιμοποιήστε επαγωγή.) □

DFA

Παράδειγμα 4.3.21. DFA για L_4 (βλ. παραδ. 4.3.7):



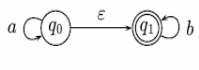
Ελάχιστο DFA για L_4:



NFA$_\varepsilon$

Μη ντετερμινιστικά αυτόματα με ε-κινήσεις - NFA$_\varepsilon$

Παράδειγμα 4.3.14. NFA$_\varepsilon$ για $L_5 := \{a^*b^*\} = \{a^n b^m \mid n, m \in \mathbb{N}\}$



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

Τυπικός ορισμός NFA$_\varepsilon$

- Q: ένα πεπερασμένο σύνολο από καταστάσεις,
- Σ: ένα πεπερασμένο αλφάβητο εισόδου,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathrm{Pow}(Q)$: η συνάρτηση μετάβασης
- $q_0 \in Q$: η αρχική κατάσταση και
- $F \subseteq Q$: το σύνολο των τελικών καταστάσεων.

ε-κλείσιμο

Ορισμός 4.3.15. Ως ε-κλείσιμο: $Q \to \mathrm{Pow}(Q)$ ορίζουμε το

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

Παρατηρούμε ότι πάντα $q \in \varepsilon$-κλείσιμο(q). Επεκτείνουμε τον ορισμό αυτό:

Ορισμός 4.3.16. Ως ε-κλείσιμο: $\mathrm{Pow}(Q) \to \mathrm{Pow}(Q)$ ορίζουμε το

$$\varepsilon\text{-κλείσιμο}(P) = \bigcup_{q \in P} \varepsilon\text{-κλείσιμο}(q)$$

Παρατηρούμε επιπλέον ότι ε-κλείσιμο(ε-κλείσιμο$(P)) = \varepsilon$-κλείσιμο(P).

Επέκταση ορισμού NFA$_\varepsilon$

Ορισμός 3.3.17. $\delta : Q \times \Sigma^* \to \mathrm{Pow}(Q)$ όπου

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

Επιπλέον, επεκτείνουμε την $\tilde{\delta}$ στην $\delta^\times$, συμπεριλαμβάνοντας στο πεδίο ορισμού το δυναμοσύνολο του Q.

Ορισμός 3.3.18. $\delta^\times : \mathrm{Pow}(Q) \times \Sigma^* \to \mathrm{Pow}(Q)$ όπου

$$\delta^\times(P, w) = \{p \in \tilde{\delta}(q, w) | q \in P\} = \bigcup_{q \in P} \tilde{\delta}(q, w)$$

Ισχύει: $\delta^\times \restriction_{Q \times \Sigma^*} = \tilde{\delta}$.

Γλώσσα αποδεκτή από NFA$_\varepsilon$

Ορισμός 3.3.19. Έχουμε:

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

Ισοδυναμία NFA και NFA$_\varepsilon$. Και πάλι, μπορεί να θεωρήσει κανείς ότι τα NFA είναι μια «υποπερίπτωση» των NFA$_\varepsilon$. Όμως, όπως και πριν, τα NFA$_\varepsilon$ δεν έχουν περισσότερες δυνατότητες υπολογισμού από τα NFA, όπως αποδεικνύει το επόμενο θεώρημα, και άρα και από τα DFA.

Ισοδυναμία NFA και NFA$_\varepsilon$

Θεώρημα 3.3.20. *Έστω M ένα NFA$_\varepsilon$ τότε* $\exists$ NFA $M' : L(M) = L(M')$

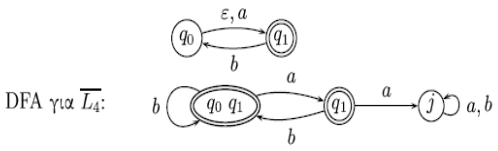
Απόδειξη. Ορίζουμε το NFA $M' = (Q, \Sigma, q_0, F', \delta')$ όπου

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

Πλέον, προκειμένου να ισχύει $L(M) = L(M')$, αρκεί να αποδειχθεί ο ισχυρισμός: $\forall w \in \Sigma^* - \{\varepsilon\}$: $\tilde{\delta}'(q_0, w) = \tilde{\delta}(q_0, w)$. (Άσκηση.) □

