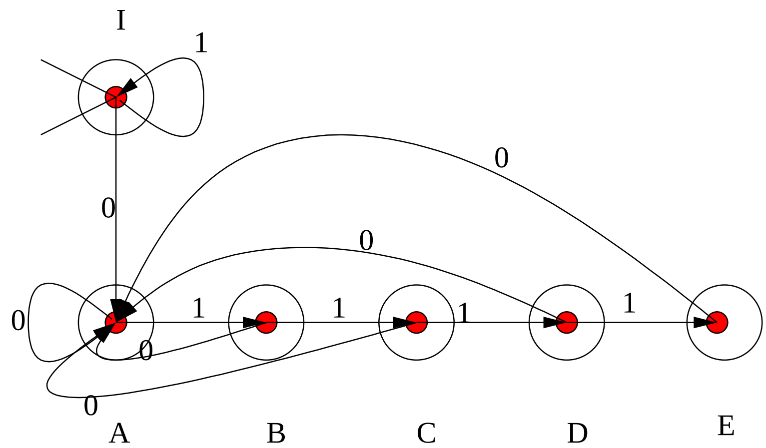
Τυπικές Γλώσσες και Υπολογισιμότητα



Μιχάλης Κολουντζάκης
Τμήμα Μαθηματικών
Πανεπιστήμιο Κρήτης
Λεωφόρος Κνωσού
714 09 Ηράκλειο

E-mail: `kolount@member.ams.org`
WWW: `http://fourier.math.uoc.gr/~ mk`

20 Σεπτεμβρίου 2004

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή και Μαθηματικό Υπόβαθρο

1.1 Αλφάβητα, λέξεις και γλώσσες

Ορισμός 1. (Αλφάβητο)
Αλφάβητο είναι ένα οποιοδήποτε μη κενό πεπερασμένο σύνολο, του οποίου τα στοιχεία ονομάζουμε *σύμβολα* ή *γράμματα* .

Παράδειγμα 1. Το *δυαδικό αλφάβητο* $\Sigma_2 = \{0, 1\}$. Υπό μία έννοια αυτό είναι το πλέον ενδιαφέρον αλφάβητο. Ένας λόγος είναι ότι αυτό είναι το αλφάβητο που χρησιμοποιείται στους σημερινούς υπολογιστές. Ο σημαντικότερος όμως λόγος είναι ότι το Σ_2 μπορεί να "μιμηθεί" αποτελεσματικά οποιοδήποτε άλλο αλφάβητο. Το τι σημαίνει αυτό ελπίζουμε να γίνει ξεκάθαρο αργότερα.

Παράδειγμα 2. Το αλφάβητο $\Sigma_{GR} = \{A, B, \Gamma, \ldots, \alpha, \beta, \gamma, \ldots, \omega, \ldots\}$ περιλαμβάνει όλα τα γράμματα και σημεία στίξης της Ελληνικής γλώσσας.

Ορισμός 2. (Λέξη)
Λέξη πάνω από ένα αλφάβητο Σ λέγεται μια πεπερασμένη ακολουθία

$$\alpha = \alpha_1 \ldots \alpha_n$$

γραμμάτων $\alpha_i \in \Sigma$, ενδεχομένως και κενή (δηλ. με $n = 0$).

Ο αριθμός $n \geq 0$ λέγεται *μήκος* της λέξης α και συμβολίζεται με $|\alpha|$.

Η κενή λέξη συμβολίζεται πάντα με το ϵ.

Με Σ^* συμβολίζουμε το σύνολο όλων των λέξεων πάνω από το αλφάβητο Σ.

Παράδειγμα 3. Η $w = 0101$ είναι μια λέξη πάνω από το δυαδικό αλφάβητο Σ_2, με μήκος $|w| = 4$.

Ορισμός 3. (Γλώσσα)
Μια *γλώσσα* L πάνω από ένα αλφάβητο Σ είναι ένα οποιοδήποτε σύνολο, πεπερασμένο ή άπειρο, λέξεων πάνω από το Σ. Ως συνήθως στα μαθηματικά με $|L|$ συμβολίζουμε τον πληθάριθμο της L.

Παράδειγμα 4. Το κενό σύνολο $\emptyset$ αποτελεί μια γλώσσα πάνω από οποιοδήποτε αλφάβητο Σ που θα το ονομάζουμε *κενή γλώσσα* .

Παράδειγμα 5. Αν είναι Σ ένα αλφάβητο τότε, καταχρηστικά, συμβολίζουμε πάλι με Σ τη γλώσσα πάνω από το Σ που αποτελείται από όλες τις δυνατές λέξεις με ακριβώς ένα γράμμα, το ίδιο δηλ. το αλφάβητο με μία έννοια.

Παράδειγμα 6. Το σύνολο Σ^* όλων των λέξεων πάνω από ένα αλφάβητο Σ είναι η *πλήρης* γλώσσα πάνω από το Σ. Για οποιοδήποτε Σ αυτή είναι μια άπειρη γλώσσα που περιέχει ως υπογλώσσες (υποσύνολα) όλες τις γλώσσες πάνω από το Σ.

Παράδειγμα 7. Το σύνολο E όλων των επωνύμων που εμφανίζονται τον τηλεφωνικό κατάλογο Κρήτης του 2003 αποτελεί μια γλώσσα πάνω από το αλφάβητο Σ_{GR} των γραμμάτων και σημείων στίξης της ελληνικής γλώσσας (υποθέτουμε εδώ ότι δεν χρησιμοποιούνται πουθενά γράμματα του λατινικού αλφαβήτου στον κατάλογο). Η E είναι μια πεπερασμένη αλλά μεγάλη γλώσσα.

Παράδειγμα 8. Το σύνολο

$$Q = \left\{ w \in \Sigma_2^* : \exists k \in \mathbf{N} : \ |w| = k^2 \right\}$$

είναι η γλώσσα όλων των δυαδικών λέξεων με μήκος που είναι τέλειο τετράγωνο. Προφανώς πρόκειται για μια άπειρη γλώσσα.

Παράδειγμα 9. Το σύνολο L_k όλων των λέξεων του Σ_2 με μήκος ακριβώς k, όπου k είναι ένας φυσικός αριθμός, είναι μια πεπερασμένη υπογλώσσα του Σ_2^*.

Άσκηση 1. Βρείτε το $|L_k|$. Αν M_k είναι η υπογλώσσα του Σ_2^* που αποτελείται από όλες τις λέξεις με μήκος το πολύ k, βρείτε το $|M_k|$.

Άσκηση 2. Πόσες λέξεις μήκους n υπάρχουν πάνω από ένα αλφάβητο με k γράμματα;

Παράδειγμα 10. Ας είναι Σ_A το αλφάβητο που αποτελείται από τα γράμματα του λατινικού αλφαβήτου, μικρά και κεφαλαία, τα δεκαδικά ψηφία, σημεία στίξης, παρενθέσεις, αγκύλες, τον *λευκό* (space) χαρακτήρα (να μη συγχέεται με την κενή λέξη ϵ), το χαρακτήρα αλλαγής γραμμής (carriage return), και γενικά όλα τα σύμβολα που εμφανίζονται σε ένα σύγχρονο πληκτρολόγιο υπολογιστή. Αυτό το αλφάβητο ονομάζεται και αλφάβητο ASCII[1].

Η γλώσσα L_C είναι εκείνο το σύνολο από λέξεις του Σ_A^* που αποτελούν συντακτικά σωστά προγράμματα στη γλώσσα προγραμματισμού C . Αν δεν είστε γνώστες της γλώσσας αυτής αλλά κάποιας άλλης, μπορείτε να ορίσετε την αντίστοιχη γλώσσα. Ένα πρόγραμμα λοιπόν δεν είναι τίποτε άλλο από μια λέξη σε ένα κατάλληλο αλφάβητο. Αν αυτό ξενίζει να θυμίσουμε ότι και οι χαρακτήρες αλλαγής γραμμής βρίσκονται μέσα στο αλφάβητο, και άρα μια λέξη του L_A μπορεί να περιέχει τα γράμματα ενός ολόκληρου αρχείου κειμένου.

Το L_C είναι μια άπειρη γλώσσα αφού δεν υπάρχει άνω όριο στο μέγεθος ενός συντακτικά σωστού προγράμματος, και άρα υπάρχουν άπειρα τέτοια.

Ορισμός 4. (Συγκόλληση λέξεων, πρόθεμα και επίθεμα λέξης)
Αν $\alpha = \alpha_1 \ldots \alpha_m$ και $\beta = \beta_1 \ldots \beta_n$ είναι δύο λέξεις πάνω από το αλφάβητο Σ με μήκη m και n, τότε ορίζουμε τη *συγκόλληση* (concatenation) $\alpha\beta$ να είναι η λέξη

$$\alpha\beta = \alpha_1 \ldots \alpha_m \beta_1 \ldots \beta_n,$$

που έχει μήκος το άθροισμα των μηκών.

Μια λέξη π λέγεται *πρόθεμα* (prefix) μιας λέξης α αν υπάρχει λέξη β τ.ώ. $\alpha = \pi\beta$. Ομοίως η π λέγεται *επίθεμα* (suffix) της α αν υπάρχει β τ.ώ. $\alpha = \beta\pi$.

Τέλος, μια λέξη π λέγεται υπολέξη της w αν υπάρχουν λέξεις α, και β, ενδεχομένως κενές, τέτοιες ώστε $w = \alpha\pi\beta$.

Παρατήρηση 1. Προφανώς ισχύει πάντα $\alpha\epsilon = \epsilon\alpha = \alpha$, για κάθε λέξη α.

Είναι φανερό ότι μια λέξη με μήκος n έχει ακριβώς $n+1$ προθέματα και άλλα τόσα επιθέματα.

Επίσης η συγκόλληση λέξεων είναι μια πράξη μη αντιμεταθετική, αφού αν $\alpha = 01$ και $\beta = 00$ τότε $\alpha\beta = 0100 \neq 0001 = \beta\alpha$.

Παράδειγμα 11. Αν $\Sigma = \{a, b, c\}$ τότε η συγκόλληση των λέξεων abc και bb είναι η λέξη $abcbb$.

Παράδειγμα 12. Τα προθέματα της λέξης $abcd$ (δεν έχει σημασία σε ποιο αλφάβητο δουλεύουμε) είναι οι λέξεις $\epsilon, a, ab, abc, abcd$. Τα επιθέματα είναι οι λέξεις $abcd, bcd, cd, d, \epsilon$.

[1] Για την ακρίβεια ο **κώδικας** ASCII είναι ένα standard αντιστοίχισης όλων των συμβόλων που αναφέραμε παραπάνω στους αριθμούς 0-127. Το standard αυτό ακολουθείται σήμερα σε όλους τους υπολογιστές

Παρατήρηση 2. Αν τ είναι γράμμα ενός αλφαβήτου Σ τότε συμβολίζουμε επίσης με τ τη γλώσσα πάνω από το Σ που περιέχει μόνο μια λέξη, τη λέξη τ, που έχει μόνο ένα γράμμα.

Ορισμός 5. (Συγκόλληση γλωσσών)
Αν L_1, L_2 είναι δύο γλώσσες πάνω από το Σ ορίζουμε τη συγκόλληση $L_1 L_2$ αυτών να είναι η γλώσσα

$$L_1 L_2 = \{xy : \ x \in L_1, y \in L_2\}.$$

Αποτελείται δηλ. η $L_1 L_2$ από όλες τις λέξεις που προκύπτουν ως συγκόλληση μιας λέξης από την L_1 με μια λέξη της L_2.

Ορίζουμε επίσης $L^0 = \{\epsilon\}$ και, για $n \geq 1$, $L^n = LL \cdots L$ (συγκόλληση της L με τον εαυτό της n φορές).

Τέλος ορίζουμε

$$L^* = \bigcup_{k=0}^{\infty} L^k$$

και

$$L^+ = \bigcup_{k=1}^{\infty} L^k.$$

Παράδειγμα 13. Έστω $L = \{00, 11, \epsilon\}$. Τότε $L^* = L^+$ είναι η γλώσσα που απαρτίζεται από όλες εκείνες τις λέξεις από 0 ή 1 με άρτιο μήκος όπου το πρώτο γράμμα είναι ίδιο με το δεύτερο, το τρίτο με το τέταρτο κλπ. Η κενή λέξη ανήκει στην L^*.

Άσκηση 3. Περιγράψτε τη γλώσσα 1^* πάνω από το δυαδικό αλφάβητο Σ_2.

Άσκηση 4. Δείξτε ότι για κάθε γλώσσα L και φυσικούς αριθμούς m, n ισχύει $L^{m+n} = L^m L^n$.

Άσκηση 5. Αν L είναι μια οποιαδήποτε γλώσσα που δεν περιέχει την κενή λέξη ϵ, ποια είναι ακριβώς η διαφορά των γλωσσών L^* και L^+; Αλλάζει κάτι στην απάντηση αν $\epsilon \in L$;

Άσκηση 6. Βεβαιωθείτε ότι καταλαβαίνετε τι περιγράφει η γλώσσα

$$\{+, -, \epsilon\}1\{0, 1\}^*.$$

(Πρόκειται για συγκόλληση τριών γλωσσών.) Όταν, όπως εδώ, δεν περιγράφουμε το αλφάβητο, αυτό συνάγεται από όλα τα σύμβολα που έχουν χρησιμοποιηθεί, στην προκειμένη περίπτωση δηλαδή $\Sigma = \{0, 1, +, -\}$.

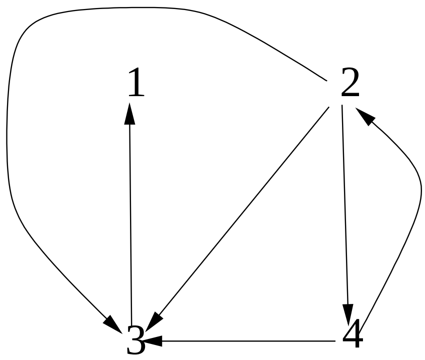
1.2 Γραφήματα

Ορισμός 6. (Κατευθυνόμενο γράφημα)
Ένα *κατευθυνόμενο γράφημα* G είναι ένα ζεύγος από δύο σύνολα V και E και μια συνάρτηση (πολλαπλότητα ακμής) $m : E \to \mathbf{N}$. Το σύνολο V λέγεται σύνολο *κορυφών* ή *κόμβων* και το σύνολο *ακμών* E είναι ένα υποσύνολο του

$$V \times V = \{(u, v) : \ u, v \in V\}.$$

Αν το ζεύγος (u, v) ανήκει στο E λέμε ότι υπάρχει τουλάχιστον μια ακμή από το u στο v. Ένα κατευθυνόμενο γράφημα το ζωγραφίζουμε συνήθως σα μια συλλογή από κορυφές και ακμές που τις ενώνουν, όπως στο Σχήμα 1 παρακάτω. Λέγεται κατευθυνόμενο επειδή υπάρχει η έννοια της κατεύθυνσης πάνω στις ακμές. Οι δε ακμές μπορούν να ενώνουν και μια κορυφή με τον εαυτό της, όπως επίσης μπορούν να υπάρχουν και πολλαπλές ακμές που ενώνουν το ίδιο ζεύγος κορυφών. Το πόσες ακμές ενώνουν την κορυφή u με την κορυφή v (από το u προς το v) δίδεται από τον αριθμό $m(u, v)$, που είναι ένας θετικός ακέραιος.

Παράδειγμα 14. Στο Σχήμα 1 παριστάνεται το γράφημα $G = (V, E, m)$ με $V = \{1, 2, 3, 4\}$, $E = \{(2, 3), (2, 4), (3, 1), (4, 2),$ και πολλαπλότητα m που είναι 1 για κάθε ακμή εκτός από τη (2,3) για την οποία έχουμε $m(2, 3) = 2$.



Σχήμα 1. Ενα κατευθυνόμενο γράφημα με 4 κορυφές και 5 ακμές. Μία από τις ακμές έχει πολλαπλότητα 2.

Πρέπει να τονίσουμε εδώ ότι σε ένα κατευθυνόμενο γράφημα (συνήθως θα λέμε απλώς γράφημα αντί για κατευθυνόμενο γράφημα όταν είναι ξακάθαρο τι εννούμε) δεν έχει καμία σημασία ο τρόπος που το σχεδιάζουμε πάνω στο χαρτί, αλλά αυτό που μας ενδιαφέρει είναι μόνο ποια κορυφή συνδέεται με ποια, και με τι πολλαπλότητα.

Ορισμός 7. (Διαδοχικές κορυφές)
Σε ένα κατευθυνόμενο γράφημα G η κορυφή v λέγεται *διαδοχική* της κορυφής u αν υπάρχει ακμή (u, v).

Ορισμός 8. (Μονοπάτια σε γραφήματα)
Σε ένα κατευθυνόμενο γράφημα *μονοπάτι* αποτελεί μια οποιαδήποτε πεπερασμένη ακολουθία κορυφών

$$\pi = u_0 u_1 \ldots u_n,$$

όπου η ακμή u_{i+1} είναι διαδοχική της u_i για κάθε $i = 0, \ldots, n-1$.

Ο αριθμός n λέγεται *μήκος* του μονοπατιού.

Αν οι κορυφές u_0 και u_n συμπίπτουν τότε το μονοπάτι λέγεται και *κύκλος*.

Παράδειγμα 15. Στο Σχήμα 1 το 24231 είναι ένα μονοπάτι μήκους 4. Το 424 είναι κύκλος.

Άσκηση 7. Έστω κατευθυνόμενο γράφημα G με n κορυφές και μονοπάτι στο G μέ μήκος $k \geq n$. Δείξτε ότι το μονοπάτι περιέχει ένα κύκλο.

Άσκηση 8. Αν σε ένα κατευθυνόμενο γράφημα G υπάρχει ένα μονοπάτι με μήκος μεγαλύτερο ή ίσο από το πλήθος των κορυφών τότε υπάρχουν άπειρα το πλήθος διαφορετικά μονοπάτια στο G.

Όταν δεν μας ενδιαφέρει η κατεύθυνση πάνω στις ακμές ενός γραφήματος και δεν επιτρέπουμε πολλαπλές ακμές από μια κορυφή σε μια άλλη τότε έχουμε μη κατευθυνόμενα, απλά γραφήματα. Στα γραφήματα αυτα γενικεύονται κατα προφανή τρόπο οι έννοιες των διαδοχικών κορυφών καθώς και των μονοπατιών.

Ορισμός 9. (Μη κατευθυνόμενο, απλό γράφημα)
Ένα *μη κατευθυνόμενο, απλό γράφημα* G είναι ένα ζεύγος από δύο σύνολα V και E. Το σύνολο V λέγεται σύνολο *κορυφών* ή *κόμβων* και το σύνολο *ακμών* E είναι ένα υποσύνολο του

$$\binom{V}{2} = \{\{u, v\} : \ u, v \in V, u \neq v\}.$$

Το E δηλ. απαρτίζεται από διμελή υποσύνολα του συνόλου των κορυφών. Αν το σύνολο $\{u, v\}$ ανήκει στο E τότε λέμε ότι η ακμή uv υπάρχει στο γράφημα G.

1.3 Σχέσεις πάνω σε σύνολα, σχέσεις ισοδυναμίας, μεταβατική κλειστότητα

Ορισμός 10. (Σχέση πάνω σε ένα σύνολο)
Σχέση R πάνω σε ένα σύνολο A είναι ένα οποιοδήποτε υποσύνολο $R \subseteq A \times A$.

Γράφουμε συνήθως $x\ R\ y$ αντί για $(x,y) \in R$.

Παράδειγμα 16. Αν $A = \mathbf{R}$, το σύνολο των πραγματικών αριθμών, ορίζουμε τη σχέση $R = \{(x,y) \in \mathbf{R}^2 :\ x \le y\}$. Αντί για $x\ R\ y$ γράφουμε κατά παράδοση $x \le y$.

Παράδειγμα 17. Αν $A = \mathbf{Z}$, το σύνολο των ακεραίων, και $m \in \mathbf{Z}$ είναι ένας δοσμένος ακέραιος, $m \neq 0$, ορίζουμε $R = \{(x,y) \in \mathbf{Z}^2 :\ m|y-x\}$. Αντί για $x\ R\ y$ γράφουμε $x \sim_m y$ και διαβάζουμε "το x είναι ισουπόλοιπο mod m με το y". Με λίγη σκέψη βλέπουμε ότι αυτό συμβαίνει αν και μόνο αν τα x και y αφήνουν το ίδιο υπόλοιπο διαιρούμενα με m.

Παράδειγμα 18. Ένα τετριμμένο παράδειγμα είναι η *πλήρης σχέση* $A \times A$, στην οποία δύο οποιαδήποτε στοιχεία του A σχετίζονται μεταξύ τους.

Παράδειγμα 19. Αν $A = \mathbf{Z}$, ορίζουμε $x\ R\ y$ αν και μόνο αν $2|x-y$ ή $3|x-y$.

Ορισμός 11. (Σχέσεις ανακλαστικές, συμμετρικές και μεταβατικές)
Μια σχέση R λέγεται *ανακλαστική* αν $x\ R\ x$ για κάθε $x \in A$, *συμμετρική* αν $x\ R\ y \Rightarrow y\ R\ x$ και, τέλος, *μεταβατική* αν $(x\ R\ y\ \&\ y\ R\ z) \Rightarrow x\ R\ z$.

Αν μια σχέση έχει και τις τρεις αυτές ιδιότητες τότε λέγεται σχέση *ισοδυναμίας*.

Παράδειγμα 20. Στο παράδειγμα 16 η σχέση είναι ανακλαστική και μεταβατική αλλά όχι συμμετρική. Στο παράδειγμα 17 η σχέση έχει και τις τρεις ιδιότητες και είναι συνεπώς μια σχέση ισοδυναμίας. Τέλος στο παράδειγμα 19 η σχέση είναι ανακλαστική και συμμετρική αλλά όχι και μεταβατική.

Άσκηση 9. Έστω G κατευθυνόμενο γράφημα με σύνολο κορυφών V. Ορίζουμε τη σχέση R στο V: $u\ R\ v$ αν υπάρχει μονοπάτι πάνω στο G που ξεκινά από το u και καταλήγει στο v. Είναι η σχέση R ανακλαστική, συμμετρική, μεταβατική;

Άσκηση 10. Γράφουμε $x\ R\ y$ για δυο ακεραίους x και y αν $x \mid y$ (ο x διαιρεί τον y). Δείξτε ότι η σχέση R είναι ανακλαστική και μεταβατική.

Άσκηση 11. Έστω R_1 και R_2 δύο σχέσεις ορισμένες πανω στο ίδιο σύνολο. Αν είναι ανακλαστικές (αντ. συμμετρικές, μεταβατικές) δείξτε ότι η σχεση $R_1 \cap R_2$ είναι επίσης ανακλαστική (αντ. συμμετρική, μεταβατική). Δείξτε το ίδιο και για την τομή

$$R = \bigcap_{j \in J} R_j$$

μιας οικογένειας σχέσεων, όπου το σύνολο δεικτών J δεν είναι κατ΄ ανάγκη πεπερασμένο, αλλά μπορεί να είναι οποιοδήποτε σύνολο.

Άσκηση 12. Ορίζουμε τη σχέση R στο σύνολο Σ_2^* (το σύνολο όλων των λέξεων που μπορεί κανείς να φτιάξει με τα γράμματα 0 και 1) ώστε να έχουμε xRy, για δύο λέξεις x και y, αν και μόνο αν η x είναι πρόθεμα της y. Είναι η σχέση R ανακλαστική, συμμετρική, μεταβατική;

Θεώρημα 1. (Ύπαρξη μεταβατικής κλειστότητας)
Έστω R μια σχέση πάνω σε ένα σύνολο A. Τότε υπάρχει μοναδική σχέση R^+ ορισμένη πάνω στο σύνολο A, που περιέχει την R, είναι μεταβατική και περιέχεται σε κάθε άλλη μεταβατική σχεση που περιέχει την R. Υπό αυτή την έννοια είναι η ελάχιστη μεταβατική σχέση που περιέχει την R *και την ονομάζουμε* μεταβατική κλειστότητα *ή* μεταβατική θήκη *της R.*

Απόδειξη. Ορίζουμε την κλάση $\mathcal{K}$ που απαρτίζεται από όλες τι σχέσεις X που περιέχουν την R και είναι ταυτόχρονα μεταβατικές. Η κλάση $\mathcal{K}$ είναι σίγουρα μη κενή μια και περιέχει την πλήρη σχέση $A \times A$. Ορίζουμε τώρα τη σχέση R^+ ως

$$R^+ = \bigcap_{X \in \mathcal{K}} X.$$

Είναι προφανές ότι $R \subseteq R^+$, και ότι καθε μεταβατική σχέση που περιέχει την R, δηλ. κάθε σχέση που ανήκει στην κλάση $\mathcal{K}$, περιέχει την R^+. Το ότι η R^+ είναι η ίδια μεταβατική προκύπτει από την Άσκηση 11.
□

Παρατήρηση 3. Υπάρχει και ένας άλλος τρόπος να δούμε τη σχέση R^+, εξ ίσου σημαντικός με τον τυπικό ορισμό. Το να μην είναι μια σχέση R μεταβατική σημαίνει ότι υπάρχουν ζεύγη $(a,b),(b,c) \in R$ αλλά $(a,c) \notin R$. Για να μετατρέψουμε λοιπόν τη σχέση R στη σχέση R^+, την 'ελάχιστη μεταβατική σχέση που περιέχει την R', προσθέτουμε στη σχέση R τα ζευγάρια αυτά που λείπουν, μέχρι να μη χρειάζεται να προσθέσουμε άλλα. Φυσικά, αυτή η διαδικασία που μόλις περιγράψαμε, για να έχει νόημα, πρέπει κάποια στιγμή να τελειώσει. Και αυτό δεν είναι πρόβλημα για σχέσεις R που είναι πεπερασμένες (τέτοιες είναι, για παράδειγμα, όλες οι σχέσεις ορισμένες σε πεπερασμένο σύνολο A) αλλά όταν θέλουμε να ορίσουμε το R^+ για άπειρες σχέσεις R πρέπει να χρησιμοποιήσουμε τον ορισμό που δώσαμε πιο πάνω με την, εν δυνάμει, άπειρη τομή.

