Εισαγωγή στην Επιστήμη των Υπολογιστών

4ο εξάμηνο Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

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

3η ενότητα: Αυτόματα και Τυπικές Γραμματικές

Στάθης Ζάχος

Άρης Παγουρτζής

Συνεργασία: Κωστής Σαγώνας

Επιμέλεια:

Πάνος Χείλαρης, Βαγγέλης Μπαμπάς, Γεωργία Καούρη

Αυτόματα

- Τρόπος κωδικοποίησης αλγορίθμων.
- Τρόπος περιγραφής συστημάτων πεπερασμένων καταστάσεων:

  Συστήματα που έχουν εσωτερικές καταστάσεις και προκαθορισμένο τρόπο μετάβασης από μία κατάσταση σε άλλη με βάση την τρέχουσα κατάσταση και την είσοδο (συνήθως ενέργεια κάποιου χρήστη). Μπορεί να έχουν και έξοδο.
- Εφαρμογές σε πλήθος επιστημονικών πεδίων

Παράδειγμα: πωλητής καφέ (i)

Προδιαγραφές

- Δύο είδη καφέ: ελληνικός ή φρέντο.
- Κόστος καφέ: 40 λεπτά.
- Επιτρέπονται κέρματα 10, 20, ή 50 λεπτών.

Παράδειγμα: πωλητής καφέ (ii)

Σχεδίαση του συστήματος

- Εσωτερικές καταστάσεις: q0, q1, q2, q3, q4, q5
  (qi: εκφράζει ότι έχουν δοθεί μέχρι στιγμής 10*i λεπτά).
- Πιθανές είσοδοι (ενέργειες): P1, P2, P5 (ρίψη κέρματος 10, 20, ή 50 λεπτών), K1, K2 (πάτημα κουμπιού 1 για ελληνικό καφέ, ή 2 για φρέντο).
- Πιθανές έξοδοι: E1, E2, E3, E4, E5
  (Ei: εκφράζει ότι επιστρέφονται 10*i λεπτά),
  ΕΛ (παροχή ελληνικού καφέ), ΦΡ (παροχή φρέντο).

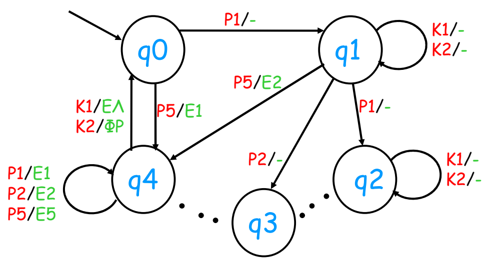
Παράδειγμα: πωλητής καφέ (iii)

Πίνακας καταστάσεων: δείχνει ποια είναι η επόμενη κατάσταση και η έξοδος για κάθε συνδυασμό τρέχουσας κατάστασης και εισόδου. Αρχική κατάσταση: q0.

Είσοδος / Κατάστ.	P1	P2	P5	K1	K2
q0	q1, -	q2, -	q4, E1	q0, -	q0, -
q1	q2, -	q3, -	q4, E2	q1, -	q1, -
q2	q3, -	q4, -	q4, E3	q2, -	q2, -
q3	q4, -	q4, E1	q4, E4	q3, -	q3, -
q4	q4, E1	q4, E2	q4, E5	q0, ΕΛ	q0, ΦΡ

Παράδειγμα: πωλητής καφέ (iv)

Διάγραμμα καταστάσεων: παρέχει τις ίδιες πληροφορίες με τον πίνακα καταστάσεων με πιο εποπτικό τρόπο. Αρχική κατάσταση: q0 (σημειώνεται με βέλος).



Αυτόματα (i)

Ένα αυτόματο έχει μερικές εσωτερικές καταστάσεις q_0, q_1, q_7, q_{15}, ... , και μια συνάρτηση μετάβασης δ που καθορίζει την επόμενη κατάσταση του αυτομάτου με βάση την τρέχουσα κατάσταση και την συμβολοσειρά εισόδου.

Αυτόματα (ii)

- Μηχανισμοί: χωρίς είσοδο – έξοδο.