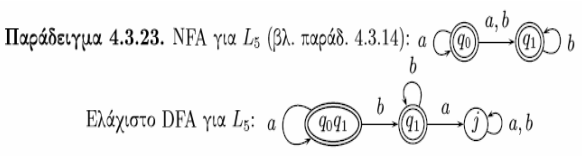
Παράδειγμα ισοδυναμίας NFA$_\varepsilon$ και DFA

Παράδειγμα 4.3.22. NFA$_\varepsilon$ για $\overline{L_4}$ (δηλαδή "όχι δύο συνεχόμενα a"):



Παράδειγμα ισοδυναμίας NFA και DFA

Παράδειγμα 4.3.23. NFA για L_5 (βλ. παράδ. 4.3.14):



Κανονικές Παραστάσεις (Regular Expressions)

Ορισμός 3.3.24. Έστω L, L_1, L_2 γλώσσες επί του ίδιου αλφαβήτου Σ.

- $L_1L_2 := \{uv \mid u \in L_1 \wedge v \in L_2\}$: 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 4.3.25. Κανονική παράσταση είναι:

$\emptyset$: παριστάνει το κενό σύνολο·

ε: παριστάνει το $\{\varepsilon\}$·

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

$(r + s)$: παριστάνει το $R \cup S$, όπου r, s κανονικές παραστάσεις που παριστάνουν τα R, S αντιστοίχως·

(rs): παριστάνει το RS, όπου r, s κανονικές παραστάσεις που παριστάνουν τα R, S αντιστοίχως·

(r^*): παριστάνει το R^*, όπου r κανονική παράσταση που παριστάνει το R.

Σύμβαση: Μπορούμε να περιορίσουμε τις παρενθέσεις αν χρησιμοποιήσουμε την ακόλουθη προτεραιότητα των τελεστών: $*$, παράθεση, ένωση.

Παράδειγμα 3.3.26.

$L_1 = a(a + b)^*$
$L_2 = (b^*ab^*a)^*b^*$
L_3 δεν είναι δυνατόν να παρασταθεί με κανονική παράσταση
$L_4 = (a + b)^*aa(a + b)^*$ (τουλάχιστον δύο συνεχόμενα a)
$\overline{L_4} = (a + \varepsilon)(ba + b)^*$ (όχι συνεχόμενα a)
$L_\varepsilon = a^*b^*$

Ισοδυναμία Κανονικών Παραστάσεων και Αυτομάτων

Θεώρημα 3.3.27. *Μία γλώσσα μπορεί να παρασταθεί με κανονική παράσταση αν $L = L(M)$ για κάποιο πεπερασμένο αυτόματο M.*

Ιδέα απόδειξης.

$\Rightarrow$ Επαγωγή στη δομή της κανονικής παράστασης. Κατασκευή NFA$_\varepsilon$. (Άσκηση.)

$\Leftarrow$ Θεωρούμε ότι το DFA είναι της μορφής $M = (Q, \Sigma, \delta, q_1, F)$, όπου τα στοιχεία του Q είναι αριθμημένα κατά αύξουσα σειρά και η αρχική κατάσταση είναι η q_1, δηλαδή $Q = \{q_1, \ldots, q_n\}$, όπου $n = |Q|$. Ορίζουμε:

$$R_{ij}^k = \{w \mid \tilde{\delta}(q_i, w) = q_j \text{ και } \forall x \text{ πρόθεμα του } w \text{ με } x \neq w, \varepsilon\colon \tilde{\delta}(q_i, x) = q_l \Rightarrow l \leq k\}$$

Δηλαδή, R_{ij}^k είναι το σύνολο των συμβολοακολουθιών που οδηγούν από την κατάσταση q_i στην q_j χωρίς να περνούν από οποιαδήποτε κατάσταση με δείκτη μεγαλύτερο από k. Προφανώς, αφού δεν υπάρχει κατάσταση με δείκτη μεγαλύτερο από n, το R_{ij}^n περιέχει όλες τις συμβολοακολουθίες από την q_i στην q_j. Μπορούμε να υπολογίσουμε το R_{ij}^k αναδρομικά, σύμφωνα με μια ιδέα των Floyd και Warshall, ως εξής

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

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

Τέλος, αρκεί να παρατηρήσουμε ότι $L(M) = \bigcup_{q_j \in F} R_{1j}^n$. (Οι λεπτομέρειες της απόδειξης αφήνονται ως άσκηση.) □

3.3.4 Παρόμοια αυτόματα

Στα 2-way FA η κεφαλή μπορεί να κινείται δεξιά και αριστερά. Παρ' όλα αυτά, για κάθε 2-way FA υπάρχει ισοδύναμο DFA (δύσκολη απόδειξη).