Άσκηση 13. Αν μια σχέση R είναι ανακλαστική και συμμετρική τότε η R^+ είναι σχέση ισοδυναμίας.

Παράδειγμα 21. Έστω A το σύνολο όλων των ανθρώπων που έχουν ζήσει ποτέ. Έστω R η σχέση της γέννησης. Έχουμε δηλ. $x\ R\ y$ για δύο ανθρώπους x και y αν ο x είναι γονέας του y. Είναι φανερό ότι η σχέση R δεν είναι μεταβατική (αφού, π.χ. οι παππούδες δεν έχουν γεννήσει τα εγγόνια). Η μεταβατική κλειστότητα R^+ της R είναι η σχέση των απογόνων. Έχουμε δηλ. $x\ R^+\ y$ αν και μόνο αν ο y είναι απόγονος του x (ο y έχει γεννηθεί από κάποιον που έχει γεννηθεί από κάποιον, κλπ, που έχει γεννηθεί από τον x).

Παράδειγμα 22. Δείχνουμε τώρα ότι η μεταβατική κλειστότητα του Παραδείγματος 19, παραπάνω, είναι η *πλήρης* σχέση στους ακεραίους, δηλ. κάθε δύο ακέραιοι σχετίζονται μεταξύ τους. Πράγματι, αν πούμε R τη σχέση 19, τότε $x\ R^+\ x$ για κάθε $x \in \mathbf{Z}$ αφού το 0 διαιρείται από το 2 (και το 3). Επίσης αποδεικνύουμε με επαγωγή, ως προς το θετικό αριθμό k, ότι για κάθε $x \in \mathbf{Z}$ ισχύει $x\ R^+\ x+k$ και $x+k\ R^+\ x$. Δείχνουμε μόνο το πρώτο μια και το δεύτερο είναι εντελώς αντίστοιχο. Για $k=1$ πρέπει να δείξουμε ότι $x\ R^+\ x+1$. Αλλά $x\ R\ x+3$ (διαφορά ίση με 3) και $x+3\ R\ x+1$ (διαφορά ίση με 2) και επειδή $R \subseteq R^+$ έχουμε επίσης $x\ R^+\ x+3,\ x+3\ R^+\ x+1$ και από τη μεταβατικότητα της R^+ έχουμε $x\ R^+\ x+1$.

Υποθέτουμε τώρα $x\ R^+\ x+k$ για κάποιο k και για κάθε x και πάμε να δείξουμε $x\ R^+\ x+k+1$. Αλλά έχουμε $x\ R^+\ x+k$ και από το βήμα 1 έχουμε $x+k-1\ R^+\ x+k$. Από μεταβατικότητα έχουμε $x\ R^+\ x+k+1$, που τελειώνει την απόδειξη.

Άσκηση 14. Για δυο ακεραίους x και y γράφουμε $x\ R\ y$ αν το $|x-y|$ ισούται με 15 ή 13. Ποια η μεταβατική κλειστότητα της σχέσης R;

Άσκηση 15. Έστω $\Sigma = \{a,b,c\}$ και R η σχέση πάνω στο Σ^* που ορίζεται ως $x\ R\ y$ αν και μόνο αν $y = xa$ ή $y = xb$. Περιγράψτε τη μεταβατική κλειστότητα της R.

Ορισμός 12. (Ισοδύναμα στοιχεία, κλάσεις ισοδυναμίας)
Αν R είναι μια σχέση ισοδυναμίας πάνω σε ένα σύνολο A, $x, y \in A$, και $x\ R\ y$ τότε τα x και y ονομάζονται R-ισοδύναμα στοιχεία ή απλώς ισοδύναμα.

Ένα μη κενό υποσύνολο $K \subseteq A$ ονομάζεται κλάση ισοδυναμίας της σχέσης R αν κάθε δύο στοιχεία του K είναι μεταξύ τους R-ισοδύναμα και, επί πλέον, αν $x \in K$ και $x\ R\ y$ τότε έπεται ότι και $y \in K$.

Άσκηση 16. Ποιες από τις παρακάτω σχέσεις στο σύνολο $\{1,2,3,4\}$ είναι σχέσεις ισοδυναμίας;

1. $\{(1,1),(2,2),(3,3),(4,4)\}$.
2. $\{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4)\}$.
3. $\{(1,1),(1,2),(1,4),(2,2),(2,4),(3,3),(4,1),(4,2),(4,4)\}$.

Θεώρημα 2. (Διαμέριση του συνόλου σε κλάσεις ισοδυναμίας)
(α) Αν R είναι μια σχέση ισοδυναμίας πάνω σε ένα σύνολο A και K_1, K_2 είναι δύο διαφορετικές κλάσεις ισοδυναμίας της R τότε $K_1 \cap K_2 = \emptyset$.
(β) Το σύνολο A γράφεται ως ξένη ένωση κλάσεων ισοδυναμίας. Έτσι δύο στοιχεία του A είναι μεταξύ τους ισοδύναμα αν και μόνο αν ανήκουν στην ίδια κλάση ισοδυναμίας.

Απόδειξη. (α) Έστω $x \in K_1 \cap K_2$. Αν $K_1 \neq K_2$ τότε υπάρχει κάποιο στοιχείο του ενός συνόλου που δεν ανήκει στο άλλο. Χωρίς βλάβη της γενικότητας ας υποθέσουμε πως $y \in K_1 \setminus K_2$. Όμως επειδή $x, y \in K_1$ και K_1 είναι κλάση έπεται ότι $x\ R\ y$. Αλλά αυτό συνεπάγεται ότι και $y \in K_2$ μια και η K_2 είναι κλάση, άτοπο γιατί υποθέσαμε αλλιώς. Άρα τα K_1, K_2 είναι μεταξύ τους ξένα.

(β) Έστω X το σύνολο όλων των κλάσεων ισοδυναμία της R. Ισχύει τότε

$$A = \bigcup_{K \in X} K,$$

μια και κάθε στοιχείο του A ανήκει σε κάποια κλάση ισοδυναμίας (για παράδειγμα στο σύνολο όλων των στοιχείων του A που είναι ισοδύναμα προς αυτό – αυτό είναι μια κλάση ισοδυναμίας). Τα σύνολα που συμμετέχουν στην παραπάνω ένωση είναι ανά δύο ξένα, σύμφωνα με το (α), άρα το A διαμερίζεται στις κλάσεις ισοδυναμίας του ως προς τη σχέση ισοδυναμίας R.
□

Ορισμός 13. (Δείκτης μια σχέσης ισοδυναμίας)
Αν R είναι μια σχέση ισοδυναμίας πάνω σε ένα σύνολο A ο πληθάριθμος του συνόλου των κλάσεων ισοδυναμίας ονομάζεται *δείκτης* της σχέσης R, και μπορεί φυσικά να είναι πεπερασμένος ή άπειρος.

Άσκηση 17. Σε ένα δωμάτιο βρίσκονται άνδρες και γυναίκες και φοράνε μπλούζες χρώματος κόκκινου, πράσινου ή μπλέ. Αν x και y είναι δύο άτομα γράφουμε $x\ R\ y$ για να δηλώσουμε ότι τα δύο άτομα είναι του ίδιου φύλου και φοράνε το ίδιο χρώμα. Δείξτε ότι η R είναι μια σχέση ισοδυναμίας και περιγράψτε τις κλάσεις ισοδυναμίας. Ποιες είναι οι κλάσεις ισοδυναμίες και πόσες το πολύ είναι;

Άσκηση 18. Σε ένα κράτος της Πολυνησίας που απαρτίζεται από 14 νησιά δυο σημεία του εδάφους θεωρούνται ότι σχετίζονται από τη σχέση R αν μπορεί κανείς να πάει από το ένα στο άλλο περπατώντας (δεν υπάρχουν ανωμαλίες στο έδαφος). Δείξτε ότι η R είναι σχέση ισοδυναμίας και βρείτε το δείκτη της.

Άσκηση 19. Να βρεθεί ο δείκτης της σχέσης του Παραδείγματος 17 πάνω στους ακεραίους και να περιγραφούν οι κλάσεις ισοδυναμίας.

Άσκηση 20. Έστω f μια συνάρτηση με πεδίο ορισμού το $\mathbf{R}$ και πεδίο τιμών το σύνολο $\{1, \ldots, 10\}$. Για $x, y \in \mathbf{R}$ ορίζουμε $x\ R\ y$ αν και μόνο αν $f(x) = f(y)$. Δείξτε ότι η R είναι σχέση ισοδυναμίας, προσδιορίστε τις κλάσεις, κα πείτε ποια η σχέση του δείκτη με το σύνολο τιμών $f(\mathbf{R})$.

Άσκηση 21. Έστω $G = (V, E)$ ένα μη κατευθυνόμενο, απλό γράφημα. Για $u, v \in V$ γράφουμε $u\ R\ v$ αν και μόνο αν $u = v$ ή $u \neq v$ και $\{u, v\} \in E$. Περιγράψτε τις κλάσεις ισοδυναμίας της σχέσης R^+, αφού εξηγήσετε γιατί αυτή είναι σχέση ισοδυναμίας.

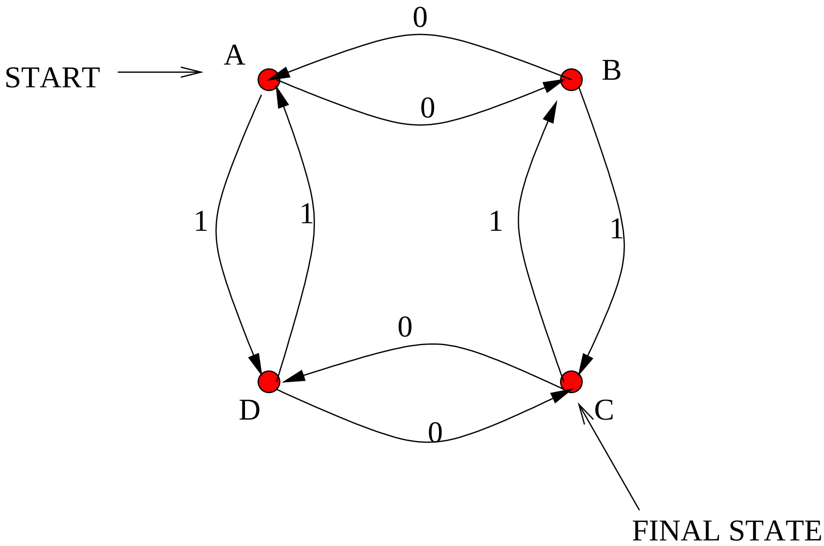
Κεφάλαιο 2

Αυτόματα

2.1 Ντετερμινιστικά Αυτόματα

Ένα Ντετερμινιστικό Αυτόματο (Deterministic Finite Automaton ή DFA) είναι ουσιαστικά ένα κατευθυνόμενο γράφημα, του οποίου οι κορυφές Q ονομάζονται *καταστάσεις* (states) και από κάθε κορυφή φεύγει ακριβώς μια ακμή για κάθε γράμμα του αλφαβήτου Σ. Υπάρχει μια διακεκριμένη κατάσταση q_0, η *αρχική κατάσταση* και ένα μη-κενό σύνολο F από *τελικές καταστάσεις*.

Δείτε π.χ. ένα τέτοιο αυτόματο στο Σχήμα 2.



Σχήμα 2. Ένα απλό ντετερμινιστικό αυτόματο

Ορισμός 14. (Ντετερμινιστικό Αυτόματο)
Ένα *ντετερμινιστικό αυτόματο* είναι μια πεντάδα

$$(Q, \Sigma, \delta, q_0, F)$$

όπου

- Q είναι ένα πεπερασμένο σύνολο *καταστάσεων*,

- Σ είναι ένα πεπερασμένο αλφάβητο,
- δ είναι η *συνάρτηση μετάβασης* (transition function) με πεδίο ορισμού το $Q \times \Sigma$ και πεδίο τιμών το Q,
- $q_0 \in Q$ είναι μια από τις καταστάσεις που ονομάζεται *αρχική*, και
- $F \subseteq Q$ είναι το σύνολο των *τελικών* καταστάσεων.

Παράδειγμα 23. Στο αυτόματο που φαίνεται στο Σχήμα 2 έχουμε $Q = \{A, B, C, D\}$, $\Sigma = \{0, 1\}$, $q_0 = A$, $F = \{C\}$.

Για να μιλήσουμε για τη συνάρτηση μετάβασης δ θα πρέπει πρώτα να αναφερθούμε στον τρόπο "λειτουργίας " του αυτομάτου. Ένα αυτόματο χρησιμεύει για να *αναγνωρίζει*, όπως λέμε, μια γλώσσα $L \subseteq \Sigma^*$. Η διαδικασία αναγνώρισης είναι η εξής.

Έστω λέξη $w \in \Sigma^*$, $w = w_1 \ldots w_n$, μήκους n ($w_i \in \Sigma$).

- Το αυτόματο αρχίζει τη λειτουργία του στην αρχική κατάσταση q_0.
- Διαβάζει έπειτα τα γράμματα της λέξης ένα προς ένα, από τα αριστερά προς τα δεξιά πάντα. Πρώτο διαβάζεται το γράμμα w_1 και τελευταίο το w_n.
- Μόλις διαβάσει το γράμμα $\alpha \in \Sigma$ και ευρισκόμενο στην κατάσταση q το αυτόματο *μεταβαίνει* στην κατάσταση r αν και μόνο αν η ακμή $q \to r$ έχει ετικέτα (label) ίση με α. Για κάθε κατάσταση του αυτομάτου και κάθε γράμμα του αλφαβήτου υπάρχει εξ ορισμού ακριβώς μια ακμή που ξεκινά από την κατάσταση αυτή και έχει ετικέτα αυτό το γράμμα. Ισχύει τότε για τη συνάρτηση μετάβασης $\delta(q, \alpha) = r$, χρησιμεύει δηλ. η συνάρτηση μετάβασης για να μας προσδιορίσει σε ποια κατάσταση πάει το αυτόματο αν βρίσκεται σε μια δεδομένη κατάσταση q και διαβάσει ένα συγκεκριμένο γράμμα α. Όλη η συνδεσμολογία του αυτομάτου δηλ. είναι κωδικοποιημένη στη συνάρτηση δ.
- Αφού διαβάσει το αυτόματο το τελευταίο γράμμα της λέξης και κάνει την τελευταία του μετάβαση δηλώνει ότι αποδέχεται τη λέξη αν και μόνο αν βρίσκεται σε τελική κατάσταση, σε μια κατάσταση δηλ. του συνόλου F. Αλλιώς η λέξη απορρίπτεται.

Ορισμός 15. (Γλώσσα ενός DFA)
Το σύνολο των λέξεων που αποδέχεται το αυτόματο M ονομάζεται *η γλώσσα που αναγνωρίζει το αυτόματο* και συμβολίζεται με $L(M)$.

Παράδειγμα 24. Ποια είναι η γλώσσα L που αναγνωρίζεται από το αυτόματο της εικόνας 2;

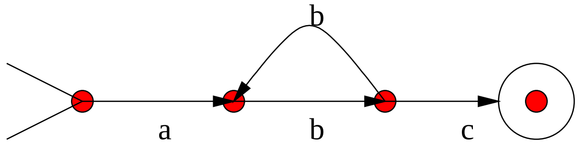
Δε θέλει και πολλή σκέψη για να πειστούμε ότι στην L ανήκουν ακριβώς εκείνες οι λέξεις του $\{0, 1\}^*$ που έχουν περιττό αριθμό από 0 και περιττό αριθμό από 1. Για να το δούμε αυτό παρατηρούμε ότι οποτεδήποτε το αυτόματο βρίσκεται σε μια από τις δύο αριστερές καταστάσεις το πλήθος των μηδενικών που έχει διαβάσει είναι άρτιο. Αυτό γίνεται γιατί αυτή η πρόταση ισχύει προφανέστατα τη χρονική στιγμή 0, αφού τότε δεν έχει διαβάσει κανένα μηδενικό και βρίσκεται στην αρχική κατάσταση A που είναι στην αριστερή μεριά. Οποτεδήποτε διαβάσει επίσης κάποιο μηδενικό το αυτόματο αλλάζει μεριά και διατηρείται έτσι η ιδιότητα αυτή. Ομοίως επιχειρηματολογώντας βλέπουμε ότι το αυτόματο βρίσκεται σε μια από τις δύο πάνω καταστάσεις (A και B) αν και μόνο αν έχει διαβάσει άρτιο αριθμό από 1. Έτσι, το να είναι το αυτόματο στην κατάσταση C (τελική) σημαίνει ότι έχει δει περιττό αριθμό από ένα και περιττό από μηδέν, αφού η C είναι κάτω (περιττοί άσσοι) και δεξιά (περιττά μηδενικά).

Αν π.χ. θελήσουμε να έχουμε ένα αυτόματο που θα αναγνωρίζει ακριβώς εκείνες τις λέξεις του $\{0, 1\}^*$ που έχουν *περιττά μηδενικά ή άρτιους άσσους* η μόνη αλλαγή που χρειάζεται να κάνουμε στην περιγραφή του αυτομάτου είναι να αλλάξουμε το σύνολο F των τελικών καταστάσεων και να το θέσουμε ίσο με το $\{A, B, C\}$.

Άσκηση 22. Σχεδιάστε ένα DFA που να αναγνωρίζει εκείνες τις λέξεις πάνω από το $\Sigma = \{0, 1\}$ που έχουν άρτιο πλήθος άσσων και πλήθος μηδενικών που είναι πολλαπλάσιο του 3.

Άσκηση 23. Σχεδιάστε ένα DFA που αναγνωρίζει εκείνες τις λέξεις πάνω από το $\Sigma = \{0, 1\}$ που αρχίζουν από 11011.

Παράδειγμα 25. Στο σχήμα 3 βλέπουμε ένα αυτόματο που αναγνωρζει τη γλώσσα $a\{bb\}^*bc$, δηλ. όλες εκείνες τις λέξεις στο αλφάβητο $\Sigma = \{a, b\}$ που αρχίζουν με a, ακολουθεί ένας οποιοσδήποτε αριθμός (ακόμη και μηδέν) από αντίγραφα της λέξης bb, και τελειώνουν με τη λέξη bc.



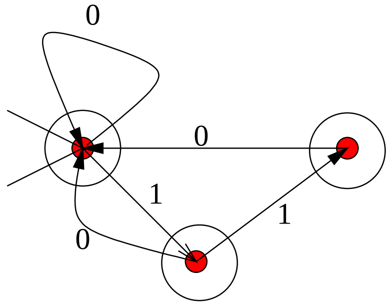
Σχήμα 3. DFA για τη γλώσσα $a\{bb\}^*bc$

Συμβολισμός: Στο Σχήμα 3, όπως και παρακάτω, συμβολίζουμε την αρχική κατάσταση με μια ανοιχτή γωνία (σα "χωνί", απ΄ όπου μπαίνουμε στο αυτόματο) και κάθε τελική κατάσταση μπαίνει μέσα σ'ενα κύκλο.

Συμβολισμός: Προσέξτε ότι στο αυτόματο του Σχήματος 3 δεν ισχύει η αρχική μας απαίτηση ότι από κάθε κορυφή πρέπει να φεύγει ακριβώς μια ακμή για κάθε γράμμα του αλφαβήτου (ώστε ό,τι και να διαβάσουμε όταν είμαστε σε κάποια κατάσταση να έχουμε που να πάμε). Για παράδειγμα, για τη δεύτερη κορυφή από αριστερά δεν υπάρχει ακμή με ετικέτα a που να ξεκινάει από αυτή. Η σύμβαση που ακολουθούμε είναι ότι αν διαβάσουμε ένα σύμβολο και βρισκόμαστε σε μια κατάσταση από την οποία δεν ξεκινάει ακμή με ετικέτα αυτό το σύμβολο, τότε η λέξη *δεν αναγνωρίζεται* από το αυτόματο.

Η σύμβαση αυτή ακολουθείται καθαρά για λόγους αισθητικής και κατανόησης του σχεδίου και δεν αλλάζει απολύτως τίποτα στην ουσία. Κάθε αυτόματο που ακολουθεί τη σύμβαση αυτή μπορεί εύκολα να μετατραπεί σε ένα αυτόματο που δε χρησιμοποιεί αυτή τη σύμβαση και έχει πράγματι μια ακμή για κάθε γράμμα από κάθε κορυφή. Απλά δηλώνουμε μια νέα *μη τελική* κατάσταση K, την κατάσταση καταστροφής, όπως επιλέγουμε να την ονομάσουμε, και από κάθε άλλη κατάσταση του αυτομάτου που δεν έχει ακμή από αυτή με ετικέτα έστω x ορίζουμε μια τέτοια ακμή προς την K. Όλες οι ακμές από το K επιστρέφουν πάλι στο K. Είναι εύκολο να δούμε ότι οι ίδιες ακριβώς λέξεις αναγνωρίζονται από το αρχικό και το νέο αυτόματο.

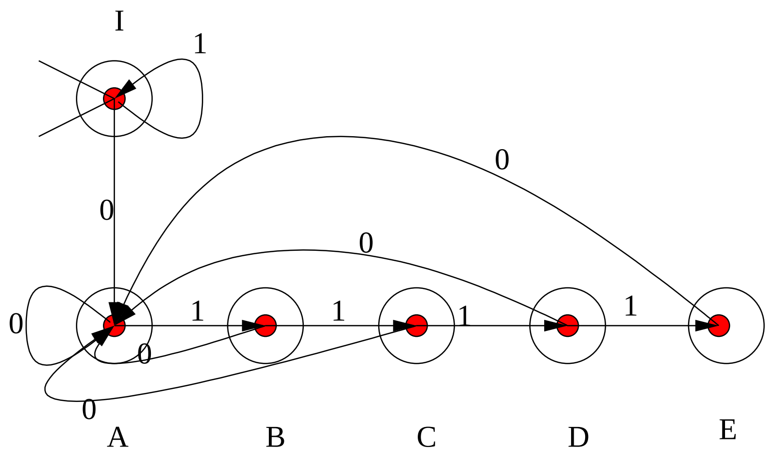
Παράδειγμα 26. Ακολουθεί στο Σχήμα 4 ένα αυτόματο που αναγνωρίζει τη γλώσσα L εκείνων των λέξεων του $\{0, 1\}^*$ που έχουν μέσα το πολύ δύο διαδοχικά 1. Ισοδύναμα, είναι εκείνες οι λέξεις στις οποίες *δεν εμφανίζεται* η μορφή 111.



Σχήμα 4. DFA για τις λέξεις χωρίς τρία διαδοχικά 1

Στο αυτόματο αυτό όλες οι καταστάσεις είναι τελικές, αλλά αυτό δεν αποτελεί αντίφαση, αφού ακολουθούμε τη σύμβαση της άνω παρατήρησης. Έτσι υπάρχουν πράγματι λέξεις που δεν αναγνωρίζονται από το αυτόματο, μια και κάποιες κορυφές δεν έχουν ακμές που ξεκινάνε από αυτές και για το 0 και το 1. Η μόνη τέτοια κορυφή είναι η δεξιά, και, όντως, η μόνη περίπτωση να απορριφθεί μια λέξη από το αυτόματο είναι να είμαστε στη δεξιά κατάσταση και να διαβάσουμε 1, πράγμα το οποίο συμβαίνει αν και μόνο αν υπάρχουν τρία διαδοχικά 1 στη λέξη.

Παράδειγμα 27. Το παράδειγμα αυτό είναι κάπως πιο ενδιαφέρον: L είναι η γλώσσα εκείνων των λέξεων του $\{0,1\}^*$ που είναι τέτοιες ώστε, *μετά το πρώτο 0* υπάρχει τουλάχιστον ένα 0 σε κάθε πεντάδα διαδοχικών γραμμάτων της λέξης. Για παράδειγμα, οι λέξεις 11111111111, 11111110111101111 είναι στην L ενώ η λέξη 111111011111 δεν είναι (η τελευταία πεντάδα δεν έχει 0). Το αυτόματο δίνεται στο Σχήμα 5. Πώς πρέπει να



Σχήμα 5. DFA για τις λέξεις με τουλάχιστον ένα 0 σε κάθε πεντάδα, μετά το πρώτο 0

σκεφθεί κανείς για να καταλάβει πώς δουλεύει το συγκεκριμένο αυτόματο;

Το κλειδί σε αυτή την περίπτωση, όπως και στις περισσότερες, είναι να αποδώσουμε κάποιο νόημα στην πρόταση 'βρισκόμαστε στην τάδε κατάσταση'. Για παράδειγμα, στην κατάσταση I βρισκόμαστε ακριβώς όταν δεν έχουμε διαβάσει ακόμη το πρώτο 0 της λέξης. Η πρόταση αυτή ισχύει κατ΄ αρχήν (πριν έχουμε διαβάσει τίποτε) και διατηρείται από οποιαδήποτε μετάβαση, αφού φεύγουμε από την κατάσταση I ακριβώς μόλις διαβάσουμε το πρώτο μηδενικό και δεν ξαναγυρνάμε σε αυτή. Είναι τώρα φανερό, έχοντας ξεκαθαρίσει τι σημαίνει να βρισκόμαστε στην κατάσταση I, ότι αυτή πρέπει να είναι τελική κατάσταση, μια και αν ποτέ δε διαβάσουμε κάποιο 0, ισχύει ο κανόνας που βάλαμε στην αρχή για το ποιες λέξεις ανήκουν στην L και άρα πρέπει να δεχτούμε τη λέξη.