  $\delta(q_i) = q_j$

  εκτέλεση: $q_0 \rightarrow q_j \rightarrow q_k \rightarrow q_m \ \cdots$

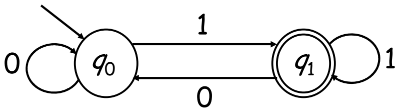
- Αυτόματα πεπερασμένων καταστάσεων (FSA): με είσοδο που γίνεται αποδεκτή ή απορρίπτεται.

  $\delta(q_i\,,\, a) = q_j$

  (a είναι ένα από τα σύμβολα της εισόδου)

  εκτέλεση: *εξαρτάται από την είσοδο*

Παράδειγμα FSA για αναγνώριση περιττών αριθμών



- q_0: το τελευταίο ψηφίο που διαβάστηκε είναι διάφορο του 1.
- q_1: το τελευταίο ψηφίο που διαβάστηκε είναι ίσο με 1.
- η q_0 είναι αρχική κατάσταση ενώ η q_1 είναι κατάσταση αποδοχής (ή τελική).

Ισχυρότερα αυτόματα

- Μηχανές πεπερασμένων καταστάσεων (FSM): είναι FSA με έξοδο: $\delta(q_i\,,\, a) = (q_j\,,\, \beta)$

  Αυτόματα στοίβας (PDA, pushdown automata): έχουν πολύ περισσότερες δυνατότητες καθώς έχουν μνήμη (σε μορφή στοίβας).
- Μηχανές Turing (TM): έχουν ακόμη περισσότερες δυνατότητες καθώς έχουν απεριόριστη μνήμη (σε μορφή ταινίας).
- Γραμμικά περιορισμένα αυτόματα (LBA): είναι TM με μνήμη περιορισμένη γραμμικά ως προς το μήκος της εισόδου.

Αυτόματα και τυπικές γλώσσες

- 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.

  Κάθε αυτόματο (χωρίς έξοδο) *αναγνωρίζει* μια τυπική γλώσσα.

Τυπικές γλώσσες

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

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

Ορισμός DFA

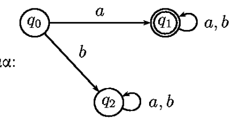
Ορισμός 3.3.1. Τυπικά ένα DFA είναι μία πεντάδα $M = (Q, \Sigma, \delta, q_0, F)$, όπου:

- 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:



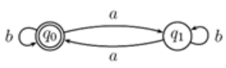
Πίνακας:

	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 συνάρτησης δ DFA

Δέχεται ως ορίσματα μια κατάσταση q και μια συμβολοσειρά w και δίνει την κατάσταση όπου θα βρεθεί το αυτόματο αν ξεκινήσει από την q και διαβάσει την w.

Ορισμός 4.3.5. $\delta : Q \times \Sigma^* \to Q$ όπου

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

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

Γλώσσα αποδεκτή από DFA

Ορισμός 4.3.6. Έχουμε:

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

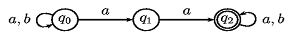
Άσκηση: Δείξτε ότι $\delta(q, uv) = \delta(\delta(q, u), v)$, όπου $u, v \in \Sigma^*$.

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$_\varepsilon$: *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

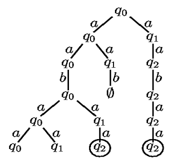
Παράδειγμα 3.3.7. NFA για $L_4 := \{w \in \Sigma^* \mid w$ περιέχει δύο συνεχόμενα $a\}$



	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 ($\mathrm{Pow}(Q)$).

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

Γλώσσα αποδεκτή από NFA

Ορισμός 4.3.12. Έχουμε:

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

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.

Ισοδυναμία DFA και NFA (i)

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 (ii)

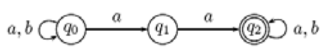
Έστω το NFA M = $(Q, \Sigma, q_0, F, \delta)$.

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

- $Q' = \mathrm{Pow}(Q)$, δηλαδή οι καταστάσεις του M' είναι όλα τα υποσύνολα καταστάσεων του M.
- $q'_0 = \{q_0\}$,
- $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$, 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 $r \in R\}$, δηλαδή δ'(R, a) είναι το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το M ξεκινώντας από οποιαδήποτε κατάσταση του R και διαβάζοντας a.