Πεπερασμένες μηχανές με έξοδο έχουμε τριών τύπων, με αντίστοιχες συναρτήσεις εξόδου λ:

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

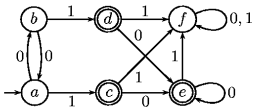
Και τρεις τύποι είναι ισοδύναμοι (εύκολη απόδειξη).

Ελαχιστοποίηση DFA

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

- Το p είναι διακρίσιμο από το q εάν υπάρχει ένα x τέτοιο ώστε $\delta(p, x)$ είναι στο F και $\delta(q, x)$ δεν είναι ή αντίστροφα.
- Φτιάχνουμε ένα πίνακα για να συγκρίνουμε κάθε ζεύγος καταστάσεων. Βάζουμε ένα X σε κάθε θέση του πίνακα κάθε φορά που ανακαλύπτουμε ότι δύο καταστάσεις δεν είναι ισοδύναμες. Αρχικά εγγράφουμε X σε όλα τα ζεύγη που προφανώς διακρίνονται γιατί η μία είναι τελική και η άλλη δεν είναι. Μετά προσπαθούμε να δούμε αν διακρίνονται δύο καταστάσεις, διότι από αυτές με ένα σύμβολο a οδηγούμαστε σε διακρίσιμες καταστάσεις. Επαναλαμβάνουμε την πιο πάνω προσπάθεια ώσπου να μην προστίθεται κανένα X πια στον πίνακα. Τα υπόλοιπα ζευγάρια είναι μη διακρίσιμα και συνεπώς συγχωνεύσιμα.

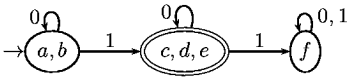
Παράδειγμα 3.3.28. Έστω το αυτόματο M που φαίνεται στο σχήμα, το οποίο αποδέχεται την γλώσσα $L = 0^*10^*$.



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

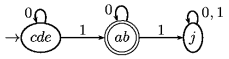
b					
ⓒ	X_1	X_1			
ⓓ	X_1	X_1			
ⓔ	X_1	X_1			
f	X_2	X_2	X_1	X_1	X_1
	a	b	ⓒ	ⓓ	ⓔ

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

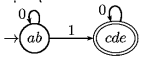


Μέθοδος Beckmann

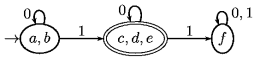
1. Κατασκεύασε το καθρέφτισμα του FA: δηλαδή ανάστρεψε τη φορά των τόξων. Αντάλλαξε τελικές με αρχικές καταστάσεις.
2. Κατασκεύασε DFA ισοδύναμο με το προκύπτον.



3. Ανακατασκεύασε το καθρέφτισμα.

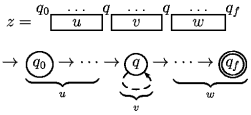


4. Ανακατασκεύασε ισοδύναμο DFA. Αυτό τώρα είναι το ελάχιστο DFA που βρήκαμε και παραπάνω, δηλ.



Pumping Lemma. Αν μία γλώσσα είναι κανονική τότε την αποδέχεται ένα DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ με κάποιο συγκεκριμένο αριθμό από καταστάσεις, έστω n, δηλαδή $|Q| = n$. Έστω μία λέξη z που γίνεται αποδεκτή από το αυτόματο M και η οποία έχει μήκος μεγαλύτερο από n.

Καθώς επεξεργαζόμαστε το z, το αυτόματο μας M πρέπει να περάσει ξανά από μία κατάσταση, γιατί δεν υπάρχουν περισσότερες από n καταστάσεις (αρχή του περιστερώνα, pigeonhole principle). Έχουμε ότι ένα μονοπάτι που αποδέχεται το z είναι το ακόλουθο:



Το uvw γίνεται αποδεκτό, όπως επίσης και το uw, ή το $uvvw$ ή γενικά το uv^iw για οποιοδήποτε $i \in \mathbb{N}$. Δηλαδή το v μπορεί να επαναληφθεί όσες φορές θέλουμε.