Κάνουμε τώρα την εξής παρατήρηση. Ας πούμε πως ένας άνθρωπος, με λίγη μνήμη, έχει επιφορτισθεί με το καθήκον να αναγνωρίζει ακριβώς αυτές τις λέξεις. Αυτός μαθαίνει τα σύμβολα της λέξης ένα-ένα, πρώτα από τα αριστερά, όπως ακριβώς και το αυτόματο, και πρέπει κάποια στιγμή να πεί αν αυτή η λέξη ανήκει ή όχι στην L. Επειδή ακριβώς ο άνθρωπος αυτός έχει λίγη μνήμη, κι επειδή οι λέξεις τις οποίες κρίνει μπορούν να έχουν οσοδήποτε μεγάλο μήκος, αποφασίζει αντί ανά πάσα στιγμή να θυμάται ολόκληρη τη λέξη που έχει δει μέχρι τότε, να θυμάται απλώς *πόσο χρόνο πριν είδε το τελευταίο μηδενικό*. Είναι φανερό ότι αυτή η πληροφορία του αρκεί για να αποφασίσει. Θα απορρίψει δε τη λέξη αν και μόνο αν αυτός ο χρόνος ξεπεράσει το 5 σε κάποια χρονική στιγμή, γιατί ακριβώς τότε έχει δει μια πεντάδα γεμάτη με 1.

Μετά από αυτή την παρατήρηση το νόημα της κατάστασης A είναι ότι βρίσκεται εκεί αν έχει δεί 0 ακριβώς πριν από χρόνο 1. Το νόημα της κατάστασης B ομοίως είναι ότι έχει δεί το τελευταίο 0 ακριβώς πριν από χρόνο 2, κ.ο.κ., ενώ στην κατάσταση E βρισκόμαστε αν έχουμε δεί το τελευταίο 0 ακριβώς 5 χρονικές στιγμές πριν. Είναι φανερό τώρα γιατί η κατάσταση E δεν έχει μετάβαση που ξεκινά από αυτή με 1: αν είδαμε το τελευταίο 0 χρόνο 5 πίσω και μας έρθει 1 πάει να πει πως έχουμε μόλις συμπληρώσει μια πεντάδα γεμάτη με 1. Άρα δεν υπάρχει λόγος να συνεχίσουμε γιατί, ό,τι και να δούμε από δω και πέρα, η λέξη πρέπει να απορριφθεί.

Έυκολα μπορούμε τώρα να ελέγξουμε ότι οι διάφορες μεταβάσεις στο σχήμα 5 είναι φτιαγμένες ακριβώς ώστε ακολουθώντας αυτές να διατηρείται το νόημα της κάθε κατάστασης, και άρα το αυτόματο δουλεύει. Αν, για παράδειγμα, βρισκόμαστε σε μια οποιαδήποτε από τις καταστάσεις $A,\ldots,E$ και διαβάσουμε 0 πρέπει να μεταβούμε στην κατάσταση A αφού μετά τη μετάβασή μας θα έχουμε διαβάσει 0 ακριβώς χρόνο 1 πριν, και αυτό είναι το νόημα της κατάστασης A.

Άσκηση 24. Κατασκευάστε ένα DFA που αναγνωρίζει εκείνες τις λέξεις πάνω από το $\Sigma = \{0,1\}$ που έχουν μήκος τουλάχιστον 5 και το 5ο γράμμα τους από τα αριστερά είναι 1.

Άσκηση 25. Κατασκευάστε ένα DFA που αναγνωρίζει εκείνες τις λέξεις πάνω από το $\Sigma = \{0,1\}$ που έχουν μήκος τουλάχιστον 5 και το 5ο γράμμα τους από τα δεξιά είναι 1.

Άσκηση 26. L είναι η γλώσσα εκείνων των λέξεων του $\{0,1\}^*$ που είναι τέτοιες ώστε, *μετά το πρώτο 0* υπάρχουν τουλάχιστον δύο 0 σε κάθε πεντάδα διαδοχικών γραμμάτων της λέξης. Κατασκευάστε ένα DFA που αναγνωρίζει την L.

Άσκηση 27. Κατασκευάστε DFA για τις γλώσσες του $\{0,1\}^*$:

1. Λέξεις που τελειώνουν σε 00
2. Λέξεις που περιέχουν τρία διαδοχικά 0

Άσκηση 28. Κατασκευάστε DFA για τις γλώσσες

(α) 00^*1^*1, και

(β) $\{w \in \{a,b\}^* : \quad 6 \mid A(w) + 2B(w)\}$,

όπου $A(w)$, και $B(w)$ είναι το πλήθος των a και των b στη λέξη w και $x|y$ σημαίνει ότι το x διαιρεί το y.

Άσκηση 29. Αν M είναι DFA δείξτε πώς να κατασκευάσετε ένα DFA για τη γλώσσα $\Sigma^* \setminus L(M)$ (το συμπλήρωμα της γλώσσας $L(M)$).

2.2 Μη ντετερμινιστικά αυτόματα

Εισάγουμε τώρα μια παραλλαγή των ντετερμινιστικών αυτομάτων, τα *μη ντετερμινιστικά αυτόματα* (Non-deterministic Finite Automata ή NFA). Αυτά αποτελούν, φαινομενικά, μια ενίσχυση του μοντέλου των ντετερμινιστικών αυτομάτων, αφού, από τον ορισμό που θα δώσουμε, κάθε DFA θα είναι και NFA, και άρα οι γλώσσες που αναγνωρίζονται από τα NFA είναι ένα υπερσύνολο των γλωσσών που αναγνωρίζονται από DFA.

Θα δούμε όμως σύντομα ότι αυτό είναι μόνο φαινομενικό, και ότι τα δύο μοντέλα είναι απολύτως ισοδύναμα, όσον αφορά τουλάχιστον το ποιες γλώσσες αναγνωρίζουν. Με άλλα λόγια, θα δούμε μια μέθοδο που για κάθε NFA θα κατασκευάζει ένα ισοδύναμο DFA, ένα DFA δηλ. που θα αναγνωρίζει ακριβώς την ίδια γλώσσα με το δοθέν NFA. Παρ' όλα αυτα το να μελετούμε NFA προσφέρει μερικά πλεονεκτήματα σε σχέση με τα DFA σε ορισμένες περιπτώσεις, όπως θα δούμε στα παραδείγματα.

Ορισμός 16. (Μη ντετερμινιστικό Αυτόματο (NFA))
Ένα *μη ντετερμινιστικό αυτόματο* είναι μια πεντάδα

$$(Q, \Sigma, \delta, q_0, F)$$

όπου

- Q είναι ένα πεπερασμένο σύνολο *καταστάσεων*,
- Σ είναι ένα πεπερασμένο αλφάβητο,
- δ είναι η *συνάρτηση μετάβασης* (transition function) με πεδίο ορισμού το $Q \times \Sigma$ και πεδίο τιμών το δυναμοσύνολο 2^Q, το σύνολο δηλ. όλων των υποσυνόλων του Q.
- $q_0 \in Q$ είναι μια από τις καταστάσεις που ονομάζεται *αρχική*, και
- $F \subseteq Q$ είναι το σύνολο των *τελικών* καταστάσεων.

Ένα μη ντετερμινιστικό αυτόματο είναι σαν ένα DFA αλλά για κάθε κορυφή και για κάθε γράμμα του αλφαβήτου μπορεί κανεις να έχει οποιοδήποτε πεπερασμένο πλήθος από ακμές που ξεκινούν από την κορυφή και έχουν το γράμμα ως ετικέτα Μπορεί ακόμη και να υπάρχει καμία ακμή από κάποια κορυφή για κάποιο γράμμα. (Αν $\delta(q, \alpha) = \emptyset$ είναι το κενό σύνολο τότε δεν υπάρχει στο γράφημα καμία ακμή από την κορυφή q με ετικέτα α.)

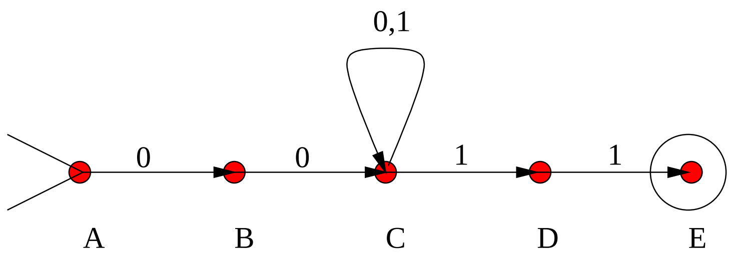
Η σημαντική διαφορά με πριν είναι ότι, όταν διαβάζουμε μια λέξη, δεν είναι πλέον μονοσήμαντα καθορισμένη η κίνησή μας πάνω στο γράφημα αφού ενδέχεται να έχουμε περισσότερες από μία επιλογές μετάβασης όντας σε μία κορυφή και διαβάζοντας ένα γράμμα. Το σε ποιες καταστάσεις μπορούμε να πάμε από μια κατάσταση $q \in Q$ αφού διαβάσουμε το γράμμα α μας το λέει η συνάρτηση μετάβασης. Το σύνολο αυτών των δυνατών καταστάσεων είναι το $\delta(q, \alpha) \subseteq Q$.

Παρατήρηση 4. Αν για μια κατάσταση q ενός NFA και για ένα γράμμα $\alpha \in \Sigma$ έχουμε $\delta(q, \alpha) = \emptyset$ αυτό ερμηνεύεται ως να μην είναι δυνατή η μετάβαση από την κατάσταση q προς οποιαδήποτε άλλη κατάσταση με το γράμμα α.

Ορισμός 17. (Αναγνώριση λέξης από NFA)
Θεωρούμε ότι μια λέξη $w = w_1 w_2 \ldots w_n$, $w_j \in \Sigma$, αναγνωρίζεται από ένα NFA αν υπάρχει τρόπος να κινηθούμε, διαβάζοντας το w, πάνω στο γράφημα ώστε να καταλήξουμε σε τελική κορυφή. Ξεκινούμε ευρισκόμενοι στην αρχική κατάσταση q_0 και διαβάζουμε τα γράμματα w_1 έως w_n με αυτή τη σειρά. Κάθε φορά που διαβάζουμε ένα γράμμα w_j, και ευρισκόμενοι στην κατάσταση q, επιλέγουμε ως επόμενη κατάσταση μια από τις καταστάσεις του συνόλου $\delta(q, w_j)$. (Αν το σύνολο αυτό είναι κενό η λέξη απορρίπτεται.) Είναι φανερό ότι για κάθε λέξη υπάρχουν ενδεχομένως περισσότεροι από ένας τρόποι να κινηθούμε πάνω στο αυτόματο διαβάζοντας τα γράμματα αυτής της λέξης, αφού σε κάθε βήμα ενδέχεται να έχουμε τη δυνατότητα να επιλέξουμε την επόμενή μας κατάσταση. Απορρίπτεται η λέξη w αν και μόνο αν κανείς από τους δυνατούς τρόπους κίνησης δεν καταλήγει σε τελική κορυφή.

Ορισμός 18. (Γλώσσα ενός NFA)
Το σύνολο των λέξεων που αποδέχεται το μη ντετερμινιστκό αυτόματο M ονομάζεται *η γλώσσα που αναγνωρίζει το αυτόματο* και συμβολίζεται με $L(M)$.

Παράδειγμα 28. Το ακόλουθο παράδειγμα μη ντετερμινιστικού αυτομάτου αναγνωρίζει τη γλώσσα $00\{0,1\}^*11$.



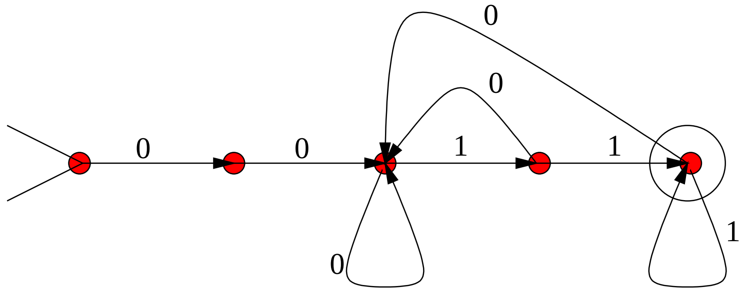
Σχήμα 6. NFA για τη γλώσσα $00\{0,1\}^*11$

Πρόκειται για NFA και όχι για DFA μια και στη μεσαία κορυφή υπάρχουν δύο ακμές με ετικέτα 1. Παρατηρείστε τη συντομογραφία που επιλέξαμε εδώ: ο βρόγχος (loop) από τη μεσαία κορυφή στον εαυτό της έχει δύο ετικέτες 0 και 1. Αυτό σημαίνει ότι επιτρέπεται να διανύσουμε την κορυφή αυτή όταν διαβάσουμε είτε 0 είτε 1. Θα μπορούσαμε να είχαμε το ίδιο αποτέλεσμα αν φτιάχναμε δύο βρόγχους από την κορυφή αυτή στον εαυτό της, ένα με ετικέτα 0 κι ένα με ετικέτα 1. Επιλέγουμε τη συντομογραφία για καθαρότητα στο σχήμα και τίποτε άλλο.

Γιατί το NFA του σχήματος 6 αναγνωρίζει τη γλώσσα $L = 00\{0,1\}^*11$; Παρατηρείστε ότι η L αποτελείται από εκείνες ακριβώς τις λέξεις που αρχίζουν με 00 και τελειώνουν με 11. Κάθε τέτοια λέξη περνάει από το NFA ως εξής: με τα δύο πρώτα 0 μεταβαίνει στη μεσαία κορυφή, στην οποία και παραμένει χρησιμοποιώντας το βρόγχο μέχρι να διαβάσει και το προπροτελευταίο γράμμα. Με τα δύο τελευταία 1 επιλέγει να κινηθεί προς τα δεξιά για να καταλήξει στη μοναδική τελική κορυφή. Θα μπορούσε να επιλέξει με τα δύο τελευταία 1 να κινηθεί πάνω στο βρόγχο της μεσαίας κορυφής, αλλά με αυτό τον τρόπο δεν καταλήγει σε τελική κορυφή. Αρκεί όμως για την αποδοχή μιας λέξης να υπάρχει έστω και ένας τρόπος να γίνει αποδεκτή.

Αντίστροφα τώρα, αν η λέξη w περνά από το NFA τότε υποχρεωτικά αρχίζει από δύο 0 (αλλιώς δε μπορεί να πάει δεξιότερα από τη δεύτερη κορυφή) και τελειώνει σε δύο 1 (αλλιώς δε μπορεί να καταλήξει στην τελική κορυφή).

Ας δώσουμε τώρα και ένα DFA για την ίδια γλώσσα:



Σχήμα 7. DFA για τη γλώσσα $00\{0,1\}^*11$

Είναι μια καλή άσκηση να να αποδείξετε ότι το άνω αυτόματο αναγνωρίζει τη γλώσσα $00\{0,1\}^*11$. Για να το κάνετε δώστε κάποιο 'νόημα' στις καταστάσεις, όπως κάναμε παραπάνω για να δείξουμε ότι το DFA του σχήματος 5 δουλεύει.

Παρατήρηση 5. Συγκρίνοντας το NFA και το DFA για τη γλώσσα $00\{0,1\}^*11$ φαίνεται καθαρά το πλεονέκτημα του να σχεδιάζουμε NFA αντί για DFA. Επιτρέποντας στον εαυτό μας ένα πιο διευρυμένο μοντέλο μπορούμε να σχεδιάζουμε ευκολότερα αυτόματα για συγκεκριμένες γλώσσες, και η απλούστερη κατασκευή συνήθως επιτρέπει και μια ευκολότερη ανάλυση του γιατί το συγκεκριμένο αυτόματο δουλεύει. Και, θα δούμε αργότερα, δε χάνεται τίποτα σχεδιάζοντας NFA αντί για DFA, γιατί αυτά τα δύο υπολογιστικά μοντέλα είναι ισοδύναμα. Όχι μόνο αυτό, αλλά ο τρόπος μετατροπής ενός NFA σε DFA μπορεί να είναι τελείως μηχανικός (αλγοριθμικός).

Παράδειγμα 29. Στο αυτόματο που φαίνεται στο σχήμα 6 παραπάνω έχουμε $Q = \{A,B,C,D,E\}$, $\Sigma = \{0,1\}$, $q_0 = A$, $F = \{E\}$, και

$$\begin{aligned}
\delta(A,0) &= \{B\}\\
\delta(A,1) &= \emptyset\\
\delta(B,0) &= \{C\}\\
\delta(B,1) &= \emptyset\\
\delta(C,0) &= \{C\}\\
\delta(C,1) &= \{C,D\}\\
\delta(D,0) &= \emptyset\\
\delta(D,1) &= \{E\}\\
\delta(E,0) &= \emptyset\\
\delta(E,1) &= \emptyset
\end{aligned}$$

Άσκηση 30. Κατασκευάστε ένα NFA που να αναγνωρίζει τη γλώσσα της άσκησης 25.

Άσκηση 31. Κατασκευάστε ένα NFA που να αναγνωρίζει εκείνες τις λέξεις πάνω από το $\Sigma = \{0,1\}$ που περιέχουν δύο μηδενικά που απέχουν μεταξύ τους στη λέξη κατά πολλαπλάσιο του 4. Δύο διαδοχικά μηδενικά θεωρούνται ότι απέχουν απόσταση μηδέν, άρα πολλαπλάσιο του 4.

2.3 Ισοδυναμία NFA και DFA.

Επαναλαμβάνουμε ότι για ένα NFA η συνάρτηση μετάβασης δ είναι ο τρόπος κωδικοποίησης των μεταβάσεων του. Αν δηλ. $q \in Q$ είναι μια κατάσταση και $\alpha \in \Sigma$ είναι ένα γράμμα του αλφαβήτου, το σύνολο $\delta(q, \alpha) \subseteq Q$ είναι το σύνολο όλων των καταστάσεων του NFA στις οποίες μπορεί αυτό να μεταβεί όντας στην κατάσταση q και διαβάζοντας το γράμμα α.

Ορισμός 19. (Η συνάρτηση δ πάνω σε σύνολα)
Αν $A \subseteq Q$ είναι ένα σύνολο καταστάσεων και $B \subseteq \Sigma$ είναι ένα σύνολο γραμμάτων τότε

$$\delta(A, B) = \bigcup_{q \in A, \alpha \in B} \delta(q, \alpha).$$

Δηλ. $\delta(A, B)$ είναι το σύνολο όλων των καταστάσεων στις οποίες μπορεί κανείς να φτάσει ξεκινώντας από κάποια κατάσταση του A και ακολουθώντας κάποιο γράμμα α του συνόλου B. Η επέκταση αυτή της συνάρτησης δ ώστε να παίρνει ως όρισμα σύνολα αντί για γράμματα είναι η συνηθισμένη επεκταση μιας συνάρτησης $f : C \to D$ που με $f(X)$, $X \subseteq C$, συμβολίζει το σύνολο εικόνων του συνόλου X μέσω της f.

Επεκτείνουμε τώρα το πεδίο ορισμού της συνάρτησης δ όσον αφορά το δεύτερο όρισμά της.

Ορισμός 20. (Η συνάρτηση μετάβασης δ πάνω σε λέξεις)
Αν $q \in Q$ είναι μια κατάσταση και $w \in \Sigma^*$ είναι μια λέξη (ενδεχομένως και κενή) τότε ορίζουμε το σύνολο καταστάσεων $\delta(q, w)$ ως εξής:

$$\delta(q, w) = \begin{cases} \{q\} & \epsilon\alpha\nu \quad w = \epsilon \\ \delta(\delta(q, v), \alpha) & \epsilon\alpha\nu \quad w = v\alpha, v \in \Sigma^*, \alpha \in \Sigma. \end{cases} \tag{2.1}$$

Παρατήρηση 6. Ο ορισμός (2.1) απαιτεί λίγη προσοχή όσον αφορά το δεύτερο σκέλος του μια και είναι *αυτοαναφορικός*, αναφερόμαστε δηλ. στη συνάρτηση δ για να ορίσουμε τη συνάρτηση δ. Η ουσία εδώ είναι ότι αυτή η αυτοαναφορά δε δημιουργεί κάποιο πρόβλημα, μια και για να ορίσουμε το $\delta(q, w)$ αναφερόμαστε σε τιμές του $\delta(q, x)$ για λέξεις x με μήκος $|x| < |w|$, και υπάρχει και ο ξεχωριστός ορισμός για τη λέξη μήκους 0 όπου σταματά η αυτοαναφορά. Για να υπολογιστεί έτσι το $\delta(q, w)$ υπολογίζεται πρώτα το δ για ολοένα και μικρότερες λέξεις, ώσπου τελικά φτάνει να χρησιμοποιείται το πρώτο μέλος του ορισμού (2.1) που δεν έχει αυτοαναφορά. Για παράδειγμα, αν θέλει κανείς να υπολογίσει την τιμή $\delta(q, abc)$, όπου $a, b, c \in \Sigma$, ακολουθώντας τον ορισμό (2.1) αυτό γίνεται ως εξής:

$$\begin{aligned} \delta(q, abc) &= \delta(\delta(q, ab), c) \\ &= \delta(\delta(\delta(q, a), b), c). \end{aligned}$$

Παρατηρούμε ότι στην τελευταία γραμμή η συνάρτηση δ είναι η γνωστή μας συνάρτηση που σαν δεύτερο όρισμα έχει σύμβολο του αλφαβήτου και όχι λέξη. Ορισμοί όπως ο (2.1) είναι πολύ κοινοί και ονομάζονται *αναδρομικοί*.

Με άλλα λόγια $\delta(q, w), w \in \Sigma^*$, είναι το σύνολο όλων εκείνων των καταστάσεων του αυτομάτου στις οποίες μπορεί κανείς να βρεθεί ξεκινώντας από την κατάσταση q και ακολουθώντας τη λέξη w. Και, αν $R \subseteq Q$ είναι ένα σύνολο καταστάσεων, με $\delta(R, w)$ συμβολίζεται το σύνολο όλων των καταστάσεων στις οποίες μπορεί κανείς να πάει ξεκινώντας από κάποια κατάσταση στο R και ακολουθώντας τη λέξη w.

Παρατήρηση 7. Μια λέξη w αναγνωρίζεται από ένα αυτόματο αν και μόνο αν

$$\delta(q_0, w) \cap F \neq \emptyset,$$

όπου F το σύνολο των τελικών καταστάσεων του NFA. Αναγνωρίζεται δηλ. μια λέξη αν και μόνο αν *είναι δυνατή η μετάβαση* από την αρχική σε *κάποια* τελική κατάσταση ακολουθώντας τη λέξη.

Ορισμός 21. (Ισοδυναμία αυτομάτων)
Δύο ντετερμινιστικά ή μη ντετερμινιστικά αυτόματα M_1 και M_2 λέγονται *ισοδύναμα* αν και μόνο αν $L(M_1) = L(M_2)$, αν αναγνωρίζουν δηλ. ακριβώς τις ίδιες λέξεις.

Θεώρημα 3. (Ισοδυναμία NFA και DFA)
Για κάθε NFA M υπάρχει ισοδύναμο DFA M'.

Απόδειξη. Περιγράφουμε ένα αλγόριθμο μετάβασης από ένα τυχόν NFA M σε ένα *ισοδύναμο* DFA M'.

Έστω λοιπόν ότι το NFA M είναι η πεντάδα $(Q, \Sigma, \delta, q_0, F)$. Το DFA $M' = (Q', \Sigma, \delta', q_0', F')$ θα έχει σύνολο καταστάσεων $Q' = 2^Q$ το δυναμοσύνολο (σύνολο όλων των υποσυνόλων) του Q, αρχική κατάσταση $q_0' = \{q_0\}$, ίδιο αλφάβητο Σ και τελικές καταστάσεις όλα εκείνα τα σύνολα καταστάσεων του M που περιέχουν κάποια τελική κατάσταση

$$F' = \{A \subseteq Q : A \cap F \neq \emptyset\}.$$

Τέλος η συνάρτηση μετάβασης $\delta' : Q' \times \Sigma \to Q'$ του M' ορίζεται ως

$$\delta'(q', \alpha) = \delta(q', \alpha).$$

Θυμηθείτε ότι το σύμβολο q' παριστάνει μια κατάσταση του M' άρα ένα σύνολο καταστάσεων του M, και άρα η δ συνάρτηση που εμφανίζεται δεξιά στον άνω ορισμό είναι η επεκτεταμένη δ συνάρτηση του αυτομάτου M όπως την ορίσαμε παραπάνω, μια και σαν πρώτο όρισμα έχει ολόκληρο σύνολο.