Παράδειγμα μετατροπής NFA σε DFA

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	$\{q_2\}$	$\{q_2\}$

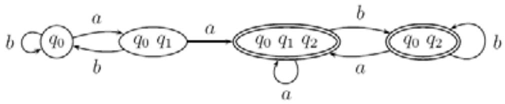
DFA για τη γλώσσα L_4:

$Q' \backslash \Sigma$	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_2\}$	$\emptyset$
$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Παράδειγμα μετατροπής NFA σε DFA (ii)

DFA για τη γλώσσα L_4:

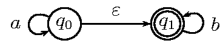
$Q'\backslash\Sigma$	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
$\{q_1\}$	$\{q_2\}$	$\emptyset$
$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0,q_1\}$	$\{q_0,q_1,q_2\}$	$\{q_0\}$
$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_2\}$
$\{q_1,q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_2\}$



NFA$_\varepsilon$

Τα μη ντετερμινιστικά αυτόματα με ε-κινήσεις (NFA$_\varepsilon$) επιτρέπουν και ορισμένες 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 4.3.15. NFA$_\varepsilon$ 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	$\emptyset$	$\{q_1\}$	$\{q_1\}$

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.

Γλώσσα αποδεκτή από NFA$_\varepsilon$

Ορισμός 4.3.20. Έχουμε:

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

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, χρησιμοποιώντας ενδεχομένως και ε-κινήσεις όπου αυτό επιτρέπεται.

ε-κλείσιμο

Για να ορίσουμε τυπικά την αποδοχή σε NFA$_\varepsilon$, αλλά και για να δείξουμε την ισοδυναμία με DFA, χρειαζόμαστε την έννοια του ε-κλεισίματος μιας κατάστασης q, που είναι το σύνολο των καταστάσεων στις οποίες μπορεί να φτάσει το αυτόματο ξεκινώντας από την q και χρησιμοποιώντας μόνο ε-κινήσεις.

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

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

Ισοδυναμία NFA$_\varepsilon$ και DFA

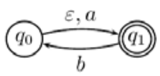
Έστω το NFA$_\varepsilon$ M = (Q,Σ,q_0,F,δ).

Ένα ισοδύναμο DFA M' = (Q',Σ,q'_0,F',δ'), ορίζεται ως εξής:

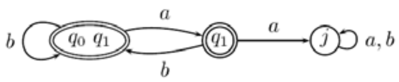
- Q' = Pow(Q), δηλαδή οι καταστάσεις του M' είναι όλα τα υποσύνολα καταστάσεων του M.
- q'_0 = ε-κλείσιμο(q_0),
- F' = {R ∈ Q' | R ∩ F ≠ ∅}, δηλαδή μια κατάσταση του M' είναι τελική αν περιέχει μια τελική κατάσταση του M.
- δ'(R, a) = {q ∈ Q | q ∈ ε-κλείσιμο(δ(r, a)) για r ∈ R}, δηλαδή δ'(R, a) είναι το σύνολο των καταστάσεων όπου μπορεί να βρεθεί το M ξεκινώντας από οποιαδήποτε κατάσταση του R και διαβάζοντας a και χρησιμοποιώντας στη συνέχεια ε-κινήσεις.

Παράδειγμα ισοδυναμίας NFA$_\varepsilon$ και DFA

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



DFA για $\overline{L_4}$:



Ελαχιστοποίηση DFA

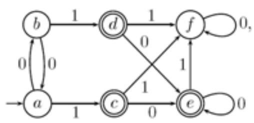
- Εξαλείφουμε τις απρόσιτες καταστάσεις
- Σημειώνουμε ως διακρίσιμες δύο καταστάσεις αν:
  - η μία είναι τελική ενώ η άλλη όχι
  - οδηγούν με ένα ή περισσότερα σύμβολα σε διακρίσιμες καταστάσεις (βλ. παρακάτω για αναλυτική μέθοδο)
- Συγχωνεύουμε ισοδύναμες (= μη διακρίσιμες) καταστάσεις.