Λήμμα 4.3.29 (Pumping Lemma). *Εάν L είναι regular τότε:*
$\exists n \in \mathbb{N}, \forall z \in L$ *με* $|z| \geq n, \exists u, v, w \in \Sigma^*$:
$[z = uvw \ \wedge |uv| \leq n \ \wedge |v| \geq 1 \wedge \forall i \in \mathbb{N}\ (uv^iw \in L)]$

Χρησιμοποιώντας το λήμμα δείχνουμε ότι ένα δοσμένο σύνολο δεν είναι regular.

Μέθοδος απόδειξης ότι μια γλώσσα δεν είναι regular

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 $|\ z\ | \geq n$.
4. Ο αντίπαλος (PL) σπάει το z σε u, v και w που ικανοποιούν τους περιορισμούς $|uv| \leq n$ και $|v| \geq 1$.
5. Φτάνεις σε αντίφαση δείχνοντας ότι για κάθε u, v, w που καθορίζονται από τον αντίπαλο, υπάρχει ένα i για το οποίο uv^iw 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.

Παράδειγμα 4.3.30. $L = \{a^kb^k \mid k \in \mathbb{N}\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$ δεν είναι regular.

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

Άτοπο

Γραμματικές Χωρίς Συμφραζόμενα

Παράδειγμα 3.4.1. Έστω η γραμματική

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

Μια πιθανή ακολουθία παραγωγών είναι η:

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

Η γλώσσα που παράγεται από την G_1 είναι η $L(G_1) = \{a^nb^n | n \in \mathbb{N}^*\}$.

Παράδειγμα 3.4.2. $G_2 = (V, T, P, S)$ με $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *\}$, $V = \{S\}$ και το P περιέχει τους κανόνες:

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

Παράδειγμα 3.4.3. $G_3 = (V, T, P, S)$, $V = \{S, A, B\}$, $T = \{a, b\}$ και το P περιέχει τους κανόνες:

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

Μια πιθανή ακολουθία παραγωγών της πιο πάνω γραμματική είναι:

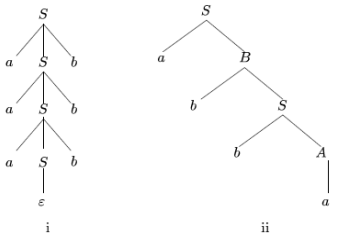
$$S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba$$

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

Συντακτικά Δένδρα

Ορισμός 3.4.4. Έστω $G = \{V, T, P, S\}$ μια c.f. γραμματική. Ένα δένδρο είναι συντακτικό δένδρο της G αν

1. Κάθε κόμβος του δένδρου έχει μια *επιγραφή*, η οποία είναι ένα σύμβολο στο $V \cup T \cup \{\varepsilon\}$.
2. Η επιγραφή της ρίζας είναι το S.
3. Αν ένας κόμβος είναι εσωτερικός και έχει επιγραφή A, τότε το A πρέπει να είναι στοιχείο του V.
4. Αν ο κόμβος n έχει επιγραφή A και οι κόμβοι $n_1, n_2, \ldots, n_k$ είναι παιδιά του n, σε διάταξη από αριστερά προς τα δεξιά, με επιγραφές $X_1, X_2, \ldots, X_k$ αντίστοιχα, τότε ο $A \to X_1 X_2 \ldots X_k$ πρέπει να είναι κανόνας παραγωγής στο P.
5. Αν ένας κόμβος έχει επιγραφή ε, τότε είναι φύλλο και είναι το μοναδικό παιδί του γονέα του.



Σχήμα 4.1: Παραδείγματα συντακτικών δένδρων

Ονομάζουμε *φύλλωμα (leaf string)* τη συμβολοακολουθία που προκύπτει από τα φύλλα ενός δένδρου, αν τα διατρέξουμε από το αριστερότερο προς το δεξιότερο. Ορίζουμε ως *A-δένδρο* ενός συντακτικού δέντρου ένα υποδέντρο το οποίο περιλαμβάνει ως ρίζα έναν κόμβο με επιγραφή A καθώς και όλους τους απογόνους αυτού του κόμβου και τις μεταξύ τους ακμές.

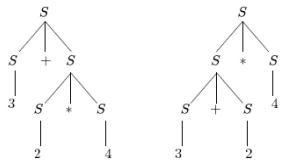
Θεώρημα 3.4.5. *Έστω $G(V, T, P, S)$ μια c.f. γραμματική. Τότε $S \stackrel{*}{\Rightarrow} \alpha$ ανν υπάρχει συντακτικό δένδρο με φύλλωμα το α.*

Η απόδειξη της κατεύθυνσης «$\Leftarrow$» γίνεται με επαγωγή ως προς τον αριθμό των εσωτερικών κόμβων, ενώ της «$\Rightarrow$» με επαγωγή ως προς των αριθμό των βημάτων της ακολουθίας παραγωγών (άσκηση).