Με άλλα λόγια: στο DFA M' που κατασκευάζουμε από την κατάσταση q_1 (υποσύνολο του Q) με το σύμβολο $\alpha \in \Sigma$ μεταβαίνουμε στην κατάσταση q_2, που είναι το σύνολο όλων εκείνων των καταστάσεων του M στις οποίες μπορούμε να μεταβούμε από κάποια κατάσταση του συνόλου q_1 με το γράμμα α κινούμενοι πάνω στο M. Δείτε το Παράδειγμα 30 παρακάτω.

Άσκηση 32. Αποδείξτε με επαγωγή ως προς το μήκος $|w|$ της λέξης $w \in \Sigma^*$ ότι για κάθε κατάσταση q του M' ισχύει

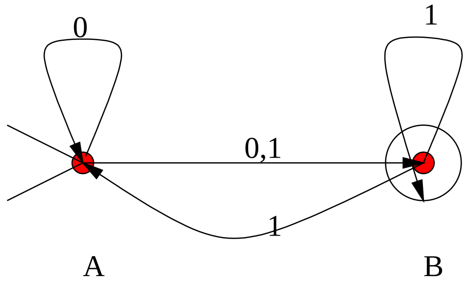
$$\delta'(q, w) = \delta(q, w).$$

Για να δείξουμε το Θεώρημα αρκεί να δείξουμε την ισοδυναμία

$$w \in L(M) \Longleftrightarrow w \in L(M') \tag{2.2}$$

για όλες τις λέξεις $w \in \Sigma^*$. Αλλά $w \in L(M)$ αν και μόνο αν $\delta(q_0, w) \in F$, ενώ $w \in L(M')$ αν και μόνο αν $\delta'(q_0', w) \in F'$. Χρησιμοποιώντας το Πρόβλημα 32 το τελευταίο συμβαίνει αν και μόνο αν $\delta(\{q_0\}, w) \in F'$ και, χρησιμοποιώντας τον ορισμό του F' βλέπυμε ότι αυτό ισχύει αν και μόνο αν $\delta(q_0, w) \cap F \neq \emptyset$. Όμως το σύνολο $\delta(q_0, w)$ είναι μονοσύνολο μια και το M είναι DFA, άρα η τελευταία συνθήκη είναι ισοδύναμη με $\delta(q_0, w) \in F$, και η απόδειξη είναι πλήρης.
□

Παράδειγμα 30. Ας δούμε τώρα πώς μετατρέπεται το ακόλουθο απλό NFA του Σχήματος 8 σε DFA. Το αποτέλεσμα δίνεται στο Σχήμα 9.

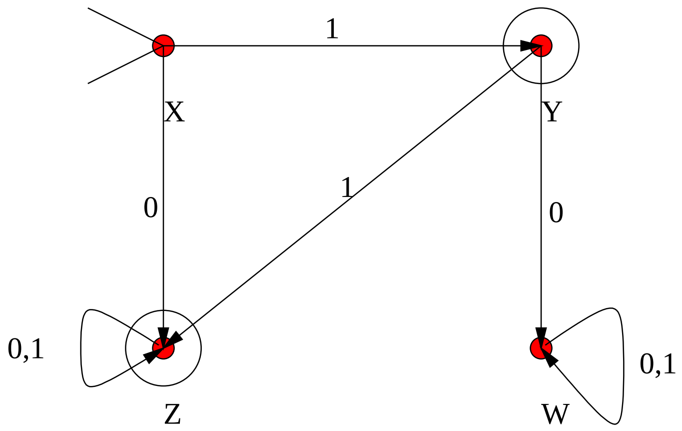


Σχήμα 8. M, ένα απλό NFA

Το σύνολο καταστάσεων του M' θα είναι το δυναμοσύνολο του συνόλου καταστάσεων του M δηλ. του $\{A, B\}$. Είναι δηλ. το σύνολο

$$Q' = \{X = \{A\}, Y = \{B\}, Z = \{A, B\}, W = \emptyset\}.$$

Σύνολο τελικών καταστασεων είναι το $F' = \{Y, Z\}$, όλα εκείνα τα σύνολα δηλ. που περιέχουν κάποια τελική κατάσταση του M. Αρχική κατάσταση είναι η X.



Σχήμα 9. Το NFA M του Σχήματος 8 μετατετραμμένο στο DFA M'

Άσκηση 33. Ελέγξτε την ορθότητα την ισοδυναμίας του NFA του σχήματος 8 στο DFA του σχήματος 9, πάνω στις λέξεις 001 και 10.

Άσκηση 34. Φτιάξτε ισοδύναμα DFA για τα NFA

1. $(Q = \{p, q, r, s\}, \Sigma = \{0, 1\}, \delta_1, q_0 = p, F = \{s\})$
2. $(Q = \{p, q, r, s\}, \Sigma = \{0, 1\}, \delta_2, q_0 = p, F = \{q, s\})$

όπου οι συναρτήσεις μετάβασης είναι οι

δ_1 :

	0	1
p	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
s	$\{s\}$	$\{s\}$

δ_2 :

	0	1
p	$\{q, s\}$	$\{q\}$
q	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s	$\emptyset$	$\{p\}$

Άσκηση 35. Αν ένα NFA έχει n το πλήθος καταστάσεις, πόσες το πολύ καταστάσεις χρειάζεται να έχει ένα ισοδύναμό του DFA;

2.4 NFA με ϵ-κινήσεις

Μια ακόμη παραλλαγή του μοντέλου του πεπερασμένου αυτομάτου είναι το μη ντετερμινιστικό αυτόματο με ϵ-κινήσεις. Αυτό είναι ένα NFA που έχει, ενδεχομένως, και κάποιες ακμές που αντί να έχουν ως ετικέτα ένα γράμμα του αλφαβήτου έχουν την κενή λέξη ϵ. Κινούμενοι πάνω στο αυτόματο αυτό για να αναγνωρίσουμε μια λέξη έχουμε το δικαίωμα να διανύσουμε μια ϵ-ακμή οποτεδήποτε υπάρχει μια τέτοια από την τρέχουσα κατάσταση *χωρίς να μας ενδιαφέρει ποιο είναι το επόμενο γράμμα της λέξης*. Η διάνυση μιας ϵ-ακμής *δε συνοδεύεται από μεταπήδηση στο επόμενο γράμμα της λέξης*. Ας τα λέμε αυτά ϵ-NFA.

Ορισμός 22. (Μη ντετερμινιστικό Αυτόματο με ϵ-κινήσεις (ϵ-NFA))
Ένα *μη ντετερμινιστικό αυτόματο* με ϵ-κινήσεις (ϵ-NFA) είναι μια πεντάδα

$$(Q, \Sigma, \delta, q_0, F)$$

όπου

- Q είναι ένα πεπερασμένο σύνολο *καταστάσεων*,
- Σ είναι ένα πεπερασμένο αλφάβητο,
- δ είναι η *συνάρτηση μετάβασης* (transition function) με πεδίο ορισμού το $Q \times (\Sigma \cup \{\epsilon\})$ και πεδίο τιμών το δυναμοσύνολο 2^Q, το σύνολο δηλ. όλων των υποσυνόλων του Q.
- $q_0 \in Q$ είναι μια από τις καταστάσεις που ονομάζεται *αρχική*, και
- $F \subseteq Q$ είναι το σύνολο των *τελικών* καταστάσεων.

Παρατήρηση 8. Η ουσιαστική διαφορά από τον ορισμό του NFA είναι ότι η συνάρτηση μετάβασης μπορεί να έχει και το ϵ ως δεύτερο όρισμα και όχι απαραίτητα ένα γράμμα του αλφαβήτου Σ.

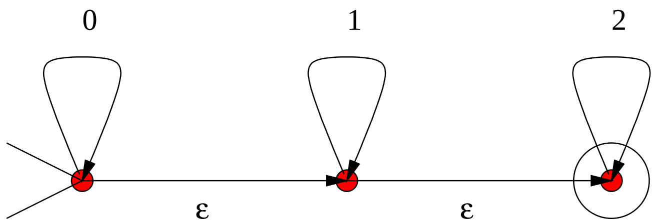
Παρατήρηση 9. Το νόημα μιας ϵ-ακμής από μια κορυφή q_1 σε μια κορυφή q_2 είναι το εξής: αν το NFA βρίσκεται στην κατάσταση q_1 τότε μπορεί να επιλέξει να μεταβεί στην κατάσταση q_2 *χωρίς να διαβάσει το επόμενο γράμμα της λέξης.*

Ορισμός 23. (Αναγνώριση λέξης από ϵ-NFA)
Θεωρούμε ότι μια λέξη $w = w_1 w_2 \ldots w_n$, $w_j \in \Sigma$, αναγνωρίζεται από ένα ϵ-NFA αν υπάρχει τρόπος να κινηθούμε, διαβάζοντας το w, πάνω στο γράφημα ώστε να καταλήξουμε σε τελική κορυφή. Ξεκινούμε ευρισκόμενοι στην αρχική κατάσταση q_0 και διαβάζουμε τα γράμματα w_1 έως w_n με αυτή τη σειρά. Σε κάθε βήμα έχουμε τη δυνατότητα να διαβάσουμε το επόμενο γράμμα w_j και να εκτελέσουμε μια μετάβαση σε νέα κατάσταση ή όχι.

Κάθε φορά που διαβάζουμε ένα γράμμα w_j, και ευρισκόμενοι στην κατάσταση q, επιλέγουμε ως επόμενη κατάσταση μια από τις καταστάσεις του συνόλου $\delta(q, w_j)$. (Αν το σύνολο αυτό είναι κενό η λέξη απορρίπτεται.) Αν επιλέξουμε να μη διαβάσουμε το επόμενο γράμμα της λέξης μπορούμε να μεταβούμε σε νέα κατάσταση ακολουθώντας μια ϵ-ακμή, μπορούμε δηλ. να μεταβούμε σε μια αποιαδήποτε από τις καταστάσεις του συνόλου $\delta(q, \epsilon)$, όπου q είναι η τρέχουσα κατάσταση. (Αν το σύνολο αυτό είναι κενό η λέξη απορρίπτεται.) Είναι φανερό ότι για κάθε λέξη υπάρχουν ενδεχομένως περισσότεροι από ένας τρόποι να κινηθούμε πάνω στο αυτόματο διαβάζοντας τα γράμματα αυτής της λέξης, αφού σε κάθε βήμα ενδέχεται να έχουμε τη δυνατότητα να επιλέξουμε την επόμενή μας κατάσταση. Απορρίπτεται η λέξη w αν και μόνο αν κανείς από τους δυνατούς τρόπους κίνησης δεν καταλήγει σε τελική κορυφή.

Ορισμός 24. (Γλώσσα ενός ϵ-NFA)
Το σύνολο των λέξεων που αποδέχεται το μη ντετερμινιστικό αυτόματο με ϵ κινήσεις M ονομάζεται *η γλώσσα που αναγνωρίζει το αυτόματο* και συμβολίζεται με $L(M)$.

Παράδειγμα 31. Το ϵ-NFA του Σχήματος 10 αναγνωρίζει τη γλώσσα $0^*1^*2^*$. Για παράδειγμα, πώς



Σχήμα 10. Ένα ϵ-NFA για τη γλώσσα $0^*1^*2^*$.

αναγνωρίζει τη λέξη $0^2 1^3 2^5$ (συντομογραφία για τη λέξη 0011122222); Κάνει δύο μεταβάσεις από την πρώτη κορυφή στον εαυτό της διαβάζοντας τα δύο 0, μετά μεταβαίνει στη δεύτερη κορυφή *χωρίς να διαβάσει τίποτα*, μεταβαίνει από τη δεύτερη κορυφή στον εαυτό της τρεις φορές διαβάζοντας τα 1, μεταβαίνει στην τρίτη κορυφή χωρίς πάλι να διαβάσει τίποτα, και τέλος μεταβαίνει από την τρίτη κορυφή στον εαυτό της διαβάζοντας τα πέντε 2.

Άσκηση 36. Κατασκευάστε ένα ϵ-NFA που να αναγνωρίζει τη γλώσσα $(0101)^* \cup (111)^*$.

Άσκηση 37. Αν M_1 και M_2 είναι δύο DFA περιγράψτε πως θα τα χρησιμοποιήσετε αυτά για να κατασκευάσετε ένα ϵ-NFA για τη γλώσσα $L(M_1) \cup L(M_2)$.

Άσκηση 38. Αν M_1 και M_2 είναι δύο DFA περιγράψτε πως θα τα χρησιμοποιήσετε αυτά για να κατασκευάσετε ένα ϵ-NFA για τη γλώσσα $L(M_1)L(M_2)$. Αν το M_1 έχει ακριβώς μια τελική κατάσταση κατασκευάστε DFA για τη γλώσσα αυτή.

2.5 Ισοδυναμία ϵ-NFA και NFA

Εδώ θα αποδείξουμε ότι τα ϵ-NFA είναι ισοδύναμα με τα NFA, και άρα και με τα DFA. Για κάθε ϵ-NFA δηλ. υπάρχει ένα NFA χωρίς ϵ-κινήσεις που αναγνωρίζει την ίδια γλώσσα.

Ορισμός 25. (ϵ-μονοπάτι)
Ένα *ϵ-μονοπάτι* στο γράφημα ενός ϵ-NFA είναι μια πεπερασμένη ακολουθία διαδοχικών ϵ-ακμών (ακμών δηλ. που φέρουν την ετικέτα ϵ).

Ορισμός 26. (α-μονοπάτι)
Ένα *α-μονοπάτι*, όπου $\alpha \in \Sigma$, είναι ένα μονοπάτι στο γράφημα ενός ϵ-NFA που απαρτίζεται από ένα αρχικό ϵ-μονοπάτι, ακολουθούμενο από μια ακμή με ετικέτα α, ακολουθούμενο από ένα ϵ-μονοπάτι. Τα δύο ϵ-μονοπάτια μπορούν να είναι και κενά.

Θεώρημα 4. (Ισοδυναμία ϵ-NFA και NFA)
Για κάθε ϵ-NFA M υπάρχει ισοδύναμο NFA M'.

Απόδειξη. Για να μεταβούμε από ένα ϵ-NFA M σε ένα ισοδύναμο NFA M' κάνουμε τα εξής:

- Το σύνολο καταστάσεων του M' είναι ίδιο με του M.
- Η αρχική κατάσταση του M' είναι ίδια με αυτή του M.
- Οι τελικές καταστάσεις του M', το σύνολο των οποίων συμβολίζουμε με F', είναι όλες εκείνες τις καταστάσεις από τις οποίες μπορούμε να φτάσουμε σε κάποια κορυφή του F (το σύνολο τελικών καταστάσεων του M) με ένα ϵ-μονοπάτι. Επειδή ένα κενό μονοπάτι είναι ϵ-μονοπάτι, ο ορισμός αυτός συνεπάγεται ότι $F \subseteq F'$.
- Για κάθε κατάσταση q και γράμμα $\alpha \in \Sigma$ θέτουμε στο M' ως $\delta_{M'}(q, \alpha)$ το σύνολο όλων των καταστάσεων στις οποίες μπορεί κανείς να μεταβεί από την q στο M διανύοντας κάποιο α-μονοπάτι.
- Καταργούμε όλες τις ϵ-κινήσεις.

Έστω τώρα $w = w_1 w_2 \cdots w_n \in \Sigma^*$, $n \geq 0$. Πρέπει να δείξουμε ότι η λέξη w γίνεται δεκτή από το M αν και μόνο αν γίνεται δεκτή από το M.

Αν $w \in L(M)$ αυτό σημαίνει ότι υπάρχει τρόπος κίνησης πάνω στο M, διαβάζοντας τα γράμματα της λέξης w είτε ακολουθώντας ϵ-κινήσεις, ώστε να μεταβούμε από την q_0 σε κάποια κατάσταση $f \in F$. Σε αυτή την περίπτωση δείχνουμε ότι υπάρχει τρόπος κίνησης πάνω στο M τέτοιος που να οδηγεί από την q_0 σε κάποια κατάσταση του F'.

Έστω λοιπόν ότι $w \in L(M)$. Αυτό σημαίνει ότι υπάρχει μια ακολουθία καταστάσεων του M έστω $q_0, q_1, \ldots, q_{k-1}, q_k$, $k \geq 0$, τέτοια ώστε $q_k \in F$ και η μετάβαση από την q_j στην q_{j+1}, $0 \leq j < k$, γίνεται είτε με κάποιο γράμμα w_j είτε με μια ϵ-ακμή, και τα γράμματα w_j χρησιμοποιούνται όλα, από μια φορά το καθένα, και με τη σειρά.

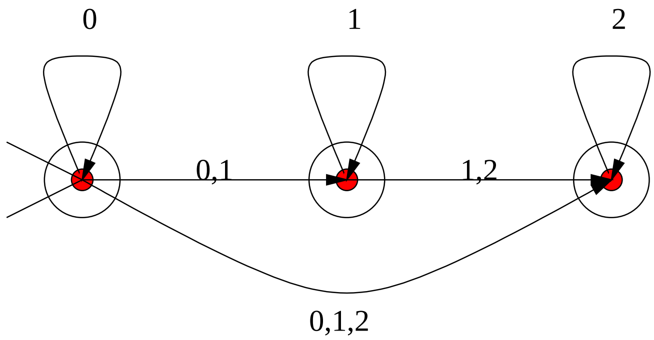
Χωρίζουμε την πεπερασμένη αυτή ακολουθία των μεταβάσεων $q_0 \to q_1, q_1 \to q_2, \ldots, q_{k-1} \to q_k$ σε κομμάτια $q_{a_0} \to q_{a_1}, q_{a_1} \to q_{a_2}, \ldots, q_{a_{n-1}} \to q_{a_n}$ (με $a_0 = 0$, $a_n = k$) τα οποία αντιστοιχούν σε ένα w_1-μονοπάτι, ακολουθούμενο από ένα w_2-μονοπάτι, κλπ., τελειώνοντας με ένα w_n-μονοπάτι. Από τον ορισμό του NFA M' προκύπτει ότι στο M' είναι δυνατή η μετάβαση από την κορυφή q_{a_0} στην q_{a_1} με μια w_1-ακμή, από την q_{a_1}

στην q_{a_2} με μια w_2-ακμή, κλπ. Επειδή η κατάσταση q_k παραμένει τελική κατάσταση και του M' προκύπτει ότι $w \in L(M')$.

Ο εγκλεισμός $L(M') \subseteq L(M)$ αφήνεται ως άσκηση (Άσκηση 39).
□

Άσκηση 39. Με το συμβολισμό της άνω απόδειξης δείξτε ότι αν μια λέξη αναγνωρίζεται από το NFA M' τότε αναγνωρίζεται και από το ϵ-NFA M.

Παράδειγμα 32. Με τη μέθοδο της απόδειξης του Θεωρήματος 4 το ϵ-NFA του Σχήματος 10 μετατρέπεται



Σχήμα 11. Το ϵ-NFA του Σχήματος 10 μετατετραμμένο σε NFA

στο NFA του Σχήματος 11.

Κεφάλαιο 3

Κανονικές εκφράσεις και γλώσσες

3.1 Κανονικές εκφράσεις και οι γλώσσες τους

Θα δώσουμε κατ΄ αρχήν ένα αναδρομικό ορισμό για το τι είναι μια *κανονική έκφραση* (regular expression) και το ποια γλώσσα παριστάνει.

Ορισμός 27. (Κανονικές εκφράσεις και οι γλώσσες τους)
Οι παρακάτω λέξεις ορίζονται να είναι *κανονικές εκφράσεις* με τις αντίστοιχες γλώσσες. (Με $L(\cdot)$ συμβολίζουμε τη *γλώσσα μιας έκφρασης*.)

- $\emptyset$, και $L(\emptyset) = \emptyset$
- ϵ, και $L(\epsilon) = \{\epsilon\}$
- α, για κάθε $\alpha \in \Sigma$, και $L(\alpha) = \{\alpha\}$.

Επίσης, αν r και s είναι κανονικές εκφράσεις τότε και οι ακόλουθες λέξεις είναι επίσης κανονικές εκφράσεις.

1. (r), και $L((r)) = L(r)$.
2. $(r+s)$, και $L((r+s)) = L(r) \cup L(s)$,
3. (rs), και $L((rs)) = L(r)L(s)$,
4. (r^*), και $L((r^*)) = L(r)^*$.

Μια γλώσσα L λέγεται κανονική αν υπάρχει κανονική έκφραση r με $L = L(r)$.

Παρατήρηση 10. Αν μπορούμε να παρελείψουμε τις παρανθέσεις χωρίς να αλλάξουμε το νόημα τότε το κάνουμε αυτό. Λαμβάνουμε υπόψιν ότι τη μεγαλύτερη προτεραιότητα έχει ο εκθέτης $*$, μετά έρχεται η συγκόλληση και τέλος η πρόσθεση. Για παράδειγμα η κανονική έκφραση $01^* + 10^*$ ερμηνεύεται ως $(0(1^*)) + (1(0^*))$. Αν r είναι μια κανονική έκφραση τότε χρησιμοποιούμε επίσης τη συντομογραφία r^+ αντί για rr^*.

Παράδειγμα 33. Τα παρακάτω είναι παραδείγματα κανονικών εκφράσεων και των γλωσσών τους.

Κανονική έκφραση	**Αντίστοιχη γλώσσα**
00	$\{00\}$
$(0+1)^*$	Όλες οι λέξεις πάνω από το $\Sigma = \{0, 1\}$
$(0+1)^*00(0+1)^*$	Όλες οι λέξεις πάνω από το $\Sigma = \{0, 1\}$ που έχουν δύο διαδοχικά μηδενικά
$(1+10)^*$	Ότι αρχίζει με 1 και δεν έχει διαδοχικά 0
$(0+\epsilon)(1+10)^*$	Ότι δεν έχει διαδοχικά 0
$0^*1^*2^*$	Λέξεις όπου τα 0 έρχονται πριν τα 1 κι αυτά πριν τα 2.

Άσκηση 40. Γράψτε κανονικές εκφράσεις για τις ακόλουθες γλώσσες:

1. Λέξεις που δεν περιέχουν τη μορφή 101.
2. Λέξεις όπου όλες οι εμφανίσεις διαδοχικών 0 συμβαίνουν πριν από οποιαδήποτε εμφάνιση διαδοχικών 1.

Άσκηση 41. Περιγράψτε τις γλώσσες που ορίζονται από τις κανονικές εκφράσεις:

1. $(11 + 0)^*(00 + 1)^*$
2. $(1 + 01 + 001)^*(\epsilon + 0 + 00)^*$
3. $(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10))^*$

Άσκηση 42. Δείξτε τις παρακάτω ταυτότητες όπου r, s, t είναι οποιεσδήποτε κανονικές εκφράσεις. Η ισότητα παρακάτω δε σημαίνει ότι οι δύο εκφράσεις είναι ίσες (και δεν είναι, μια και οι εκφράσεις είναι λέξεις, και ως τέτοιες διαφέρουν) αλλά ότι οι αντίστοιχες γλώσσες είναι ίσες.

1. $r + s = s + r$
2. $(r + s) + t = r + (s + t)$
3. $(rs)t = r(st)$
4. $r(s + t) = rs + rt$
5. $(r + s)t = rt + st$
6. $\emptyset^* = \epsilon$
7. $(r^*)^* = r^*$
8. $(\epsilon + r)^* = r^*$
9. $(r^*s^*)^* = (r + s)^*$

3.2 Κανονικότητα γλωσσών των αυτομάτων

Η βασική πρόταση είναι η ακόλουθη.

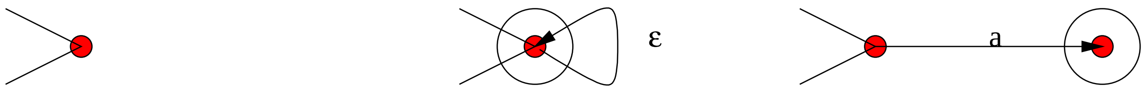
Θεώρημα 5. (Κανονικότητα γλωσσών των αυτομάτων)
Μια γλώσσα είναι κανονική αν και μόνο αν αναγνωρίζεται από κάποιο αυτόματο.

Παρατήρηση 11. Παρατηρείστε ότι στο Θεώρημα 5 δεν ξεκαθαρίζουμε για τι είδους αυτόματο μιλάμε. Έχουμε όμως ήδη αποδείξει ότι κάθε ϵ-NFA είναι ισοδύναμο με κάποιο NFA και κάθε NFA με κάποιο DFA. Αυτό σημαίνει ότι μια γλώσσα αναγνωρίζεται από ένα αυτόματο κάποιου είδους από τα τρία αν και μόνο αν αναγνωρίζεται από αυτόματο οποιουδήποτε είδους.

Απόδειξη. Θα δείξουμε ότι (α) Κάθε κανονική γλώσσα αναγνωρίζεται από κάποιο ϵ-NFA, και (β) Η γλώσσα που αναγνωρίζει κάθε DFA είναι κανονική.

(α) Χρησιμοποιούμε επαγωγή ως προς το μήκος της κανονικής έκφρασης r που παράγει τη γλώσσα. Αν πρόκειται για μια από τις εκφράσεις $\emptyset$, ϵ ή a, με $a \in \Sigma$, πολύ εύκολα φτιάχνουμε αυτόματα που τις αναγνωρίζουν όπως φαίνεται στο παρακάτω Σχήμα 12.