Εύρεση διακρίσιμων καταστάσεων

- Κατασκευάζουμε 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, δηλαδή ισοδύναμα (και επομένως συγχωνεύσιμα).

Παράδειγμα ελαχιστοποίησης DFA

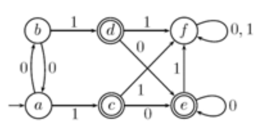
Παράδειγμα 4.3.30. Έστω το αυτόματο M που φαίνεται στο σχήμα, το οποίο αποδέχεται την γλώσσα $L = 0^*10^*$.



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

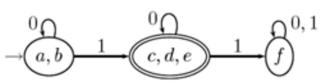
b					
c	X_1	X_1			
d	X_1	X_1			
e	X_1	X_1			
f	X_2	X_2	X_1	X_1	X_1
	a	b	c	d	e

Παραδ. ελαχιστοποίησης DFA (συν.)



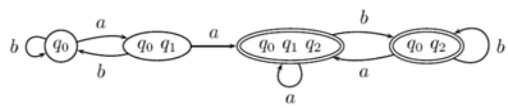
b					
c	X_1	X_1			
d	X_1	X_1			
e	X_1	X_1			
f	X_2	X_2	X_1	X_1	X_1
	a	b	c	d	e

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



2ο παραδ. ελαχιστοποίησης DFA

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



Ελάχιστο DFA για L_4:

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. Κάθε αυτόματο (χωρίς έξοδο) *αναγνωρίζει* μια τυπική γλώσσα.
- Τυπικές γραμματικές: άλλος τρόπος περιγραφής τυπικών γλωσσών. Κάθε τυπική γραμματική *παράγει* μια τυπική γλώσσα.

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.
- vw = παράθεση λέξεων v και w.
- Άλλες έννοιες: πρόθεμα (prefix), κατάληξη (suffix), υποσυμβολοσειρά (substring), αντίστροφη (reversal), παλινδρομική ή καρκινική (palindrome).

Τυπικές γλώσσες (συν.)

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

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

Ορισμός 4.1.1. 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^*.

Παράδειγμα γραμματικής για την γλώσσα των περιττών αριθμών

S → X 1
X → X 0
X → X 1
X → ε

S: το αρχικό σύμβολο
X: μη τερματικό σύμβολο
0,1: τερματικά σύμβολα
ε: η κενή συμβολοσειρά

Τα S και X αντικαθίστανται με βάση τους κανόνες

Τυπικές γραμματικές (i)

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:

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

Τυπικές γραμματικές (ii)

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 $L(G_1) = L(G_2)$.

Παράδειγμα τυπικής γραμματικής

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

Πιθανή ακολουθία παραγωγής:

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

Γλώσσα που παράγεται:

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

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

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

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.

Η θεωρία αυτή είναι ιδιαίτερα χρήσιμη στα εξής πεδία: Ψηφιακή Σχεδίαση, Γλώσσες Προγραμματισμού, Μεταγλωττιστές, Τεχνητή Νοημοσύνη, Θεωρία Πολυπλοκότητας κ.ο.κ. Ιστορικά σημαντικοί ερευνητές: *Chomsky, Backus, Rabin, Scott, Kleene, Greibach, κ.α.*

Κανονικές Εκφράσεις (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 κανονικών εκφράσεων

$\emptyset$: 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.

Παραδείγματα κανονικών εκφράσεων

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)
$L_5=a^*b^*$

Ισοδυναμία Κανονικών Παραστάσεων και Αυτομάτων

Θεώρημα 4.3.28. *Μία γλώσσα μπορεί να παρασταθεί με κανονική παράσταση ανν $L = L(M)$ για κάποιο πεπερασμένο αυτόματο M.*

Ιδέα απόδειξης.

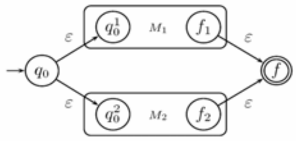
$\Rightarrow$ Επαγωγή στην δομή της κανονικής παράστασης. Έστω r η κανονική παράσταση.