Διφορούμενες γραμματικές (ambiguous grammars)

Ορισμός 4.4.6. Μια γραμματική G ονομάζεται *διφορούμενη (ambiguous)* αν υπάρχουν δύο συντακτικά δένδρα που να έχουν ως φύλλωμα ένα $w \in L(G)$.

Για παράδειγμα, η συμβολοακολουθία $3 + 2 * 4$ ανήκει στην $L(G_2)$, του παραδείγματος 4.4.2, σελίδα 65 και υπάρχουν δύο συντακτικά δένδρα που αντιστοιχούν σε αυτήν, όπως φαίνεται στο σχήμα 4.2. Παρ'όλα αυτά, μπορούμε να κατασκευάσουμε μια γραμματική που είναι ισοδύναμη με την G_2 και η οποία δεν είναι διφορούμενη (άσκηση).



Σχήμα 4.2: Συντακτικά δένδρα διφορούμενης γραμματικής

Υπάρχουν όμως και γλώσσες χωρίς συμφραζόμενα, για τις οποίες όλες οι γραμματικές που τις παράγουν είναι αναγκαστικά διφορούμενες:

Ορισμός 4.4.7. Μια γλώσσα χωρίς συμφραζόμενα ονομάζεται *εγγενώς διφορούμενη* (inherently ambiguous) αν όλες οι γραμματικές που την παράγουν είναι διφορούμενες.

Παράδειγμα 4.4.8. Η γλώσσα χωρίς συμφραζόμενα $\{a^i b^j c^k \mid i = j \vee j = k\}$ είναι εγγενώς διφορούμενη.

Απλοποίηση Γραμματικών

Είναι δυνατόν να απλοποιήσουμε μία γραμματική χωρίς συμφραζόμενα ως εξής: Κρατάμε μόνον τα σύμβολα που χρειάζονται (για παράδειγμα εξαλείφουμε όλα τα μη τερματικά που δεν παράγουν συμβολοακολουθίες με τερματικά και όλα τα σύμβολα που δεν μπορούν να εμφανισθούν σε καμία παραγωγή που ξεκινάει από το αρχικό σύμβολο). Επίσης, μπορούμε να εξαλείψουμε κανόνες του τύπου $A \to B$ (unit productions).

Επιπλέον, αν το ε δεν ανήκει στην L, δεν χρειάζονται κανόνες παραγωγής της μορφής $A \to \varepsilon$ (ε-productions).

Κανονικές Μορφές

Θεώρημα 3.4.9 (Κανονικής μορφής Chomsky, CNF). *Κάθε c.f. γλώσσα χωρίς το ε παράγεται από μια γραμματική στην οποία όλες οι παραγωγές είναι της μορφής $A \to BC$ ή $A \to a$. όπου A, B, C μεταβλητές και a τερματικό.*

Θεώρημα 3.4.10 (Κανονικής μορφής Greibach, GNF). *Κάθε c.f. γλώσσα χωρίς το ε παράγεται από μια γραμματική στην οποία όλες οι παραγωγές είναι της μορφής $A \to a\alpha$, όπου $\alpha \in V^*$, $a \in T$.*

Από γραμματικές στις παραπάνω κανονικές μορφές, προκύπτουν σχετικά απλούστερα συντακτικά δένδρα: για την CNF τα συντακτικά δένδρα έχουν πάντοτε διακλάδωση βαθμού δύο αν προκύπτουν μεταβλητές, αλλιώς από μία μεταβλητή προκύπτει ένα μόνον φύλλο (τερματικό σύμβολο), ενώ για την GNF, τα αριστερά παιδιά κάθε κόμβου είναι πάντοτε τερματικά σύμβολα.