Αν τώρα έχουμε μια έφραση x του τύπου $r + s$, rs η r^*, τότε τα μήκη των r και s είναι αυτστηρά μικρότερα του $|x|$, άρα μπορούμε να υποθέσουμε επαγωγικά ότι έχουμε κάποια αυτόματα M και N που αναγνωρίζουν τις γλώσσες $L(r)$ και $L(s)$. Χρησιμοποιούμε τα M και N σα μαύρα κουτιά και μας ενδιαφέρει μόνο να 'βλέπουμε' απ΄ έξω τις αρχικές και τελικές τους καταστάσεις.



(a) (b) (c)

Σχήμα 12. Τα ϵ-NFA για τις εκφράσεις $\emptyset$ (a), ϵ (b) και a (c)

Στο Σχήμα 13 βλέπουμε στα (a) και (b) τα αυτόματα M και N που αντιστοιχούν στις εκφράσεις r και s. Στα (c), (d) και (e) βλέπουμε πως αυτά συνδυάζονται ώστε να φτιάξουν αυτόματα για τις γλώσσες $r+s$, rs και r^*.

Στο (c) ορίζουμε μια νέα αρχική κορυφή και την ενώνουμε με ϵ-ακμές με τις δύο αρχικές κορυφές των M και N. Οι τελικές καταστάσεις παραμένουν οι ίδιες.

Στο (d) αρχική κορυφή είναι αυτή του M του οποίου οι τελικές καταστάσεις συνδέονται με ϵ-ακμές με την αρχική του N. Τελικές καταστάσεις του συμπλέγματος είναι αυτές του N.

Στο (e) οι τελικές καταστάσεις του M συνδέονται με ϵ-ακμές με την αρχική κατάσταση του M. Αρχικές και τελικές καταστάσεις παραμένουν οι ίδιες.

(β) Έστω DFA $M = (Q, \Sigma, \delta, q_1, F)$, όπου $Q = \{q_1, \ldots, q_n\}$. Ορίζουμε για $i, j = 1, \ldots, n$, $k = 0, \ldots, n$, τις γλώσσες $R^k_{i,j}$ να είναι εκείνες οι λέξεις του Σ^* που είναι τέτοιες ώστε αν ξεκινήσουμε από την κορυφή q_i και τις ακολουθήσουμε τότε φτάνουμε στην κορυφή q_j **χωρίς να χρησιμοποιήσουμε κορυφή q_l με $l > k$.**

Είναι φανερό ότι

$$L(M) = \bigcup_{q_j \in F} R^n_{1,j}. \tag{3.1}$$

Με άλλα λόγια, αποδεκτές γίνονται εκείνες οι λέξεις που μας επιτρέπουν να φτάσουμε, ξεκινώντας από την αρχική κορυφή q_1, σε κάποια από τις κορυφές $q_j \in F$, χωρίς κανένα περιορισμό ως προς τις ενδιάμεσες κορυφές (επειδή ο άνω δείκτης είναι n, και, ούτως ή άλλως, δεν υπάρχουν κορυφές q_l, με $l > n$.

Θα δείξουμε με επαγωγή ως προς το k ότι οι γλώσσες $R^k_{i,j}$ είναι όλες κανονικές. Άρα κανονική είναι και η $L(M)$ αφού με βάση την (3.1) είναι **πεπερασμένη** ένωση από κανονικές γλώσσες, και η ένωση δύο κανονικών γλωσσών είναι εξ ορισμού κανονική.

Για $k = 0$ τώρα, παρατηρούμε ότι η απαίτηση, στον ορισμό της $R^k_{i,j}$ όσον αφορά το ποιες κορυφές δεν πρέπει να χρησιμοποιηθούν είναι ιδιαίτερα αυστηρή, αφού η συνθήκη $l > 0$ ισχύει για κάθε κορυφή $q_l \in Q$. Άρα έχουμε

$$R^0_{i,j} = \begin{cases} \{\alpha \in \Sigma : \ \ \delta(q_i, \alpha) = q_j\} & (i \neq j) \\ \{\alpha \in \Sigma : \ \ \delta(q_i, \alpha) = q_i\} \cup \{\epsilon\} & (i = j) \end{cases}$$

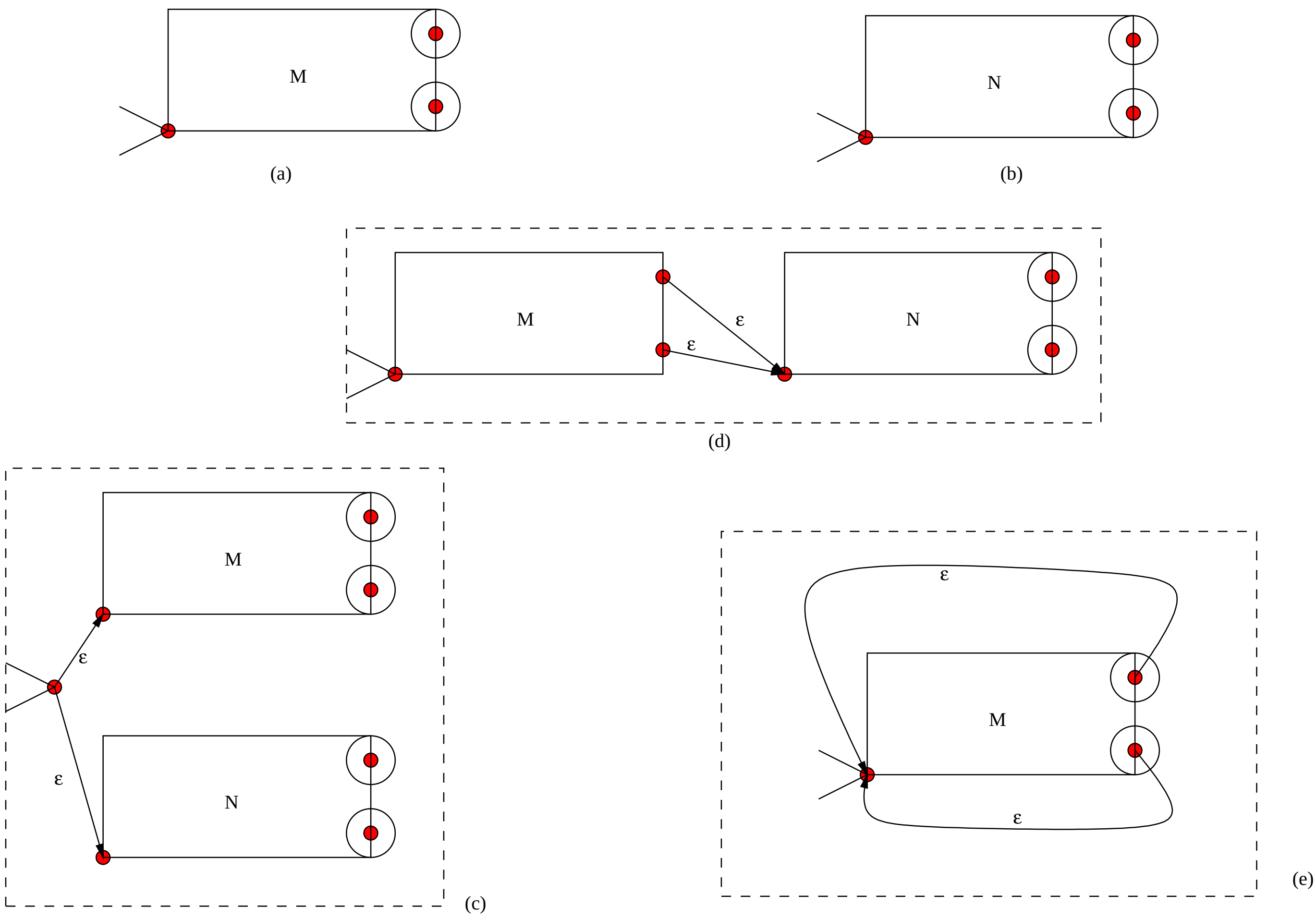
Αυτό σημαίνει ότι οι γλώσσες $R^0_{i,j}$ είναι πεπερασμένα σύνολα από σύμβολα του Σ ή το γράμμα ϵ. Κάθε ένα όμως από αυτά είναι κανονική γλώσσα άρα είναι τέτοια και η $R^0_{i,j}$.

Όσον αφορά το επαγωγικό βήμα, αν υποθέσουμε ότι οι $R^{k-1}_{i,j}$ είναι όλες κανονικές τότε το ίδιο συνάγουμε και για τις $R^k_{i,j}$ αφού παρατηρήσουμε ότι ισχύει η αναδρομική σχέση

$$R^k_{i,j} = R^{k-1}_{i,k}(R^{k-1}_{k,k})^* R^{k-1}_{k,j} \cup R^{k-1}_{i,j}. \tag{3.2}$$

Γιατί όμως ισχύει η (3.2);

Είναι φανερό ότι η γλώσσα $R^k_{i,j}$ είναι υπερσύνολο της $R^{k-1}_{i,j}$, αφού ο περιορισμός $l \leq k$, στον ορισμό της $R^k_{i,j}$, γίνεται ασθενέστερος (ισχύει πιο συχνά) όσο μεγαλώνει το k. Ποιες είναι όμως εκείνες οι λέξεις που ανήκουν στο σύνολο $R^k_{i,j}$ αλλά όχι στο $R^{k-1}_{i,j}$, οι λέξεις με άλλα λόγια της (συνολοθεωρητικής) διαφοράς $R^k_{i,j} \setminus R^{k-1}_{i,j}$; Είναι ακριβώς εκείνες οι λέξεις που οδηγούν από την κατάσταση q_i στην q_j, χωρίς να 'πατούν' σε κορυφή q_l, $l > k$, αλλά που πατούν τουλάχιστον μια φορά στην κορυφή q_k όως φαίνεται στο Σχήμα 14.



Σχήμα 13. (α) Αυτόματο για r, (β) για s, (ς) για $r+s$, (δ) για rs, (ε) για r^*

Μια τέτοια λέξη αντιστοιχεί σε ένα μονοπάτι πάνω στο DFA που σίγουρα 'πατάει' πάνω στην κορυφή q_k, ενδεχομένως και πάνω από μία φορά (στο Σχήμα 14 πατάει δύο φορές). Αν ονομάσουμε w μια τέτοια λέξη, και ονομάσουμε σ το πρόθεμα της w που αντιστοιχεί στο μονοπάτι από το q_i στο q_k, που δεν πατάει στην q_k, και τ το επίθεμα της w γαι το μονοπάτι $q_k \to q_j$ που δεν πατάει στην q_k, τότε η w γράφεται

$$w = \sigma v_1 \cdots v_t \tau,$$

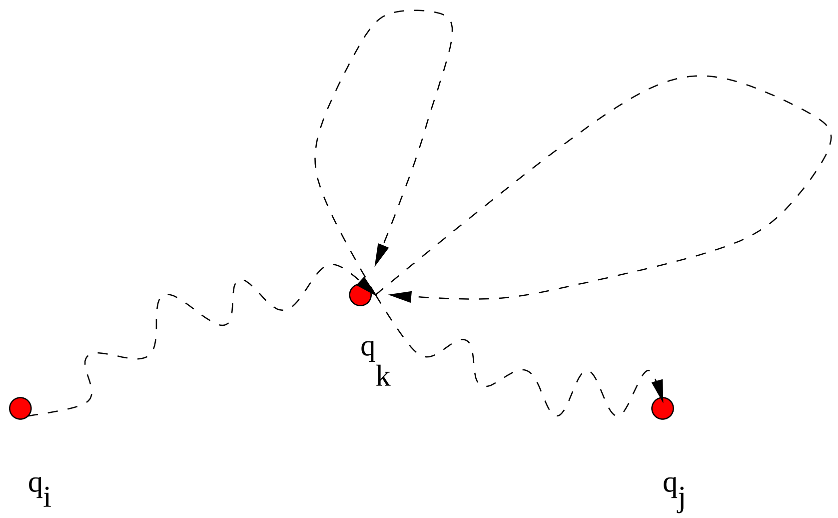
όπου οι λέξεις $v_1, \ldots, v_t$ αντιστοιχούν σε μονοπάτια που αρχίζουν και τελειώνουν από το q_k, χωρίς να πατούν στο q_k. Είναι δηλ. $\sigma \in R_{i,k}^{k-1}$, $v_l \in R_{k,k}^{k-1}$, $l = 1, \ldots, t$, και $\tau \in R_{k,j}^{k-1}$, έχουμε άρα δείξει τον εγκλεισμό

$$R_{i,j}^k \setminus R_{i,j}^{k-1} \subseteq R_{i,k}^{k-1} (R_{k,k}^{k-1})^* R_{k,j}^{k-1}.$$

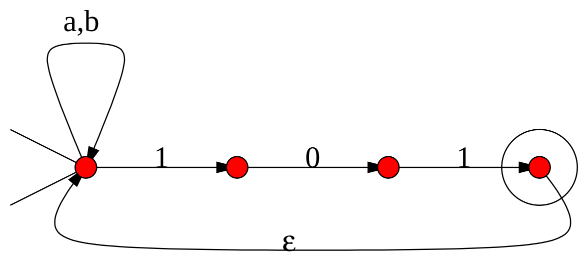
Ο αντίστροφος εγκλεισμός είναι ακόμη πιο εύκολος και παραλείπεται.
□

Άσκηση 43. Κατασκευάστε DFA για τις κανονικές εκφράσεις:

1. $10 + (0+11)0^*1$
2. $01\left(((10)^* + 111)^* + 0\right)^* 1$
3. $((0+1)(0+1))^* + ((0+1)(0+1)(0+1))^*$



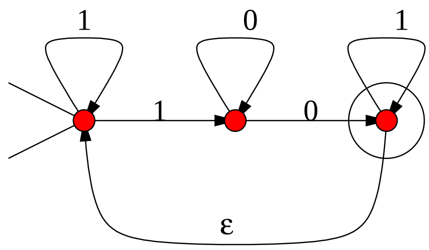
Σχήμα 14. Ένα μονοπάτι στο DFA που αντιστοιχεί σε λέξη του $R_{i,j}^k \setminus R_{i,j}^{k-1}$



Σχήμα 15. Το Σχήμα για την Άσκηση 44

Άσκηση 44. Δώστε μια κανονική έκφραση για τη γλώσσα που αναγνωρίζει το αυτόματο του Σχήματος 15.

Άσκηση 45. Δώστε μια κανονική έκφραση για τη γλώσσα που αναγνωρίζει το αυτόματο του Σχήματος 16.



Σχήμα 16. Το Σχήμα για την Άσκηση 45

3.3 Κλειστότητα κανονικών γλωσσών κάτω από απλές πράξεις

Από τον ορισμό των κανονικών εκφράσεων και των γλωσσών είναι φανερό ότι αν L_1 και L_2 είναι κανονικές γλώσσες τότε κανονική είναι και η $L_1 \cup L_2$ όπως και η γλώσσα L_1^*. Δεν είναι όμως καθόλου προφανές από τον ορισμό των κανονικών εκφράσεων το αν η συμπληρωματική γλώσσα

$$L_1^c = \{w \in \Sigma^* : \quad w \notin L_1\}$$

είναι κανονική. Γίνεται όμως και αυτό φανερό αν χρησιμοποιήσουμε το Θεώρημα 5 που λέει ότι μια γλώσσα είναι κανονική αν και μόνο αν υπάρχει ένα DFA που την αναγνωρίζει. Έστω λοιπόν ότι το DFA M αναγνωρίζει τη γλώσσα L_1. Φτιάνχουμε ένα DFA M' από το M κρατώντας τις ίδιες καταστάσεις, αλφάβητο, αρχική κατάσταση και ακμές του M αλλά κάνουμε όλες τις τελικές καταστάσεις του M μη τελικές και όλες τις μη τελικές καταστάσεις του M τις κάνουμε τελικές στο M'. (Εδώ πρέπει να είμαστε λίγο προσεκτικοί και να δουλεύουμε στο μορφή του M που δε χρησιμοποιεί συντομογραφία, αλλά που ικανοποιεί την αρχική απαίτηση για DFA, ότι δηλ. για κάθε κατάσταση και για κάθε γράμμα του αλφαβήτου υπάρχει ακριβώς μια ακμή.) Είναι φανερό τώρα ότι μια λέξη γίνεται δεκτή στο M' αν και μόνο αν *δε γίνεται δεκτή στο* M, άρα το M' είναι ένα DFA για τη γλώσσα L_1^c.

Είναι τώρα εύκολο να δείξουμε ότι και η γλώσσα $L_1 \cap L_2$ είναι κανονική. Αρκεί να δείξουμε ότι το συμπλήρωμά της είναι, και

$$(L_1 \cap L_2)^c = L_1^c \cup L_2^c,$$

και από αυτά που είπαμε προηγουμένως προκύπτει ο ισχυρισμός.

Ορισμός 28. *Αντικατάσταση* (substitution) σε ένα αλφάβητο Σ είναι μια οποιαδήποτε αντιστοίχιση των γραμμάτων του αλφαβήτου σε *γλώσσες* πάνω από το Σ, μια οποιαδήποτε συνάρτηση δηλ.

$$f : \Sigma \to 2^{\Sigma^*},$$

μια συνάρτηση δηλ. που παίρνει τιμές τυχόντα υποσύνολα του Σ^*. Αν για κάθε $\alpha \in \Sigma$ η γλώσσα $f(a)$ είναι κανονική, τότε και η αντικατάσταση ονομάζεται κανονική.

Έχοντας μια αντικατάσταση $f : \Sigma \to 2^{\Sigma^*}$ μπορούμε να επεκτείνουμε το πεδίο ορισμού της από γράμματα του Σ σε λέξεις του Σ^*. Αν $w = \alpha_1 \cdots \alpha_n$, $\alpha_i \in \Sigma$, είναι μια λέξη, τότε ορίζουμε (χρησιμοποιούμε το ίδιο σύμβολο f για να συμβολίσουμε και την επεκτεταμένη συνάρτηση)

$$f(w) = f(\alpha_1) \cdots f(\alpha_n),$$

η γλώσσα δηλ. που προκύπτει από τη συγκόλληση των γλωσσών $f(\alpha_1), \ldots, f(\alpha_n)$.

Αν L είναι μια γλώσσα ορίζουμε ως συνήθως

$$f(L) = \bigcup_{w \in L} f(w).$$

Θεώρημα 6. *Αν L κανονική, και η αντικατάσταση f είναι κανονική, τότε η $f(L)$ είναι κανονική γλώσσα.*

Σκίτσο απόδειξης. Αφού η L είναι κανονική τότε προκύπτει ως γλώσσα μιας κανονικής έκφρασης r. Είναι εύκολο να δούμε (αλλά δεν το κάνουμε με λεπτομέρεια εδώ) ότι η γλώσσα $f(L)$ είναι ακριβώς αυτή που προκύπτει αν τροποποιήσουμε την έκφραση r αντικαθιστώντας κάθε γράμμα α που εμφανίζεται στην r με μια κανονική έκφραση για τη γλώσσα $f(\alpha)$. Άρα η γλώσσα $f(L)$ δίδεται από κανονική έκφραση, άρα είναι κανονική.
$\square$

Άσκηση 46. Αποδείξτε το Θεώρημα 6 ακολουθώντας τη 'συνταγή' του σκίτσου απόδειξης που δόθηκε. Μπορείτε για παράδειγμα να χρησιμοποιήσετε επαγωγή ως προς το μήκος μιας κανονικής έκφρασης της κανονικής γλώσσας L.

Ορισμός 29. Μια αντικατάσταση f λέγεται *ομομορφισμός* αν για κάθε $\alpha \in \Sigma$ γλώσσα $f(\alpha)$ είναι μονοσύνολο.

Επειδή κάθε πεπερασμένο σύνολο λέξεων είναι κανονική γλώσσα (από τον ορισμό του τι είναι κανονική γλώσσα) από το προηγούμενο θεώρημα έπεται ότι οι ομομορφισμοί διατηρούν την κανονικότητα των γλωσσών.

Ορισμός 30. Αν f είναι ομομορφισμός και L είναι μια γλώσσα, η *αντίστροφη ομομορφική εικόνα* της L είναι η γλώσσα

$$f^{-1}(L) = \{w \in \Sigma^* : \quad f(w) \in L\}.$$

Θεώρημα 7. *Αν f ομομορφισμός και L κανονική γλώσσα τότε και η $f^{-1}(L)$ είναι κανονική.*

Απόδειξη. Έστω M ένα DFA που αναγνωρίζει την L. Φτιάχνουμε DFA M' ως εξής: κρατάμε ίδιες καταστάσεις και αρχικές και τελικές καταστάσεις με το M και ορίζουμε μόνο νέα συνάρτηση μετάβασης ως εξής: αν q είναι μια κατάσταση του M και $\alpha \in \Sigma$ ορίζουμε

$$\delta_{M'}(q, \alpha) = \delta_M(q, f(\alpha)).$$

Μεταβαίνουμε δηλ. στο M' σε εκείνη την κατάσταση όπου θα καταλήγαμε αν στο M ξεκινάγαμε από την κατάσταση q και ακολουθούσαμε τη *λέξη* $f(\alpha)$. Είναι φανερό ότι η λέξη w γίνεται αποδεκτή από το M' αν και μόνο αν η λέξη $f(w)$ γίνεται αποδεκτή από το M, άρα το DFA αναγνωρίζει τη γλώσσα $f^{-1}(L)$, οπότε αυτή είναι κανονική.
□

Άσκηση 47. Αν L είναι κανονική γλώσσα τότε και η γλώσσα

$$L^R = \left\{x \in \Sigma^* : \ x^R \in L\right\}$$

είναι κανονική. Με x^R συμβολίζουμε τη λέξη x με τα γράμματά της σε ανάποδη σειρά. Π.χ. $011^R = 110$. (*Υπόδειξη:* Δουλέψτε πάνω σε μια κανονική έκφραση για την L.)

3.4 Το Λήμμα Άντλησης και μη κανονικές γλώσσες

Υπάρχουν γλώσσες που δεν είναι κανονικές. Αυτό είναι προφανές για πολύ γενικούς λόγους, που δεν έχουν τόσο πολύ να κάνουν με τη φύση των κανονικών γλωσσών, παρά μόνο με το γεγονός ότι κάθε κανονική γλώσσα επιδέχεται μια *πεπερασμένη περιγραφή*. Αυτή μπορεί να είναι είτε μια κανονική έκφραση, είτε η δομή ενός DFA που την αναγνωρίζει, για παράδειγμα.

Πόσες όμως διαφορετικές πεπερασμένες περιγραφές υπάρχουν;

Είναι φανερό ότι αυτές είνάι άπειρες το πλήθος (αφού π.χ. υπάρχουν οσοδήποτε μεγάλες κανονικές εκφράσεις), αλλά αυτό δεν είναι επαρκής πληροφορία για το πλήθος τους. Είναι πολύ πιο ακριβές το ότι υπάρχουν *αριθμήσιμες* το πλήθος πεπερασμένες περιγραφές, μπορούν δηλ. όλες οι πεπερασμένες περιγραφές να απαριθμηθούν: η περιγραφή 1, η περιγραφή 2, ..., η περιγραφή N, ..., με τρόπο ώστε να μη μείνει καμιά περιγραφή απ΄ έξω. Αυτός είναι ο ορισμός ενός αριθμήσιμου συνόλου. Ένα σύνολο δηλ. λέγεται αριθμήσιμο αν υπάρχει μια 1-1 απεικόνιση του συνόλου αυτού στους φυσικούς αριθμούς (βεβαιωθείτε ότι το καταλαβαίνετε αυτό).

Το γιατί το σύνολο όλων των πεπερασμένων εκφράσεων (πάνω από ένα σταθερό αλφάβητο Σ) είναι αριθμήσιμο είναι απλό. Αφού το Σ είναι πεπερασμένο υπάρχουν πεπερασμένο το πλήθος εκφράσεις που μπορούν να φτιάξουμε με μήκος n, για κάθε n. Για την ακρίβεια, υπάρχουν το πολύ $|\Sigma|^n$ τέτοιες εκφράσεις. Για να απαριθμήσουμε λοιπόν το σύνολο όλων των πεπερασμένων εκφράσεων, αριθμούμε πρώτα τις εκφράσεις μήκους 1, μετά αυτές μήκους 2, και συνεχίζουμε κατ΄ αυτόν τον τρόπο, ώστε δε μένει τίποτα αμέτρητο.

Είναι αρκετά πιο δύσκολο (και παραλείπουμε την απόδειξη) να δούμε ότι το σύνολο όλων των υποσυνόλων οποιουδήποτε άπειρου συνόλου *δεν είναι αριθμήσιμο*. Έτσι, το σύνολο όλων των γλωσσών πάνω από το Σ, δηλ. το σύνολο όλων των υποσυνόλων του άπειρου συνόλου Σ^* δεν είναι αριθμήσιμο. Άρα υπάρχει κάποια γλώσσα που δεν έχει αντίστοιχη πεπερασμένη περιγραφή, οπότε δεν είναι κανονική.