1. Επαγωγική Βάση:

$r = \varepsilon$: , $r = \emptyset$: , $r = a \in \Sigma$:

2. Επαγωγικό βήμα. Έστω ότι για r1, r2 έχουμε αυτόματα M1, M2, με τελικές καταστάσεις f1, f2:

Περίπτωση α: $r = r_1 + r_2$

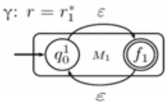


$L(M) = L(M_1) \cup L(M_2)$

Περίπτωση β: $r = r_1 r_2$

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

Περίπτωση γ: $r = r_1^*$



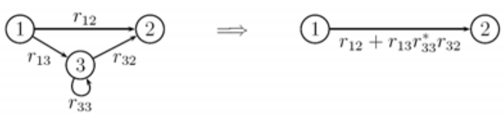
$L(M) = L(M_1)^*$

Ισοδυναμία Κανονικών Παραστάσεων και Αυτομάτων (συν.)

"<=" :

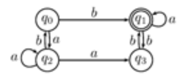
Κατασκευή κανονικής παράστασης από FA

Απαλείφουμε ενδιάμεσες καταστάσεις σύμφωνα με το σχήμα:

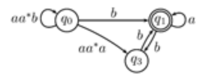


Παράδειγμα κατασκευής κανονικής παράστασης από FA

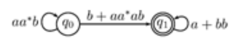
Παράδειγμα 4.3.29. Έστω πως έχουμε το ακόλουθο FA:



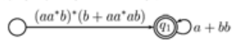
Επιλέγουμε να διαγράψουμε τη κατάσταση q_2, οπότε ενημερώνουμε τις μεταβάσεις των άλλων καταστάσεων:



Επιλέγουμε να διαγράψουμε τη κατάσταση q_3, οπότε ενημερώνουμε τα υπόλοιπα:



Επιλέγουμε να διαγράψουμε τη κατάσταση q_0:



Τελικά, η κανονική έκφραση που προκύπτει είναι:

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

Κανονικές Γραμματικές

Οι κανονικές γραμματικές είναι γραμματικές όπου όλοι οι κανόνες είναι της μορφής:

- Δεξιά γραμμικοί (right linear)

  $A \to wB$ ή $A \to w$

- Αριστερά γραμμικοί (left linear)

  $A \to Bw$ ή $A \to w$

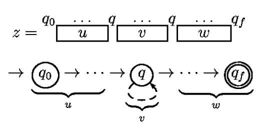
(όπου w είναι μια ακολουθία από τερματικά σύμβολα της γλώσσας)

Οι κανονικές γλώσσες είναι γλώσσες που παράγονται από κανονικές γραμματικές

Γλώσσες που δεν είναι regular

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.

Pumping Lemma

Έστω άπειρη κανονική γλώσσα L. Τότε:

- υπάρχει ένας φυσικός n ώστε
- για κάθε w ∈ L με μήκος τουλάχιστον n
- υπάρχει «σπάσιμο» του w σε x, y, z, w = x y z
  δηλαδή w = xyz, με $|xy| \leq n$ και $|y| > 0$
- ώστε για κάθε i = 0, 1, 2, ... :

$$x\,y^i\,z \in L$$

Μη Κανονικές Γλώσσες

Χρήση του Pumping Lemma για να δείξουμε ότι μια γλώσσα L δεν είναι κανονική:

- έστω ότι η L είναι κανονική
- τότε υπάρχει ένα μήκος n (pumping length)
- επιλέγουμε ένα w ∈ L με μήκος τουλάχιστον n
- και επιχειρηματολογούμε ότι για κάθε τρόπο γραφής w = xyz που ικανοποιεί τις συνθήκες του λήμματος, το «φούσκωμα» του y δίνει μία λέξη που δεν είναι στη γλώσσα L
- άρα καταλήγουμε σε άτοπο.

Παράδειγμα χρήσης

Θεώρημα: Η γλώσσα
L = {w: w έχει τον ίδιο αριθμό από 0 και 1}
δεν είναι κανονική.