Αλγόριθμος CYK

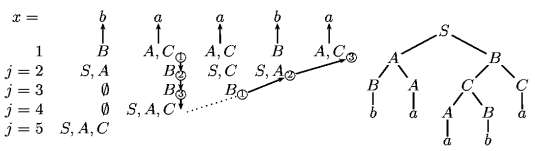
Μπορούμε να εκμεταλλευτούμε αυτές τις ιδιότητες των συντακτικών δένδρων για να επιλύσουμε ταχύτερα ορισμένα προβλήματα όπως αν κάποια συμβολοσειρά x ανήκει στην γλώσσα που παράγει μία γραμματική: Δεδομένης γραμματικής χωρίς συμφραζόμενα G, όχι απαραίτητα σε κανονική μορφή, υπάρχει μηχανιστικός αλγόριθμος 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):

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

Παράδειγμα 3.4.11. 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 3.3 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 3.3: Εκτέλεση αλγορίθμου CYK

Αυτόματα Στοίβας (PushDown Automata – PDA)

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.

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).

Ορισμός 3.4.12. 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: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Pow(Q \times \Gamma^*)$ (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: $F = \emptyset$).

Αντίστοιχα ορίζουμε και την γλώσσα που αποδέχεται ένα PDA:

1. Γλώσσα που αποδέχεται σε τελική κατάσταση $L_f(M)$
2. Γλώσσα που αποδέχεται με κενή στοίβα $L_e(M)$

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.

Σχέση c.f. γλωσσών και PDA

Θεώρημα 3.4.13. *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. *Η L είναι γλώσσα χωρίς συμφραζόμενα (context free).*

Παραλείπεται η λεπτομερής απόδειξη.

Γενικές Γραμματικές

Τύπου 0: *γενικές γραμματικές (general or unrestricted grammars)*, semi-Thue, phrase structure

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

Παράδειγμα 3.5.1. $L = \{a^{2^n} \mid n \in N\}$

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

Θεώρημα 4.5.2. *Τα ακόλουθα είναι ισοδύναμα:*

1. *η L γίνεται αποδεκτή από μία Turing μηχανή (βλέπε κεφάλαιο 6, ενότητα 6.1)*
2. *$L = L(G)$, όπου G είναι γενική γραμματική*

Χωρίς απόδειξη. Μία τέτοια γλώσσα λέγεται και *αναδρομικά αριθμήσιμη* (recursively enumerable)· βλέπε ορισμό 7.1.3 στην σελίδα 92.

Γραμματικές Με Συμφραζόμενα (context sensitive)

Τύπου 1: *γραμματικές με συμφραζόμενα (context sensitive grammars, c.s.)*

Παραγωγές: $\alpha \to \beta$, με $\alpha \neq \varepsilon$, $|\alpha| \leq |\beta|$ (noncontracting grammar, non-decreasing, not ε string)

Η ονομασία context sensitive οφείλεται στην παρακάτω εναλλακτική περιγραφή αυτών των γραμματικών: Κάθε c.s. γραμματική μπορεί να τεθεί σε κανονική μορφή στην οποία όλοι κανόνες παραγωγής είναι της μορφής:

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

Παράδειγμα 4.6.1. $1^n 0^n 1^n$. C.s. γραμματικές:

	ή	ή
$S \to 1Z1$	$S \to WZW$, $W \to 1$	$S \to WZW$, $W \to 1$
$Z \to 0 \mid 1Z0A$	$Z \to 0 \mid WZZA$	$Z \to 0 \mid WZZA$
$A0 \to 0A$	$AZ \to ZA$	$AZ \to HZ, HZ \to HA,$
		$HA \to ZA$
$A1 \to 11$	$AW \to WW$	$AW \to WW$

Άλλα παραδείγματα τέτοιων γλωσσών: $\{a^n b^n c^n\}$, $\{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$, $\{ww \mid w \in \Sigma^*\}$, $0^n 1^n 0^n 1^n$.

Σχέση c.s. γλωσσών και LBA

Γραμμικά φραγμένο αυτόματο (Linearly bounded automaton (LBA)): Είναι μία μη-ντετερμινιστική μηχανή Turing (T.M.) της οποίας η κεφαλή είναι περιορισμένη να κινείται μόνον στο τμήμα της ταινίας που περιέχει την είσοδο.

Θεώρημα 4.6.2. *Τα ακόλουθα είναι ισοδύναμα (L χωρίς ε):*

1. *η L γίνεται αποδεκτή από LBA*
2. *η L είναι c.s.*

Ιεραρχία κλάσεων γλωσσών

Θεώρημα 3.6.3 (Ιεραρχίας). *(Θεωρούμε γλώσσες που δεν περιέχουν το ε.)*

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

Κανονικές Γραμματικές

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

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

Θεώρημα 3.7.1. *Τα κανονικά σύνολα παράγονται από κανονικές (δεξιο- ή αριστερογραμμικές γραμματικές)*