Το παραπάνω είναι ένα πολύ γενικό επιχείρημα, όπως είπαμε, και δεν εξαρτάται σχεδόν καθόλου από το τι εννοούμε κανονική γλώσσα, παρά μόνο ότι, ό,τι κι αν εννοούμε, η κανονική γλώσσα μπορεί να περιγραφεί πλήρως με μια πεπερασμένη ακολουθία από γράμματα, διαλεγμένα από ένα πεπερασμένο αλφάβητο. Με τον ίδιο τρόπο φαίνεται, π.χ. ότι υπάρχουν συναρτήσεις $f : \mathbf{N} \to \mathbf{N}$ που δεν είναι υπολογίσιμες από πρόγραμμα, ουσιαστικά ότι κι αν εννούμε με αυτό. Ας πούμε για παράδειγμα ότι θεωρούμε μια τέτοια συνάρτηση υπολογίσιμη αν υπάρχει ένα πρόγραμμα σε κάποια γλώσσα προγραμματισμού, ας πούμε την C που υπολογίζει τη συνάρτηση αυτή. Μα ένα πρόγραμμα αποτελεί πεπερασμένη περιγραφή της συνάρτησης, άρα, επειδή το πλήθος των συναρτήσεων $f : \mathbf{N} \to \mathbf{N}$ δεν είναι αριθμήσιμο, υπάρχει κάποια στην οποία δεν αντιστοιχεί πρόγραμμα, που δεν είναι δηλ. υπολογίσιμη.

Δε βοηθάει όμως αυτό το επιχείρημα στο να αποδείξει κανείς ότι συγκεκριμένες γλώσσες δεν είναι κανονικές. Αν και γνωρίζουμε δηλ. ότι οι περισσότερες γλώσσες (υπό μία έννοια) δεν είναι κανονικές, είναι παρ' όλα αυτά δύσκολο να δείξουμε αυτό για μια συγκεκριμένη γλώσσα που μας δίδεται, π.χ. για την

$$L_1 = \{0^n 1^n : \quad n = 0, 1, 2, \ldots\}.$$

Αυτή η γλώσσα απαρτίζεται από όλες τις λέξεις που έχουν αρχικά μια ομάδα από μηδενικά και μετά μια ίση σε μήκος ομάδα από άσσους, και τίποτε άλλο. Θα δούμε σε λίγο ένα τρόπο απόδειξης του ότι η L_1 δεν είναι κανονική. Διαισθητικά όμως ο λόγος είναι ο εξής (το παρακάτω *δεν* αποτελεί απόδειξη): ένα αυτόματο είναι ουσιαστικά ισοδύναμο με ένα υπολογιστή σαν κι αυτούς που ξέρουμε με μόνο τον περιορισμό ότι ο υπολογιστής αυτός έχει πεπερασμένη και σταθερή μνήμη, δεν εξαρτάται δηλ. η ποσότητα της μνήμης του από το πρόβλημα που έχει να λύσει. Αν είχαμε ένα πρόγραμμα που τρέχει σε ένα τέτοιο υπολογιστή και το οποίο αναγνωρίζει τη γλώσσα L_1, αυτό το πρόγραμμα θα είχε πρόσβαση σε μνήμη της οποίας το μήκος δε μπορεί να εξαρτάται από το μήκος της λέξης προς αναγνώριση. Όμως, δε μπορούμε να φανταστούμε ένα πρόγραμμα (αλγόριθμο) που θα αποφασίζει αν μια λέξη ανήκει στην L_1 ή οχι χωρίς να 'θυμάται' κάπου το πόσα μηδενικά έχει δει. Κι αυτό δε γίνεται αν ο αριθμός των μηδενικών είνάι πολύ μεγάλος, γιατί όσο αυξάνει το πλήθος των μηδενικών τόσο αυξάνει, χωρίς άνω φράγμα, το πλήθος των ψηφίων (η μνήμη) που χρειαζόμαστε για να αποθηκεύσουμε τον αριθμό αυτό.

Είναι δύσκολο να μετατρέψουμε το παραπάνω διαισθητικό επιχείρημα σε απόδειξη, οπότε η μέθοδος που ακολουθούμε για να δείξουμ εότι η L_1 δενείναι κανονική είναι αρκετά διαφορετική. Χρησιμοποιούμε το ακόλουθο πολύ χρήσιμο λήμμα.

Θεώρημα 8. *(Λήμμα Άντλησης – Pumping Lemma)*
Έστω ότι η L είναι κανονική. Τότε υπάρχει φυσικός αριθμός n (ο οποίος δε χρειάζεται να είναι μεγαλύτερος από τον ελάχιστο αριθμό καταστάσεων ενός DFA που αναγνωρίζει την L) ώστε για κάθε λέξη $z \in L$ με $|z| \ge n$, μπορούμε να γράψουμε

$$z = uvw, \quad (u, v, w \in \Sigma^*, \ |v| \ge 1, |uv| \le n),$$

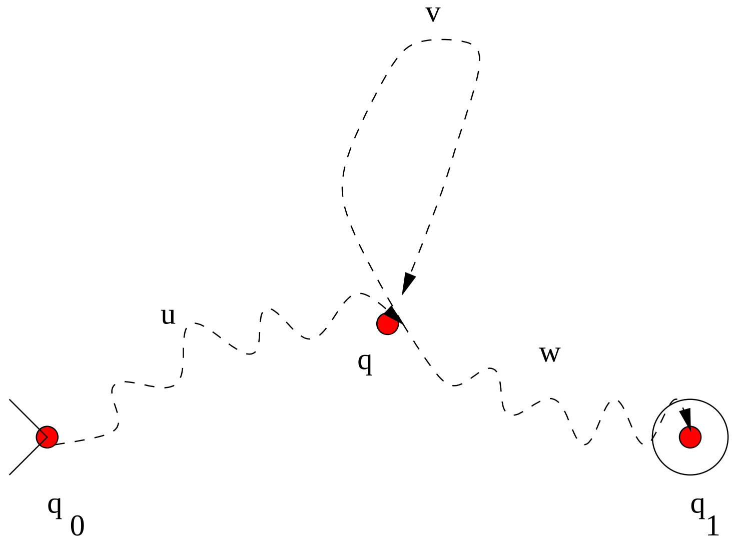
ούτως ώστε για κάθε $i = 0, 1, 2, \ldots$, η λέξη $uv^iw \in L$ (αυτή είναι η λέξη που αρχίζει με u ακολουθείται από την v οποιοδήποτε πεπερασμένο αριθμό από φορές – ακόμη και 0–και τελειώνει με w).

Απόδειξη: Έστω M ένα DFA με ελάχιστο αριθμό καταστάσεων που αναγνωρίζει τη γλώσσα L, και έστω n το πλήθος των καταστάσεων του M. Αν η λέξη z αναγνωρίζεται τότε ξεκινώντας από την αρχική κορυφή q_0 καταλήγουμε διαβάζοντας την z, με $|z| = m \ge n$, σε κάποια τελική κορυφή q_1 διανύοντας ένα μονοπάτι πάνω στο M:

$$q_0 = q_{i_1} \to q_{i_2} \to \cdots \to q_{i_{m+1}} = q_1$$

όπου το πλήθος των ακμών είναι m (μια ακμή για κάθε γράμμα της z) και άρα το πλήθος των κορυφών είναι $m + 1 > n$. Συμπεραίνουμε ότι υπάρχουν κάποιες κορυφές που εμφανίζονται τουλάχιστον δύο φορές στο μονοπάτι. Από αυτές τις κορυφές ονομάζουμε q αυτή που πρωτοεμφανίζεται για δεύτερη φορά στο μονοπάτι και ονομάζουμε v τη λέξη που μεσολαβεί μέχρι την επόμενη εμφάνιση της q, ενώ u ονομάζομε το πρόθεμα της z μέχρι την πρώτη εμφάνιση της q και w ονομάζομε το επίθεμα της z μετά τη δεύτερη εμφάνιση της q (βλ. Σχήμα 17).

Είναι φανερό ότι $|v| \ge 1$ και ότι $|uv| \le n$, αφού σίγουρα όταν θα έχουμε διαβάσει το n-οστό γράμμα της z θα έχουμε ήδη δεί τουλάχιστον μια κορυφή δύο φορές, και ανάμεσα σε αυτές που έχουμε δεί δύο φορές η πρώτη είναι εξ ορισμού η q. Επίσης είναι φανερό ότι το βρόγχο που ξεκινά και τελειώνει στην q μπορούμε να μην το διανύσουμε καθόλου (οπότε δείχνουμε ότι η λέξη $uw \in L$) ή να τον διανύσουμε όσες φορές θέλουμε, ας πούμε i. Άρα η λέξη $uv^iw \in L$, όπως οφείλαμε να δείξουμε.
□



Σχήμα 17. Το μονοπάτι που αντιστοιχεί στη λέξη uvw

Πώς χρησιμοποιούμε το Λήμμα Άντλησης για να δείξουμε ότι η γλώσσα L_1 δεν είναι κανονική; Υποθέτουμε ότι είναι και καταλήγουμε σε άτοπο. Αν είναι λοιπόν, υπάρχει, από το Λήμμα Άντλησης, ένας φυσικός αριθμός n, τέτοις ώστε αν $z \in L_1$ και $|z| \geq n$ τότε ισχύουν τα συμπεράσματα του Λήμματος. Έχουμε $z = 0^n 1^n \in L_1$ και $|z| \geq n$, άρα $z = uvw$, με $|uv| \leq n$ και μη κενό v, τέτοια ώστε για κάθε $i = 0, 1, \ldots$ έχουμε $uv^i w \in L_1$. Αυτό όμως δε γίνεται μια και η λέξη $uv^2 w = uvvw$ έχει σίγουρα περισσότερα 0 απ' ότι 1, αφού, λόγω του ότι $|uv| \leq n$, και το u και το v έχουν μόνο μηδενικά μέσα τους.

Ας δούμε τώρα άλλη μια εφαρμογή του λήμματος άντλησης σε παρόμοιο πρόβλημα. Για κάθε λέξη $x = \alpha_1 \cdots \alpha_n$, $\alpha_i \in \Sigma$, ορίζουμε την *αντεστραμμένη* λέξη $x^R = \alpha_n \cdots \alpha_1$, να είναι η x με αντεστραμμένη τη σειρά των γραμμάτων του. Ορίζουμε τη γλώσσα

$$L_2 = \left\{x \in (0+1)^* : \quad x = x^R\right\},$$

να απαρτίζεται από όλες εκείνες τις λέξεις που διαβάζονται το ίδιο από αριστερά και από τα δεξιά.

Δείχνουμε τώρα ότι η L_2 δεν είναι κανονική. Έστω ότι είναι και έστω n ο φυσικός αριθμός του οποίου η ύπαρξη προκύπτει από το Λήμμα Άντλησης. Ορίζουμε τη λέξη $z = 1^n 0 1^n$ που είναι στην L_2. Γράφεται τότε η z ως $z = uvw$, με μη κενό v και $|uv| \leq n$, οπότε και οι λέξεις u, v έχουν μέσα μόνο 1, ώστε $uv^i w \in L_2$, για $i = 0, 1, 2, \ldots$. Αλλά προφανώς η λέξη $uw \notin L_2$ αφού αφαιρώντας το v αφαιρέσαμε κάποιους άσσους από την αριστερή ομάδα άσσων αλλά όχι από τη δεξιά ομάδα, οπότε έχουμε καταλήξει σε άτοπο, άρα η L_2 δεν είναι κανονική.

Άσκηση 48. Δείξτε ότι η γλώσσα $\left\{xx : \quad x \in \{0,1\}^*\right\}$ δεν είναι κανονική.

Άσκηση 49. Ποιες από τις παρακάτω γλώσσες του $(0+1)^*$ είναι κανονικές;

1. $\left\{0^{2n} : \quad n \geq 1\right\}$
2. $\{0^m 1^n 0^{m+n} : \quad m, n \geq 1\}$
3. Οι λέξεις που δεν έχουν τρία διαδοχικά μηδενικά
4. Λέξεις με τόσα μηδενικά όσα και άσσους.
5. $\left\{x w x^R : \quad x, w \in (0+1)^+\right\}$

Κεφάλαιο 4

Αλγόριθμοι για DFA.

4.1 Πότε ένα DFA αναγνωρίζει κενή ή άπειρη γλώσσα

Δοθέντος ενός DFA M καλούμαστε να αποφασίσουμε *με αλγοριθμικό τρόπο* για το αν

1. Υπάρχει έστω και μία λέξη που αναγνωρίζεται από το M,

2. Υπάρχουν άπειρες λέξεις που αναγνωρίζονται από το M.

Και στις δύο περιπτώσεις καλούμαστε να απαντήσουμε απλώς με ένα ναι ή όχι, και δε ζητούμε να βρούμε μία ή περισσότερες (πόσο μάλλον άπειρες) λέξεις για να αποδείξουμε τα λεγόμενά μας.

Πρέπει δηλ. να βρούμε ένα τρόπο να απαντήσουμε αν η γλώσσα του M είναι κενή ή όχι, και αν είναι άπειρη ή όχι ($L(M) = \emptyset$; και $|L(M)| = \infty$;). Και πρέπει ο τρόπος απόφασης να είναι αλγοριθμικός, δηλ. να μπορούμε να γράψουμε ένα πρόγραμμα στον υπολογιστή το οποίο, όποια και να είναι η απάντηση, να μπορεί να τη βρίσκει. Αυτό σημαίνει ότι σε κάθε περίπτωση (όποια και να είναι η απάντηση) πρέπει το πρόγραμμα αυτό (α) να τελειώσει και (β) να δώσει τη σωστή απάντηση.

Η απαίτηση για το (β) είναι προφανής στον περισσότερο κόσμο αλλά δεν είναι ίσως φανερό τι εννοούμε με το (α). Είναι λοιπόν χρήσιμο να τονίσουμε ότι ο ακόλουθος αλγόριθμος για να αποφασίσουμε το ερώτημα $L(M) = \emptyset$ δεν είναι αποδεκτός:

> Με τον υπολογιστή μας απαριθμούμε όλες τις λέξεις του Σ^* ως εξής. Πρώτα απαριθμούμε όλες τις λέξεις μήκους 0 (υπάρχει μόνο μία, η κενή λέξη ϵ), μετά απαριθμούμε τις λέξεις μήκους 1 (υπάρχουν τόσες όσα και α γράμματα του Σ, δηλ. το πλήθος τους είναι $|\Sigma|$), μετά τις λέξεις μήκους 2 (υπάρχουν ακριβώς $|\Sigma|^2$ τέτοιες), κ.ο.κ. Με αυτό τον τρόπο απαρίθμησης διανύουμε όλες τις λέξεις του Σ^*, χωρίς να ξαχνάμε καμιά. Για κάθε λέξη θα έρθει κάποτε η σειρά να την εξετάσουμε. *Παράλληλα* με την απαρίθμηση προγραμματίζουμε τον υπολογιστή μας να εξετάζει κάθε μια από τις αριθμούμενες λέξεις για το αν περνάνε από το M ή όχι. Αν έστω και μια βρεθεί που να αναγνωρίζεται από το M τότε σταματάέι ο αλγόριθμος και απαντά ΝΑΙ (στο ερώτημα αν $L(M) = \emptyset$), αλλιώς συνεχίζει.

Είναι φανερό ότι αυτή η μέθοδος κάνει, κατά κάποιο τρόπο, μισή δουλειά, μια και δεν είναι ποτέ δυνατό να απαντήσει ΟΧΙ, αφού δεν ξέρουμε, όσες λέξεις και να έχουμε δει μέχρι στιγμής, για το αν υπάρχει λέξη της $L(M)$ που να βρίσκεται παρακάτω στην αρίθμησή μας, π.χ. να έχει μεγαλύτερο μήκος απ΄ ό,τι έχουμε εξετάσει μέχρι στιγμής. Αν η απάντηση στο ερώτημα είναι ΝΑΙ τότε, αργά ή γρήγορα, ο αλγόριθμός μας θα απαντήσει ΝΑΙ, δε συμβαίνει όμως το ίδιο και με το ΟΧΙ.

Πώς θα μπορούσε κάπως να διορθωθεί ο αλγόριθμος που μόλις περιγράψαμε;

Αν είχαμε ένα τρόπο, με δεδομένη την περιγραφή του M, να ξέρουμε πόσο μεγάλη (το πολύ) είναι η μικρότερη σε μήκος λέξη της $L(M)$, αν μια τέτοια λέξη υπάρχει, τότε θα προγραμματίζαμε τον υπολογιστή μας να σταματάει την απαρίθμηση όταν έχει περάσει αυτό το φράγμα και δεν έχει βρει ακόμη λέξη της $L(M)$, αφού είναι πλέον σίγουρο ότι όσο και να συνεχίσει δε θα βρει άλλη. Αυτό το σκοπό, και για τα δύο ερωτήματα που μας απασχολούν, εξυπηρετεί το παρακάτω θεώρημα, που είναι ουσιαστικά πόρισμα του Λήμματος Άντλησης.

Θεώρημα 9. *Αν M είναι ένα DFA με n καταστάσεις τότε*

1. *$L(M) \neq \emptyset$ αν και μόνο αν υπάρχει $w \in L(M)$ με $|w| < n$.*
2. *$|L(M)| = \infty$ αν και μόνο αν υπάρχει $w \in L(M)$ με $n \leq |w| < 2n$.*

Απόδειξη. 1. Αν υπάρχει $w \in L(M)$ με $|w| < n$ τότε προφανώς $L(M) \neq \emptyset$. Αντίστροφα, έστω $L(M) \neq \emptyset$ και $w \in L(M)$ έχευ ελάχιστο μήκος. Αν $|w| \geq n$ τότε, από το Λήμμα Άντλησης, υπάρχει σπάσιμο $w = uvz$ με $v \neq \epsilon$ τέτοιο ώστε για κάθε $i \geq 0$ έχουμε $uv^iz \in L(M)$. Για $i = 0$ παίρνουμε $uz \in L(M)$, η οποία όμως λέξη έχει μήκος μικρότερο της $w = uvz$ η οποία είχε εξ αρχής υποτεθεί ότι έχει ελάχιστο μήκος, πράγμα άτοπο, άρα $|w| < n$.

2. Αν υπάρχει $w \in L(M)$ με $n \leq |w| < 2n$ τότε (χρησιμοποιώντας μόνο την ανισότητα $n \leq |w|$) από το Λήμμα Άντλησης το w σπάει σε $w = uvz$, $v \neq \epsilon$, έτσι ώστε οι λέξεις uv^iz, $i \geq 0$, ανήκουν όλες στην $L(M)$. Αλλά αυτές είναι άπειρες το πλήθος, άρα $|L(M)| = \infty$. Αντίστροφα, έστω $|L(M)| = \infty$ και υποθέσουμε ότι δεν υπάρχει λέξη w με μήκος από n έως και $2n-1$, ας πάρουμε w να είναι λέξη με ελάχιστο μήκος μεγαλύτερο ή ίσο του $2n$ (τέτοιες λέξεις υπάρχουν αναγκαστικά αφού η $L(M)$ έχει υποτεθεί άπειρη γλώσσα). Το Λήμμα Άντλησης εφαρμόζεται και πάλι αφού $|w| \geq 2n \geq n$ άρα η λέξη w γράφεται $w = uvz$, με $|uv| \leq n$ και $v \neq \epsilon$ και $uv^iz \in L(M)$ (για $i \geq 0$). Άρα $uz \in L(M)$ και αφού $|uvz| \geq 2n$ και $|v| \leq n$ συμπεραίνουμε ότι $|uz| \geq n$, άρα (έχουμε υποθέσει ότι δεν υπάρχουν λέξεις στην $L(M)$ με μήκος από n έως και $2n-1$) $|uz| \geq 2n$, πράγμα που αντιφάσκει με το ότι η w έχει ελάχιστο μήκος ανάμεσα στις λέξεις της $L(M)$ με μήκος τουλάχιστον $2n$.
□

Είμαστε τώρα σε θέση να δείξουμε το ακόλουθο αποτέλεσμα.

Θεώρημα 10. *Τα ακόλουθα ερωτήματα είναι αλγοριθμικά αποφασίσιμα:*

1. *Αναγνωρίζει το DFA M μια μη κενή γλώσσα;*
2. *Αναγνωρίζει το DFA M μια άπειρη γλώσσα;*
3. *Αναγνωρίζουν τα DFA M_1 και M_2 την ίδια γλώσσα;*

Απόδειξη. 1. Απαριθμούμε όλες τις λέξεις μήκους μέχρι και n με γράμματα από το Σ (υπάρχουν ακριβώς $1 + |\Sigma| + |\Sigma|^2 + \cdots + |\Sigma|^n$ τέτοιες λέξεις) και τις ελέγχουμε όλες αν περνάνε από το αυτόματο M. Αν έστω και μια από αυτές αναγνωρίζεται τότε απαντάμε ΝΑΙ, αλλιώς απαντάμε ΟΧΙ. Η μέθοδος είναι σωστή με βάση το Θεώρημα 9.1.

2. Απαριθμούμε όλες τις λέξεις μήκους από n έως και $2n-1$ με γράμματα από το Σ. Αν έστω και μια από αυτές αναγνωρίζεται τότε απαντάμε ΝΑΙ, αλλιώς απαντάμε ΟΧΙ. Η μέθοδος είναι σωστή με βάση το Θεώρημα 9.2.

3. Θέλουμε να απαντήσουμε στο ερώτημα

$$L(M_1) \neq L(M_2).$$

Η απάντηση σε αυτό είναι ΝΑΙ αν και μόνο αν η ακόλουθη γλώσσα είναι μη κενή

$$(L(M_1) \cap L(M_2)^c) \cup (L(M_1)^c \cap L(M_2)). \tag{4.1}$$

Μπορούμε όμως αλγοριθμικά να κατασκευάσουμε ένα DFA M που αναγνωρίζει ακριβώς αυτή τη γλώσσα, οπότε μετά χρησιμοποιούμε τον αλγόριθμο του σκέλους 1 αυτού του Θεωρήματος για να αποφασίσουμε αν η γλώσσα της (4.1) είναι κανή ή όχι. Το αυτόματο M κατασκευάζεται ως εξής. Παρατηρούμε ότι η (4.1) γράφεται

$$(L(M_1)^c \cup L(M_2))^c \cup (L(M_1) \cup L(M_2)^c)^c$$

και ότι αυτή η έκφραση χρησιμοποιεί μόνο ενώσεις και συμπληρώματα. Αν έχουμε ένα DFA N και θέλουμε να φτιάξουμε ένα DFA N' για τη συμπληρωματική γλώσσα τότε απλά αλλάζουμε τις τελικές καταστάσεις του N σε μη τελικές και τις μη τελικές σε τελικές. Για να φτιάξουμε ένα DFA για την ένωση δύο γλωσσών (των οποίων ξέρουμε κάποια DFA) φτιάχνουμε πρώτα ένα ϵ-NFA για αυτή την ένωση και το μετατρέπουμε στη συνέχεια σε DFA. Όλα αυτά γίνονται αλγοριθμικά.
□

4.2 Σχέσεις ισοδυναμίας για γλώσσες και αυτόματα. Θεώρημα Myhill–Nerode

Πάνω στο σύνολο Σ^* όλων των λέξεων ορίζονται φυσιολογικά οι εξής σχέσεις ισοδυναμίας

1. Αν $L \subseteq \Sigma^*$ είναι μια γλώσσα η σχέση R_L ορίζεται ως: $x\ R_L\ y$ αν και μόνο αν για κάθε $z \in \Sigma^*$ έχουμε

$$xz \in L \Leftrightarrow yz \in L,$$

   δηλ. τα xz, yz είτε ανήκουν και τα δυο στην L είτε κανένα.

2. Αν M είναι ένα DFA τότε ορίζουμε $x\ R_M\ y$ αν και μόνο αν

$$\delta(q_0, x) = \delta(q_0, y),$$

   όπου q_0 είναι η αρχική κατάσταση του M και $\delta(\cdot, \cdot)$ είναι η συνάρτηση μετάβασης του M. Ισχύει δηλ. $x\ R_M\ y$ αν και μόνο αν το αυτόματο M καταλήγει στην ίδια κορυφή αφού διαβάσει το x και αφού διαβάσει το y, ξεκινώντας πάντα από την αρχική του κορυφή.

Άσκηση 50. Δείξτε ότι οι σχέσεις R_L και R_M είναι σχέσεις ισοδυναμίας.

Άσκηση 51. Δείξτε ότι η γλώσσα L είναι ένωση κάποιων κλάσεων ισοδυναμίας της R_L (ανεξάρτητα από το αν είναι η L κανονική ή όχι). Κάθε R_L-κλάση δηλ. είτε περιέχεται πλήρως στην L είτε δεν περιέχει λέξεις της L.

Ορισμός 31. *Δείκτης* μιας σχέσης ισοδυναμίας R λέγεται ο πληθάριθμος του συνόλου των κλάσεων ιδοσυναμίας της R.

Άσκηση 52. Δείξτε ότι ο δείκτης της R_M είναι πεπερασμένος και ίσος με το πλήθος των κορυφών του M που είναι προσπελάσιμες από την αρχική κορυφή q_0. Μια κορυφή v λέγεται προσπελάσιμη αν υπάρχει λέξη w τ.ώ. $\delta(q_0, w) = v$.

Ορισμός 32. Μια σχέση R ορισμένη πάνω στο Σ^* λέγεται *δεξιά αναλλοίωτη* (right invariant) αν για κάθε $x, y, z \in \Sigma^*$ ισχύει

$$x\ R\ y \Rightarrow xz\ R\ yz. \tag{4.2}$$

Άσκηση 53. Δείξτε ότι οι σχέσεις R_L και R_M που ορίσαμε παραπάνω είναι δεξιά αναλλοίωτες.