Απόδειξη:

έστω n το μήκος «φουσκώματος» του L

επιλέγουμε $w = 0^n 1^n$

$$w = \underbrace{000000000\ldots0}_{p}\underbrace{111111111\ldots1}_{p}$$

w = xyz, με $|y| > 0$ και $|xy| \leq n$.

Παράδειγμα χρήσης (συν.)

- 3 ενδεχόμενα (αν "ξεχάσουμε" ότι $|xy| \leq n$):

$$w = \underbrace{00000}_{x}\underbrace{0000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots011}_{x}\underbrace{11111}_{y}\underbrace{11\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{11111111\ldots1}_{z}$$

- στις πρώτες 2 περιπτώσεις, το «φούσκωμα» του y δίνει λέξεις που δεν είναι στη γλώσσα L
- η 3η περίπτωση θέλει περισσότερη προσοχή...

Παράδειγμα χρήσης (συν.)

- Ας θυμηθούμε τώρα τη συνθήκη: $|xy| \leq n$
- επειδή $w = 0^n 1^n$, η περίπτωση αυτή δεν είναι δυνατή:

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{11111111\ldots1}_{z}$$

- άρα, και σε αυτήν την περίπτωση καταλήγουμε σε άτοπο
- και κατά συμπέρασμα η L δεν είναι κανονική.

Δεύτερο παράδειγμα χρήσης

Θεώρημα: $L = \{0^i 1^j : i > j\}$ δεν είναι κανονική.

Απόδειξη:

έστω n το μήκος «φουσκώματος» του L

επιλέγουμε $w = 0^{n+1} 1^n$

$$w = \underbrace{000000000\ldots0}_{n+1}\underbrace{1111111\ldots1}_{n}$$

w = xyz, με $|y| > 0$ και $|xy| \leq n$.

Δεύτερο παράδειγμα χρήσης (συν.)

Ένα μόνο ενδεχόμενο:

$$w = \underbrace{00000}_{x}\underbrace{0000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

το φούσκωμα του y δίνει λέξεις της γλώσσας (;)

από πρώτη άποψη αυτό φαίνεται προβληματικό…

το λήμμα ορίζει ότι για κάθε $i \geq 0$, $xy^iz \in L$

Όμως, η λέξη xy^0z δεν είναι στην L. Άρα η L δεν είναι κανονική.

Εναλλακτική χρήση Pumping Lemma (adversary argument)

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 χρήσης Pumping Lemma με adversary argument

Παράδειγμα 4.3.32. $L = \{a^kb^k \mid k \in \mathbb{N}\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$ 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:

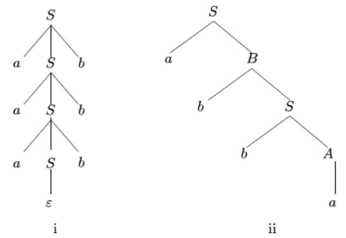
$$S \Rightarrow aB \Rightarrow abS \Rightarrow abbA \Rightarrow abba$$

Αν και δεν είναι προφανές, $L(G_3) = \{w \in T^+ \mid w$ έχει ίσο αριθμό a 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 $A \to X_1X_2\ldots X_k$ 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.

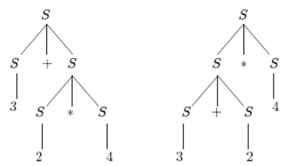
Θεώρημα 3.4.5. *Έστω $G(V,T,P,S)$ μια c.f. γραμματική. Τότε $S \overset{*}{\Rightarrow} \alpha$ ανν υπάρχει συντακτικό δένδρο με φύλλωμα το α.*

Η απόδειξη της κατεύθυνσης «⇐» γίνεται με επαγωγή ως προς τον αριθμό των εσωτερικών κόμβων, ενώ της «⇒» 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):

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. *H 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:

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

context

Παράδειγμα 4.6.1. $1^n0^n1^n$. 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: $\{a^nb^nc^n\}$, $\{a^ib^jc^k \mid 0 \leq i \leq j \leq k\}$, $\{ww \mid w \in \Sigma^*\}$, $0^n1^n0^n1^n$.

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.)*

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