Άσκηση 54. Αν M είναι ένα DFA δείξτε ότι η γλώσσα $L(M)$ που αναγνωρίζει το M είναι μια ένωση κλάσεων ισοδυναμίας μιας σχέσης ισοδυναμίας με πεπερασμένο δείκτη. (*Υπόδειξη:* Εξετάστε την R_M.)

Θεώρημα 11. (Θεώρημα Myhill–Nerode) *Αν $L \subseteq \Sigma^*$ τα ακόλουθα είναι ισοδύναμα:*

1. *Η γλώσσα L είναι κανονική.*

2. *Η γλώσσα L είναι ένωση κάποιων κλάσεων ισοδυναμίας μιας δεξιά αναλλοίωτης σχέσης ισοδυναμίας R με πεπερασμένο δείκτη.*

3. *Η σχέση R_L έχει πεπερασμένο δείκτη.*

Απόδειξη.

$1 \Rightarrow 2$
Αν η L είναι κανονική τότε αναγνωρίζεται από ένα DFA M και το ζητούμενο είναι η Άσκηση 54.

$2 \Rightarrow 3$
Έστω ότι η γλώσσα L είναι ένωση κλάσεων μιας σχέσης ισοδυναμίας R που έχει πεπερασμένο δείκτη $t < \infty$. Θα δείξουμε ότι ο δείκτης της R_L είναι το πολύ t, άρα πεπερασμένος.

Δείχνουμε κατ΄ αρχήν τη συνεπαγωγή $x\ R\ y \Rightarrow x\ R_L\ y$: Έστω $z \in \Sigma^*$, $xz \in L$ και $x\ R\ y$. Επειδή R δεξιά αναλλοίωτη έπεται $xz\ R\ yz$. Άρα το yz ανήκει στην ίδια R-κλάση με το xz. Αφού όμως $xz \in L$, και η L είναι ένωση κλάσεων της R, ολόκληρη η R-κλάση του xz περιέχεται στην L, άρα $yz \in L$. Δείξαμε δηλ. ότι $xz \in L \Rightarrow yz \in L$, και ακριβώς το ίδιο προκύπτει και η αντίστροφη συνεπαγωγή, άρα $x\ R_L\ y$, όπως θέλαμε να δείξουμε.

Η συνεπαγωγή που δείξαμε ($x\ R\ y \Rightarrow x\ R_L\ y$) σημαίνει ότι κάθε κλάση της R περιέχεται εξ ολοκλήρου σε *μια* κλάση της R_L. Πράγματι, έστω $x \in K$, όπου K μια R-κλάση, και έστω $x \in S$, όπου S η R_L-κλάση του x. Δείχνουμε τότε ότι $K \subseteq S$. Έστω λοιπόν $y \in K$. Αυτό σημαίνει $x\ R\ y$ άρα και $x\ R_L\ y$, οπότε $y \in S$.

Αφού κάθε R-κλάση περιέχεται σε μια R_L-κλάση λοιπόν δε μπορούν οι R_L-κλάσεις να είναι περισσότερες από τις R-κλάσεις, γιατι τότε θα υπήρχε κάποια R_L-κλάση που δε θα περιείχε καμιά R-κλάση και θα ήταν κενή, πράγμα που εξ ορισμού δε γίνεται.

$3 \Rightarrow 1$
Υποθέτουμε τώρα ότι η R_L έχει πεπερασμένο δείκτη και δείχνουμε ότι η L αναγνωρίζεται από κάποιο DFA M, άρα είναι κανονική. Αν $x \in \Sigma^*$ συμβολίζουμε με $[x]$ την R_L-κλάση ισοδυναμίας της λέξης x.

- Ως σύνολο καταστάσεων του αυτομάτου M παίρνουμε το σύνολο Q όλων των R_L-κλάσεων, που είναι πεπερασμένο από την υπόθεση.
- Ως αρχική κατάσταση παίρνουμε την κλάση $[\epsilon]$ της κενής λέξης.
- Τη συνάρτηση μετάβασης ορίζουμε ως εξής:
$$\delta([x], \alpha) = [x\alpha], \quad (\alpha \in \Sigma). \tag{4.3}$$
- Ως τελικές κορυφές παίρνουμε εκείνες τις κλάσεις της R_L που περιέχονται στην L (δες Άσκηση 51).

Άσκηση 55. Δείξτε ότι ο ορισμός (4.3) είναι 'καλός'. Αυτό σημαίνει ότι αν $K = [x] = [y]$ και εφαρμόσουμε τον (4.3) για να ορίσουμε την κορυφή $\delta(K, \alpha)$ τότε παίρνουμε το ίδιο αποτέλεσμα είτε χρησιμοποιήσουμε το στοιχείο x στον (4.3) είτε το y.

Επαναλαμβάνοντας τον ορισμό (4.3) (ή εφαρμόζοντας επαγωγή στο μήκος της λέξης $w \in \Sigma^*$) βλέπουμε ότι
$$\delta([x], w) = [xw],$$
άρα $\delta([\epsilon], w) = [\epsilon w] = [w]$ που είναι τελική κατάσταση του M αν και μόνο αν $w \in L$, άρα το M αναγνωρίζει ακριβώς τη γλώσσα L.
□

4.3 Ελαχιστοποίηση DFA

Το ερώτημα που μας απασχολεί εδώ είναι πώς, δοθέντος ενός DFA M, να βρούμε ένα άλλο DFA, έστω N, που να είναι *ισοδύναμο* με το M (δηλ. $L(M) = L(N)$ – αναγνωρίζουν την ίδια γλώσσα) και να έχει τον ελάχιστο αριθμό καταστάσεων.

Εξετάζοντας την απόδειξη του θεωρήματος Myhill-Nerode βλέπει κανείς εύκολα ότι υπάρχει ουσιαστικά ένα μοναδικό τέτοιο ελάχιστο DFA, και είναι αυτό που ως σύνολο καταστάσεών του έχει το σύνολο των κλάσεων ισοδυναμίας της δεξιά αναλλοίωτης σχέσης ισοδυναμίας R_L, με $L = L(M)$, και ως συνάρτηση μετάβασης τη συνάρτηση $\delta([x], \alpha) = [x\alpha]$ (δείτε την απόδειξη του Θεωρήματος Myhill-Nerode) όπου $[x]$ συμβολίζει την κλάση της λέξης $x \in \Sigma^*$ και $\alpha \in \Sigma$. Είδαμε επίσης στην απόδειξη του θεωρήματος Myhill-Nerode ότι οι κλάσεις της σχέσης R_M είναι υποσύνολα των κλάσεων της R_L, και αυτό σημαίνει ότι, όποιο και να είναι το DFA M, το ελάχιστο αυτόματο μπορεί να προκύψει από το M αν όλες οι κορυφές του M που είναι R_L ισοδύναμες *συμπτυχθούν* σε μία κορυφή, ώστε πλέον να μην υπάρχουν στο νέο αυτόματο δύο ή περισσότερες κορυφές, που η κλάσεις τους να είναι R_L ισοδύναμες.

Η μέθοδος ελαχιστοποίησης λοιπόν αποφασίζει για το ποιες θα είναι οι καταστάσεις του ελάχιστου DFA αφού βρεί, για κάθε ζεύγος κορυφών του M, αν οι καταστάσεις αυτές είναι μεταξύ τους R_L-ισοδύναμες. Οι καταστάσεις του ελάχιστου αυτομάτου θα είναι τα σύνολα R_L-ισοδυνάμων κορυφών του M. Για να υπολογίσουμε για κάθε ζεύγος κορυφών u και v του M αν είναι R_L-ισοδύναμες θεωρούμε κατ΄ αρχήν όλα τα ζεύγη κορυφών ισοδύναμα και αν προκύψει για ένα ζεύγος κορυφών ότι δεν είναι τότε μόνο τις χωρίζουμε.

Έτσι, θέλουμε να υπολογίσουμε μια συνάρτηση $f(u, v)$ όπου u και v είναι δύο οποιεσδήποτε διαφορετικές κορυφές του M, και θέλουμε η συνάρτηση αυτή να κάνει 1 αν οι u και v ΔΕΝ είναι ισοδύναμες και 0 αν είναι. Κατ΄ αρχήν λοιπόν δίνουμε τις αρχικές τιμές,

$$\forall u, v \in Q, u \neq v, \quad f(u, v) = 0, \tag{4.4}$$

όπου Q το σύνολο των κορυφών του M.

Σύμφωνα με τον ορισμό της σχέσης ισοδυναμίας R_L δύο λέξεις x και y είναι μεταξύ τους R_L-ισοδύναμες αν όποια και να είναι η λέξη z είτε και οι δύο λέξεις xz, yz ανήκουν στην L είτε κι οι δύο δεν ανήκουν. Αμέσως αυτό μας δίνει (με επιλογή $z = \epsilon$) ότι αν $u \in F$ και $v \notin F$ τότε οι u και v δεν είναι ισοδύναμες. Αυτό μας δίνει το επόμενο βήμα του αλγορίθμου μας

$$\forall u \in F, v \notin F \quad f(u, v) = 1. \tag{4.5}$$

Είναι επίσης φανερό από τον ορισμό της R_L ότι αν για τις κορυφές u και v γνωρίζουμε ήδη ότι δεν είναι ισοδύναμες, και για τις δύο κορυφές a και b υπάρχει μια λέξη σ τέτοια ώστε $u = \delta(a, \sigma)$ και $v = \delta(b, \sigma)$ τότε και οι a, b δεν είναι μεταξύ τους R_L-ισοδύναμες. Αυτή η παρατήρηση μας δίνει τον υπόλοιπο αλγόριθμο: Ορίζουμε κατ΄ αρχήν το σύνολο S από ζεύγη κορυφών να περιέχει όλα τα ζεύγη κορυφών (u, v) με $u \in F$, $v \notin F$, δηλ. όλα τα ζεύγη κορυφών για τα οποία γνωρίζουμε ότι τα μέλη τους δεν είναι ισοδύναμα. Έστω τώρα ένα νέο σύνολο από ζεύγη κορυφών T που αποτελείται από όλα τα ζεύγη κορυφών (a, b) για τα οποία ισχύει ακόμη $f(a, b) = 0$ (θεωρούνται δηλ. ακόμη ισοδύναμα) και για τα οποία υπάρχει ένα γράμμα $\sigma \in \Sigma$ τέτοιο ώστε το ζεύγος

$$(\delta(a, \sigma), \delta(b, \sigma)) \in S.$$

Αφού υπολογίσουμε το σύνολο αυτό T θέτουμε μετά $f(a, b) = 1$ για όλα τα $(a, b) \in T$, και τέλος αντιγράφουμε το T πάνω στο παλιό S, το οποίο και πετάμε, και συνεχίζουμε την ίδια διαδικασία (φτιάχνουμε ένα νέο T που προκύπτει από το νέο S, κ.ο.κ.), έως ότου το σύνολο T που φτιάξουμε να προκύψει κενό. Αυτό σηματοδοτεί ότι δεν έχουμε πλέον άλλη πληροφορία να αντλήσουμε και σταματάμε εδώ.

Αποδεικνύεται ότι ο αλγόριθμος αυτός δουλεύει, δηλ. όταν σταματήσει έχουμε $f(u, v) = 1$ ακριβώς για εκείνα τα ζεύγη κορυφών (u, v) με μη R_L-ισοδύναμα μέλη.

Η κατασκευή του ελάχιστου αυτομάτου γίνεται τώρα όπως στην απόδειξη του θεωρήματος Myhill-Nerode: βάζουμε από μιά κορυφή για κάθε ομάδα ισοδυνάμων κορυφών του M. Για να βρούμε που θα πάμε με το γράμμα α από μια τέτοια 'υπερκορυφή' επιλέγουμε μια οποιαδήποτε M-κορυφή της υπερκορυφής αυτής και βλέπουμε που μας πάει το α αν ξεκινήσουμε από αυτή την κορυφή στο παλιό μας αυτόματο M. Η M-κορυφή όπου καταλήγουμε ανήκει σε μια υπερκορυφή του νέου μας υπό κατασκευή αυτομάτου και σε αυτή πρέπει να μεταβούμε. Μια υπερκορυφή θεωρείται τελική αν περιέχει κάποια τελική κορυφή του M (αναγκαστικά τότε θα περιέχει μόνο τελικές κορυφές του F) ενώ η αρχική υπερκορυφή είναι αυτή που περιέχει την αρχική κορυφή του M.

Κεφάλαιο 5

Context free γραμματικές και γλώσσες

5.1 Ένας τρόπος περιγραφής απλών αριθμητικών εκφράσεων

Ας υποθέσουμε ότι θέλουμε να ορίσουμε το τι σημαίνει σωστά δομημένη αριθμητική έκφραση. Για να κάνουμε τα πράγματα πιο απλά (χωρίς να χαθεί η ουσία) ας περιοριστούμε σε αριθμητικές εκφράσεις όπου οι μόνες πράξεις είναι η πρόσθεση (+) και ο πολλαπλασιασμός (*), και όπου ισχύουν οι συνηθισμένοι κανόνες για τις παρενθέσεις. Ας υποθέσουμε επίσης ότι οι βασικές ποσότητες που φτιάχνουν μια έκφραση συμβολίζονται όλες με το γράμμα x. Για παράδειγμα η έκφραση $x * (x + x) + x$ είναι μια σωστή έκφραση ενώ η $xx + *()$ δεν είναι.

Έχουμε δηλαδή ένα πολύ περιορισμένο αλφάβητο

$$\Sigma = \{+, *, (,), x\}$$

και προσπαθούμε να ορίσουμε τη γλώσσα των σωστών εκφράσεων, που είναι ένα κομμάτι του Σ^*.

Ένας γρήγορος και καθαρός τρόπος να τις ορίσουμε είναι να σκεφτούμε το πώς τις φτιάχνουμε: κολλάμε μαζί, με συγκεκριμένους κανόνες, μικρότερες εκφράσεις και φτιάχνουμε μεγαλύτερες. Δίνουμε έτσι τον εξής ορισμό.

Ορισμός 33. Η λέξη x είναι σωστή έκφραση. Επίσης αν οι λέξεις w και v είναι σωστές εκφράσεις, τότε σωστές εκφράσεις είναι επίσης και οι λέξεις (w), $w + v$, $w * v$. Τέλος, σωστές εκφράσεις είναι μόνο οι λέξεις που προκύπτουν από τους άνω κανόνες.

Έτσι η λέξη $x * (x + x) + x$ είναι σωστή έκφραση επειδή οι λέξεις x και $(x + x) + x$ είναι σωστές, και η δεύτερη είναι σωστή επειδή επειδή οι $(x + x)$ και x είναι σωστές και η πρώτη από αυτές είναι σωστή επειδή η $x + x$ είναι σωστή και, τέλος, αυτή είναι σωστή επειδή η x είναι σωστή.

Ορισμοί σαν τον παραπάνω (αλλά και αρκετά πιο περίπλοκοι) κωδικοποιούνται στις λεγόμενες context free γραμματικές. Πριν δώσουμε τον ακριβή ορισμό για αυτές ας πούμε ότι η context free γραμματική που αντιστοιχεί στον παραπάνω ορσιμό είναι η:

1. $S \to x$
2. $S \to (S)$
3. $S \to S + S$
4. $S \to S * S$

Με το σύμβολο S συμβολίζουμε μια 'μεταβλητή' λέξη, η οποία μπορεί να αντικατασταθεί με το δεξί μέλος μιας από τις *παραγωγές* ($S \to \ldots$) από τις οποίες απαρτίζεται η γραμματική. Η λογική είναι ότι ξεκινάμε με τη λέξη S και εφαρμόζουμε σε αυτήν συνεχώς κάποιους από τους κανόνες παραγωγής μέχρι να πάρουμε τη λέξη που θέλουμε. Αν αυτό καταστεί εφικτό τότε, και μόνο τότε, η λέξη αυτή θα είναι στην context free γλώσσα που περιγράφεται από την άνω context free γραμματική.

Με την παρακάτω ακολουθία μετασχηματισμών προκύπτει π.χ. η λέξη $x * (x + x) + x$

$$\begin{array}{lll} S & \to S+S & \text{(κανόνας παραγωγής 3)} \\ & \to S*S+S & \text{(κανόνας παραγωγής 4)} \\ & \to S*(S)+S & \text{(κανόνας παραγωγής 2)} \\ & \to S*(S+S)+S & \text{(κανόνας παραγωγής 3)} \\ & \to x*(x+x)+x & \text{(κανόνας παραγωγής 1, τέσσερεις φορές)} \end{array}$$

Εύκολα βλέπει κανείς ότι με όποιο τρόπο και να χρησιμοποιήσουμε τους κανόνες παραγωγής δε μπορούμε να πάρουμε από την S τη λέξη $xx + *()$.

5.2 Ορισμός context free γραμματικών και των γλωσσών τους

Ορισμός 34. Μια context free γραμματική G πάνω από ένα αλφάβητο Σ (τα *τερματικά σύμβολα*) είναι μια πεπερασμένη συλλογή από

- *μη τερματικά σύμβολα*, που συνήθως τα συμβολίζουμε με κεφαλαία λατινικά γράμματα, και που περιλαμβάνουν το διακεριμένο σύμβολο S (αρχικό μη τερματικό σύμβολο)
- κανόνες παραγωγής $X \to w$, όπου X είναι ένα μη τερματικό σύμβολο και w είναι μια λέξη από γράμματα του Σ και μη τερματικά σύμβολα (η w μπορεί να είναι και η κενή λέξη).

Αν G είναι μια context free γραμματική (CFG) και w, v είναι δύο λέξεις από τερματικά ή μη τερματικά σύμβολα, τότε γράφουμε

$$w \overset{G}{\Rightarrow} v$$

αν με χρήση ενός κανόνα παραγωγής $X \to \alpha$ μπορεί να προκύψει η λέξη v από τη λέξη w. Αυτό σημαίνει ότι αντικαθιστούμε μια εμφάνιση του μη τερματικού συμβόλου X στη λέξη w με τη λέξη α (που απαρτίζεται από τερματικά και μη τερματικά σύμβολα) και με την αντικατάσταση αυτή προκύπτει η λέξη v.

Για παράδειγμα, αν G είναι η γραμματική που ορίσαμε παραπάνω τότε ισχύει

$$x + S \overset{G}{\Rightarrow} x + (S)$$

μια και από την αριστερή λέξη προκύπτει η δεξιά αν χρησιμοποιηθεί ο κανόνας 2.

Ορίζουμε επίσης

$$w \underset{*}{\overset{G}{\Rightarrow}} v$$

να σημαίνει ότι υπάρχει πεπερασμένη ακολουθία λέξεων $v_1, \ldots, v_n$ ώστε

$$w \overset{G}{\Rightarrow} v_1 \overset{G}{\Rightarrow} \cdots \overset{G}{\Rightarrow} v_n \overset{G}{\Rightarrow} v,$$

ότι δηλ. η λέξη v μπορεί να προκύψει από τη λέξη w με επανειλημμένη χρήση των κανόνων παραγωγής της G. Στην παραπάνω γραμματική για τις εκφράσεις ισχύει, για παράδειγμα,

$$S \underset{*}{\overset{G}{\Rightarrow}} (S + S)$$

αφού μπορούμε από τη λέξη S να πάμε στη λέξη $(S + S)$ εφαρμόζοντας πρώτα τον κανόνα 2 και μετά τον κανόνα 3.

Έχοντας στα χέρια μας τον συμβολισμό αυτό μπορούμε εύκολα να ορίσουμε πλέον ποια είναι η *γλώσσα* που αντιστοιχεί σε μια λέξη (από τερματικά ή μη σύμβολα).

Ορισμός 35. Αν G είναι μια CFG και w μια λέξη από τερματικά ή μη τερματικά σύμβολα της G τότε η γλώσσα της w ορίζεται ως

$$L(w) = L_G(w) = \left\{x \in \Sigma^* : \quad w \overset{G}{\underset{*}{\Rightarrow}} x\right\}.$$

Απαρτίζεται δηλ. η $L(w)$ από εκείνες τις λέξεις *χωρίς μη τερματικά σύμβολα* που μπορούν να παραχθούν σε πεπερασμένο πλήθος βημάτων από τη λέξη w με τους κανόνες παραγωγής της G.

Τέλος ορίζουμε *την* γλώσσα της G.

Ορισμός 36. Αν G είναι μια CFG η γλώσσα $L(G)$ ορίζεται να είναι η γλώσσα $L(S)$. Μια γλώσσα L λέγεται *context free γλώσσα* (CFL) αν είναι η γλώσσα κάποιας context free γραμματικής.

Είναι δηλ. η γλώσσα της G όλες οι λέξεις του Σ^* που παράγονται από το αρχικό μη τερματικό σύμβολο S.

Θα χρησιμοποιούμε συνήθως τη συντομογραφία

$$X \to w_1|w_2|\ldots|w_n$$

για να υποδηλώσουμε μια ομάδα από κανόνες παραγωγής

$$X \to w_1,\ X \to w_2,\ \ldots, X \to w_n,$$

με το ίδιο αριστερό μέλος.

Άσκηση 56. Ποια είναι η γλώσσα της γραμματικής με κανόνες $S \to \epsilon \mid 0S0 \mid 1S1$;

Άσκηση 57. Δώστε μια CFG για τη γλώσσα $\{0^n1^n : \quad n = 1, 2, \ldots\}$.

Άσκηση 58. Ανατρέξτε πίσω στον ορισμό του τι είναι κανονική έκφραση. Αν σταθεροποιήσουμε το αλφάβητο σε, ας πούμε, $\Sigma = \{a, b\}$, δώστε μια CFG για τη γλώσσα των κανονικών εκφράσεων πάνω από το Σ.

5.3 Ένα ενδιαφέρον παράδειγμα με πλήρη απόδειξη

Ας δούμε τώρα ένα παράδειγμα μιας CFG με παραπάνω από ένα μη τερματικό σύμβολο. Η γραμματική αυτή θα έχει ως γλώσσα της όλες τις λέξεις του $\{a, b\}^*$ που έχουν ίδιο πλήθος από a και b. Θυμίζουμε εδώ ότι αυτή η γλώσσα δεν είναι κανονική (αποδεικνύεται εύκολα αυτό με χρήση του Λήμματος Άντλησης (Θεώρημα 8).

Η γραμματική G_1 είναι η ακόλουθη:

1. $S \to aB \mid bA \mid \epsilon$
2. $A \to aS \mid bAA$
3. $B \to aBB \mid bS$

Θεώρημα 12. *Η γλώσσα $L(G_1)$ απαρτίζεται από εκείνες τις λέξεις του $\{a, b\}^*$ με ίδιο πλήθος από a και b.*

Απόδειξη: Κατ΄ αρχήν τα μόνα τερματικά σύμβολα που εμφανίζονται στους κανόνες της G_1 είναι τα a και b, άρα η $L(G_1)$ είναι υποσύνολο του $\{a, b\}^*$.

Ας συμβολίζουμε για μια λέξη w του $\{a, b\}^*$ με $A(w)$ και $B(w)$ αντίστοιχα το πλήθος των a και b που περιέχει.

Πρέπει να δείξουμε ότι ισχύει $A(w) = B(w)$ αν και μόνο αν $S \overset{G_1}{\underset{*}{\Rightarrow}} w$. Αλλά αν προσπαθήσουμε να δείξουμε μόνο αυτό θα συναντήσουμε δυσκολίες. Παρ΄ ότι δείχνει αντιφατικό μας διευκολύνει το να δείξουμε παράλληλα και άλλες δύο προτάσεις. Έτσι θα δείξουμε με επαγωγή ως προς το μήκος της λέξης w τις εξής τρείς ισοδυναμίες.

1. $w \in L(S)$ αν και μόνο αν $A(w) = B(w)$.
2. $w \in L(A)$ αν και μόνο αν $A(w) = B(w) + 1$.
3. $w \in L(B)$ αν και μόνο αν $B(w) = A(w) + 1$.

Ο ρόλος δηλ. των μη τερματικών συμβόλων A και B στην G_1 είναι να παράγουν όλες τις λέξεις με ακριβώς ένα a παραπάνω (για το A) και ακριβώς ένα b παραπάνω (για το B).

Όπως είπαμε συμβολίζουμε με n το μήκος της λέξης w και κάνουμε επαγωγή ως προς n. Αν $n = 0$, τότε $w = \epsilon$ και w παράγεται από το S με τον τελευταίο κανόνα παραγωγής του S ενώ δεν παράγεται από τα A ή B, αφού όλοι οι κανόνες παραγωγής των A και B παράγουν μη κενές λέξεις. Βλέπουμε έτσι ότι ισχύουν και οι τρεις ισοδυναμίες για τη λέξη $w = \epsilon$, τη μοναδική λέξη με μήκος $n = 0$.

Υποθέτουμε τώρα επαγωγικά ότι ισχύουν και οι τρεις άνω ισοδυναμίες για κάθε λέξη w με $|w| \leq n - 1$.

Έστω τώρα w μια λέξη με μήκος n. Αποδεικνύουμε την πρώτη ισοδυναμία: $w \in L(S) \iff A(w) = B(w)$.

$\underline{w \in L(S) \Longrightarrow A(w) = B(w)}$
Αφού $w \in L(S)$ υπάρχει μια ακολουθία παραγωγών της G_1 που αρχίζει από το S και καταλήγει στη w. Αν το πρώτο γράμμα της w είναι a τότε στην πρώτη παραγωγή αναγκαστικά χρησιμοποιείται ο κανόνας $S \to aB$. Αυτό συνεπάγεται ότι $w = ax$ όπου x είναι μια λέξη για την οποία ισχύει $x \in L(B)$, και $|x| \leq n - 1$. Από την επαγωγική υπόθεση έπεται ότι $B(x) = A(x) + 1$ και συνεπώς $A(w) = B(w)$. Ομοίως, αν το πρώτο γράμμα της w είναι b τότε ο πρώτος κανόνας παραγωγής από το S στο w είναι αναγκαστικά ο $S \to bA$, άρα $w = by$, με $y \in L(A)$ και $|y| = n-1$. Από την επαγωγική υπόθεση $A(y) = B(y)+1$ από το οποίο προκύπτει $A(w) = B(w)$.

$\underline{A(w) = B(w) \Longrightarrow w \in L(S)}$
Αν το πρώτο γράμμα της w είναι a τότε $w = ax$ με $A(x) = B(x) - 1$, και $|x| = n - 1$. Από την επαγωγική υπόθεση προκύπτει ότι $x \in L(B)$, άρα υπάρχει τρόπος να παράγουμε το x από το μη τερματικό σύμβολο B. Αρχίζοντας τότε από το S, εφαρμόζουμε τον κανόνα παραγωγής $S \to aB$ και ακολούθως παράγουμε από το B το x. Συνολικά έχουμε παραγάγει έτσι το w από το S δείχνοντας σε αυτή την περίπτωση $w \in L(S)$. Ομοίως, αν το πρώτο γράμμα της w είναι το b τότε γράφεται $w = by$ με $|y| = n - 1$ και $A(y) = B(y) + 1$, άρα, με την επαγωγική υπόθεση, ισχύει $y \in L(A)$. Για να παραγάγουμε λοιπόν τη w από το S ξεκινούμε εφαρμόζοντας τον κανόνα $S \to bA$, και ακολούθως παράγουμε από το A το y, παίρνοντας έτσι τελικά το w.

Εδώ έχουμε δείξει το επαγωγικό βήμα για την πρώτη ισοδυναμία.

Δείχνουμε τώρα το επαγωγγικό βήμα για την ισοδυναμία $w \in L(A) \iff A(w) = B(w) + 1$. Προσέξτε ότι εδώ υπάρχει ασυμμετρία στην απόδειξη όσον αφορά το ρόλο που παίζει το a και το b ως πρώτο γράμμα της λέξης w.

$\underline{w \in L(A) \Longrightarrow A(w) = B(w) + 1}$
Έστω ότι το πρώτο γράμμα της λέξης w είναι το a. Τότε αναγκαστικά η πρώτη παραγαγωγή από το A στο w είναι η $A \to aS$, οπότε $w = ax$ με $|x| = n - 1$ και αναγκαστικά τότε το x πρέπει να είναι παραγόμενο από το S, άρα $x \in L(S)$ και από την επαγωγική υπόθεση $A(x) = B(x)$, από το οποίο προκύπτει ότι $A(w) = B(w)+1$.

Αν το πρώτο γράμμα της w είναι το b τότε ο πρώτος κανόνας που εφαρμόζεται στην παραγωγή της w από το A είναι αναγκαστικά ο $A \to bAA$, οπότε έχουμε $w = bx_1x_2$ όπου $x_1, x_2 \in L(A)$, και, φυσικά, $|x_1|, |x_2| \leq n-1$. Από την επαγωγική υπόθεση τώρα προκύπτει ότι $A(x_1) = B(x_1) + 1$ και $A(x_2) = B(x_2) + 1$, από τα οποία προκύπτει φυσικά ότι $A(w) = B(w) + 1$.

$\underline{A(w) = B(w) + 1 \Longrightarrow w \in L(A)}$
Αν το πρώτο γράμμα της w είναι το a τότε $w = ax$ με $A(x) = B(x)$, $|x| = n - 1$, οπότε από την επαγωγική υπόθεση προκύπτει $x \in L(S)$. Παράγοντας τώρα από το A με τον κανόνα παραγωγής $A \to aS$ και συνεχίζοντας παράγοντας από το S το x, καταλήγουμε σε μια ακολουθία παραγωγής του w από το A, δηλ. $w \in L(A)$.

Αν το πρώτο γράμμα της w είναι το b τότε $w = bx$ με $|x| = n - 1$ και $A(x) = B(x) + 2$. Ισχύει όμως το παρακάτω Λήμμα.

Λήμμα 1. *Αν για μια λέξη $x \in \{a, b\}^*$ ισχύει $A(x) = B(x) + 2$ τότε το x σπάει ως $x = x_1x_2$ όπου $A(x_1) = B(x_1) + 1$ και $A(x_2) = B(x_2) + 1$.*

Άσκηση 59. Δείξτε το Λήμμα 1.

Οπότε η λέξη x σπάει σε δυο κομμάτια x_1 και x_2, $x = x_1x_2$, με μήκος το πολύ $n-2$ η καθε μία και με $A(x_i) = B(x_i) + 1$, $i = 1, 2$. Από την επαγωγική υπόθεση έχουμε τώρα $x_1, x_2 \in L(A)$. Για να παραγάγουμε λοιπόν από το A τη λέξη w αρχίζουμε με τον κανόνα παραγωγής $A \to bAA$ και ακολούθως αναπτύσσουμε το πρώτο A σε x_1 και το δεύτερο σε x_2, πετυχαίνοντας έτσι συνολικά μια παραγωγή της w από το A.

Παραλείπουμε την απόδειξη του επαγωγικού βήματος για την τρίτη ισοδυναμία μια και αυτή είναι εντελώς παρόμοια με τη δεύτερη ισοδυναμία, με τους ρόλους των a, b και A, B εναλλαγμένους.
□

5.4 Οι κανονικές γλώσσες είναι και context free

Θεώρημα 13. *Κάθε κανονική γλώσσα είναι και context free.*

Το αντίστροφο φυσικά δεν ισχύει όπως δείχνουν πολλά από τα παραδείγματα που έχουμε δει ως τώρα, π.χ. η γλώσσα $\{0^n1^n : \ n = 1, 2, \ldots\}$.

Απόδειξη. Έστω L κανονική γλώσσα. Αυτό σημαίνει ότι υπάρχει μια κανονική έκφραση r για τη γλώσσα. Αρκεί να δείξουμε ότι κάθε κανονική έκφραση περιγράφει μια context free γλώσσα.

Αρχίζουμε από τις απλούστερες κανονικές εκφράσεις και το δείχνουμε σταδιακά για όλες. Κάνουμε δηλ. επαγωγή ως προς το μήκος της κανονικής έκφρασης. Αν η κανονική έκφραση r έχει μήκος 1, τότε είναι αναγκαστικά ένα γράμμα του $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$, το σύμβολο ϵ ή το σύμβολο $\emptyset$ (ανατρέξτε πίσω στον Ορισμό 27). Αν είναι γράμμα του Σ δηλ. $r = \alpha_j$ για κάποιο $j \in \{1, \ldots, k\}$, τότε η γλώσσα $L(r)$ είναι το μονοσύνολο $\{\alpha_j\}$, το οποίο δίνεται από την CFG $S \to \alpha_j$. Αν $r = \epsilon$ τότε $L(r) = \{\epsilon\}$ που δίνεται από την CFG $S \to \epsilon$, και αν $r = \emptyset$ τότε $L(r) = \emptyset$ και αυτό το σύνολο δίνεται από τη CFG $S \to S$, που προφανώς δεν παράγει τίποτα.

Αν τώρα το μήκος της έκφρασης r είναι μεγαλύτερο του 1 τότε, με βάση τον ορισμό των κανονικών εκφράσεων, η r είναι της μορφής

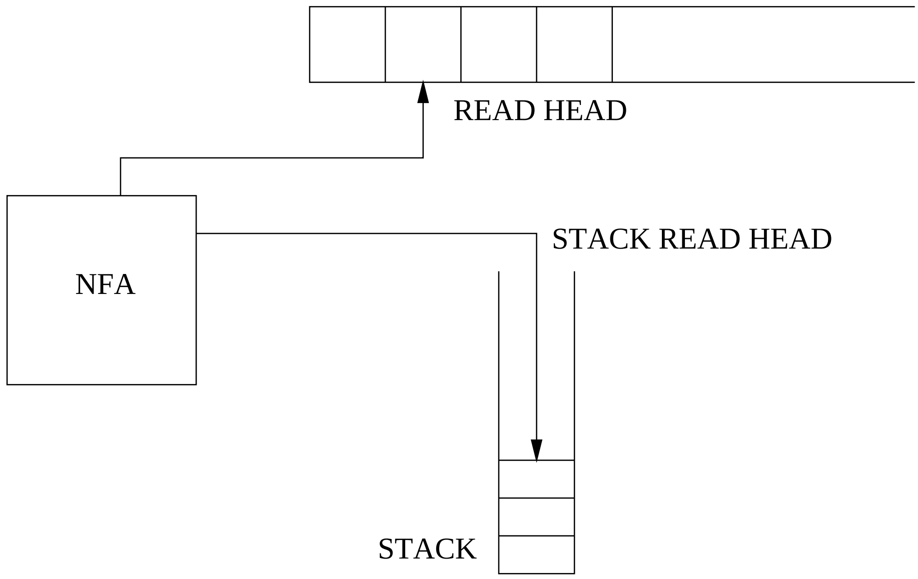
- $r = (st)$, με s, t κανονικές εκφράσεις μικρότερου μήκους. Από την επαγωγική υπόθεση υπάρχουν context free γραμματικές G_1 και G_2 τέτοιες ώστε $L(G_1) = L(s)$ και $L(G_2) = L(t)$. Για να φτιάξουμε μια CFG για τη γλώσσα $L(r) = L(s)L(t)$ μετονομάζουμε κατ΄ αρχήν όλα τα μη τερματικά σύμβολα της G_1 και της G_2 ώστε να είναι διαφορετικά μεταξύ τους και διαφορετικά από S, και ονομάζουμε τα δύο αρχικά μη τερματικά σύμβολα σε S_1 και S_2 για τις G_1 και G_2 αντίστοιχα. Η CFG για τη γλώσσα $L(r)$ έχει ως κανόνες όλους τους κανόνες της G_1 και της G_2 συν τον κανόνα $S \to S_1S_2$, που είναι και ο μόνος κανόνας για το αρχικό μη τερματικό σύμβολο S.
- $r = (s+t)$. Όπως και πριν όλα μόνο που ο επιπλέον κανόνας της γραμματικής δεν είναι τώρα ο $S \to S_1S_2$ αλλά οι δύο κανόνες $S \to S_1 \mid S_2$.
- $r = (s^*)$. Μετονομάζουμε το αρχικό μη τερματικό σύμβολο της CFG για το s σε S_1 (ή κάποιο άλλο όνομα αν αυτό υπάρχει ήδη στη γραμματική) και προσθέτουμε τον κανόνα $S \to \epsilon \mid S_1S$.

□

Άσκηση 60. Υπάρχει και άλλος τρόπος να δείξουμε ότι κάθε κανονική γλώσσα είναι κανονική. Μπορούμε να δουλέψουμε κατ΄ ευθείαν πάνω στο DFA ή NFA που αναγνωρίζει τη γλώσσα μας και να φτιάξουμε μέσω αυτού τη γραμματική. Διαλέξτε ένα από τα DFA ή NFA (η δουλειά είναι ουσιαστικά η ίδια) που εμφανίζονται σε αυτές τις σημειώσεις, π.χ. το αυτόματο που αναγνωρίζει τις λέξεις του $\{0,1\}^*$ με περιττό πλήθος από μηδενικά και περιττό πλήθος από άσσους (Σχήμα 2). Αντιστοιχείστε ένα μη τερματικό σύμβολο σε κάθε κατάσταση του αυτομάτου και βρείτε πώς πρέπει να φτιάξετε τους κανόνες παραγωγής ώστε να προκύψει μια CFG που να περιγράφει την ίδια γλώσσα. Διατυπώστε ένα γενικό τρόπο ώστε να πηγαίνετε από DFA ή NFA σε ισοδύναμη (που να περιγράφει την ίδια γλώσσα) CFG, *χωρίς να περνάτε από την αντίστοιχη κανονική έκφραση.*

5.5 Το αυτόματο με στοίβα (Push Down Automaton)

Όπως ακριβώς οι κανονικές γλώσσες έχουν αντίστοιχες μηχανές, τα DFA, ή τα NFA ή τα ϵ-NFA, που αναγνωρίζουν ακριβώς το σύνολο των κανονικών γλωσσών, έτσι υπάρχει και μια μηχανή, το αυτόματο με στοίβα, ή Push Down Automaton (PDA) που έχει την ιδότητα ότι μια γλώσσα L είναι context free αν και μόνο αν υπάρχει PDA που την αναγνωρίζει. Δε θα δείξουμε αυτή την ισοδυναμία εδώ (όπως είχαμε κάνει για τις κανονικές γλώσσες και τα πεπερασμένα αυτόματα).



Σχήμα 18. Ένα αυτόματο με στοίβα

Ένα PDA (Σχήμα 18) αποτελείται από τρία μέρη

1. Ένα NFA που το βλέπουμε σα 'μονάδα ελέγχου' της μηχανής. Είναι σημαντικό εδώ να τονίσουμε ότι δε μπορούμε εδώ να αντικαταστήσουμε το NFA με DFA – <u>τα αυτόματα με στοίβα είναι μη ντετερμινιστικές μηχανές</u>.
2. Μια ταινία ανάγνωσης (read tape) που διαβάζεται από τον έλεγχο (NFA) με μια κεφαλή ανάγνωσης (read head) η οποία αρχίζει από τα αριστερά της ταινίας ανάγνωσης και κινείται μόνο προς τα δεξιά, και το πολύ κατά ένα σε κάθε βήμα (κύκλο) της μηχανής. Η προς αναγνώριση λέξη πρέπει να τοποθετηθεί εξ αρχής πάνω σε αυτή την ταινία από τον 'χρήστη' της μηχανής.
3. Μια στοίβα (stack). Η στοίβα είναι μια απεριόριστη ποσότητα μνήμης την οποία όμως μπορούμε να διαχειριστούμε με πολύ περιορισμένο τρόπο. Φανταζόμαστε τη στοίβα σα μια (αρχικά κενή) στήλη από θέσεις μνήμης, που αρχίζει από το επίπεδό μας και εκτείνεται απείρως προς τα πάνω. Σε κάθε θέση μνήμης μπορούμε να γράψουμε ένα οποιοδήποτε από τα γράμματα του αλφαβήτου μας ή ένα πεπερασμένο αριθμό από άλλα ειδικά σύμβολα που ενδεχομένως μας διευκολύνουν στον 'προγραμματισμό' του PDA. Στη στήλη αυτή μπορούμε να κάνουμε τις εξής τρεις πράξεις μόνο:
    - Μπορούμε να δούμε τα περιεχόμενα της πρώτης (από πάνω προς τα κάτω) θέσης μνήμης που είναι μη κενή. Αν η στοίβα είναι κενή μπορούμε να το διαπιστώσουμε αυτό.
    - Μπορούμε να προσθέσουμε κάτι στη στοίβα αλλά μόνο ακριβώς στη θέση μνήμης πάνω από την πρώτη (από πάνω προς τα κάτω) γεμάτη θέση μνήμης. Η πράξη αυτή αυξάνει λοιπόν το ύψος της στοίβας κατά 1.
    - Μπορούμε να διαγράψουμε την πιο πάνω θέση μνήμης (αν η στοίβα είναι κενή αυτή η πράξη δεν έχει κανένα αποτέλεσμα). Η πράξη αυτή ελαττώνει το ύψος της στοίβας κατά ένα αν η στοίβα δεν ήταν κενή.

Το NFA του αυτομάτου διαβάζει πάλι τα γράμματα της προς αναγνώριση λέξης ένα προς ένα, από αριστερά προς τα δεξιά, χωρίς και πάλι τη δυνατότητα να γυρίσει πίσω προς τα πίσω και να ξαναδιαβάσει την αρχή της λέξης. Και πάλι σε κάθε βήμα το NFA έχει τη δυνατότητα να εκτελέσει μια κίνηση ανάμεσα σε διάφορες δυνατές κινήσεις. Μια λέξη αναγνωρίζεται από το PDA αν κάποια επιλογή κινήσεων του NFA μπορεί να οδηγήσει σε αποδοχή της λέξης (το οποίο συμβαίνει αν στο τέλος το NFA είναι σε τελική κατάσταση).

Η σύνδεση του NFA με τη στοίβα γίνεται ως εξής: σε κάθε μετάβαση το NFA δεν επιλέγει το σύνολο των δυνατών επομένων καταστάσεων με κριτήριο μόνο το σε ποια κατάσταση βρίσκεται αυτό αλλά κοιτώντας ταυτόχρονα και ποιο είναι το περιεχόμενο της πάνω θέσης μνήμης της στοίβας (η μόνη θέση μνήμης που μπορεί άλλωστε να δει). Επίσης σε κάθε μετάβαση το NFA εκτελεί ενδεχομένως και μια από τις τρεις επιτρεπτές πράξεις στη στοίβα (διαβάζει την κορυφή της, προσθέτει κάτι στην κορυφή της, πετάει την πάνω θέση μνήμης χαμηλώνοντας την κορυφή της στοίβας κατά ένα, ή δεν κάνει τίποτα στη στοίβα).

Το μοντέλο του PDA είναι λειτουργικά ισοδύναμο με προγραμματισμό σε μια συνηθισμένη γλώσσα προγραμματισμού (π.χ. στην C) με τους εξής περιορισμούς και προσθήκες.

- Η μνήμη που χρησιμοποιεί το πρόγραμμά μας πρέπει να είναι συνολικά πεπερασμένη και γνωστή εξ' αρχής, ανεξάρτητα από το ποιο θα είναι το input (η προς αναγνώριση λέξη). Στην πράξη, αυτό το εξασφαλίζουμε δηλώνοντας μερικές μεταβλητές ακέραιου τύπου μόνο (int στη C) και λαμβάνοντας υπόψιν ότι κάθε τέτοια μεταβλητή δε μπορεί να μεγαλώσει απεριόριστα, έχει δηλ. συγκεκριμένο αριθμό από bits (δυαδικά ψηφία) που αντιστοιχούν σε αυτή.
- Η πρόσβαση στο input γίνεται μέσω της συνάρτησης `int INP();` η οποία κάθε φορά που καλείται επιστρέφει και το επόμενο γράμμα της λέξης που εξετάζουμε. Όταν έχει διαβάσει όλα τα γράμματα επιστρέφει από κει και πέρα την ειδική τιμή -1.
- Υπάρχουν ακόμη άλλες τρεις 'συναρτήσεις βιβλιοθήκης' με τις οποίες γίνεται ο χειρισμός της στοίβας, και μια συνάρτηση ακόμη με την οποία μπαίνει μέσα στη γλώσσα ο μη ντετερμινισμός που είναι απαραίτητος στο PDA. Οι συναρτήσεις για τη στοίβα είναι οι εξής:

`int READ();`	Επιστρέφει τα περιεχόμενα της πάνω θέσης μνήμης της στοίβας ή -1 αν η στοίβα είναι άδεια.
`POP();`	αδειάζει μια θέση μνήμης από τη στοίβα ή δεν κάνει τίποτα αν η στοίβα είναι ήδη άδεια
`PUSH(int x);`	Γράφει τον αριθμό x στην κορυφή της στοίβας, σε μια νέα θέση μνήμης

- Η συνάρτηση `NF` μέσω της οποίας κωδικοποιείται η μη ντετερμιστική συμπεριφορά περιγράφεται στην επόμενη παράγραφο.

Μη ντετερμινιστικά 'προγράμματα'

Γενικά ο μηχανισμός του μη ντετερμινισμού χρησιμοποιείται σε προβλήματα αποφάσεων, σε ερωτήματα δηλαδή που φιλοδοξούμε να λύσουμε υπολογιστικά και τα οποία επιδέχονται ΝΑΙ ή ΟΧΙ απάντηση, όπως ακριβώς είναι και το πρόβλημα που μας ενδιαφέρει εδώ, να αποφασίζουμε δηλ. αν μια λέξη που μας δίνεται ανήκει σε μια γλώσσα L ή όχι.

Η συνάρτηση μέσω της οποίας 'υλοποιείται' ο μη ντετερμινισμός είναι η `int NF(int x);` η οποία επιστρέφει *μη ντετερμινιστικά* μια από τις επιλογές $0, 1, \ldots, x-1$. Το νόημα της συνάρτησης αυτής είναι ότι εμείς δεν έχουμε κανένα έλεγχο στο ποια από τις δυνατές επιλογές αυτή επιλέγει, αλλά πρέπει να γράψουμε το πρόγραμμά μας με τέτοιο τρόπο ώστε *αν αυτή η συνάρτηση επιστρέψει κατά τις κλήσεις της τις κατάλληλες επιλογές* τότε να κάνουμε τη λέξη δεκτή, αν αυτή είναι μέσα στη γλώσσα. Αλλά, δεν πρέπει να σε καμία περίπτωση να κάνουμε δεκτή μια λέξη που δεν ανήκει στη γλώσσα όποιες τιμές κι αν επιστρέψει η `NF`.

Για να γίνει κατανοητός ο τρόπος χρήσης της `NF` δείχνουμε παρακάτω μια συνάρτηση σε C η οποία παίρνει δύο ορίσματα `number` και `vector` και επιστρέφει 1 αν ο ακέραιος `number` περιέχεται στον πίνακα `vector` (ο οποίος έχει 1000 στοιχεία, τα `vector[0], ... , vector[999]`) και 0 αν δεν περιέχεται.

```
int NumberInVector(int number, int vector[1000])
{   int location;
    location = NF(1000);
```

```
    if(vector[location]==number) return 1;
    return 0;
}
```

Πρέπει εδώ να ξεκαθαρίσουμε ότι η συνάρτηση `NumberInVector` λύνει το πρόβλημα που θέλουμε υπό την εξής έννοια:

- Αν ο αριθμός `number` είναι μέσα στον πίνακα `vector` τότε υπάρχει τρόπος να απαντήσει η `NF` ώστε ο αλγόριθμός μας να δώσει τη σωστή απάντηση, δηλ. 1.
- Αν ο αριθμός `number` δεν είναι μέσα στον πίνακα `vector` τότε, ό,τι και να απαντήσει η `NF`, ο αλγόριθμός μας δεν υπάρχει περίπτωση να κάνει λάθος (αφού ελέγχει αν όντως βρίσκεται ο `number` στη θέση `vector[location]` πριν απαντήσει) και απαντάει πάντα 0.

Αν δεν είχαμε στη διάθεσή μας τη συνάρτηση `NF` θα είμασταν αναγκασμένοι, λίγο-πολύ, να κάνουμε χίλιους ελέγχους για να διαπιστώσουμε αν ο δεδομένος αριθμός είναι μέσα στον πίνακα ή όχι. Παρατηρείστε ότι τώρα δεν υπάρχει ανακύκλωση κανενός είδους μέσα στη συνάρτηση και ότι η απάντηση βρίσκεται σε σταθερό χρόνο! Ο αλγόριθμος της συνάρτησης `NumberInVector` 'μαντεύει' τη θέση στην οποία βρίσκεται στο `vector` ο αριθμός και απλά ελέγχει μετά αν είναι όντως έτσι πριν απαντήσει 1 ή 0, ώστε να αποφύγει να κάνει λάθος στην περίπτωση που ο αριθμός δεν είναι μέσα. Ο αλγόριθμος αυτός δηλ. δεν υπάρχει περίπτωση να απαντήσει 1 ενώ η απάντηση είναι 0, αλλά μπορεί κάλλιστα να απαντήσει 0 όταν η απάντηση είναι 1. Αυτή η ασυμμετρία είναι πολύ χαρακτηριστική στους μη ντετερμινιστικούς αλγορίθμους.

Πρέπει να είναι φανερό με αυτό το παράδειγμα, ότι μη ντετερμινιστικοί αλγόριθμοι δεν μπορούν να υλοποιηθούν. Είναι απλά ένα χρήσιμο θεωρητικό κατασκεύασμα.

5.6 Παραδείγματα PDA

Η γλώσσα $L_1 = \{0^n 1^n : \quad n = 1, 2, \ldots\}$

Η γλώσσα $L_1 = \{0^n 1^n : \quad n = 1, 2, \ldots\}$ είναι context free (η γραμματική είναι: $S \to \epsilon \mid 0S1$). Θα δώσουμε εδώ ένα PDA για τη γλώσσα αυτή. Ισοδύναμα, θα περιγράψουμε το PDA αυτό σα μια συνάρτηση σε C, σύμφωνα με αυτά που είπαμε στην προηγούμενη παράγραφο. Η συνάρτηση αυτή θα μπορεί να χρησιμοποιεί σταθερή σε μέγεθος μνήμη και θα έχει πρόσβαση στη στοίβα μέσω των συναρτήσεων `READ, POP, PUSH` που ορίσαμε στην προηγούμενη παράγραφο. Τυγχάνει για τη γλώσσα αυτή να μη χρειάζεται ο μη ντετερμινισμός οπότε δε θα χρησιμοποιήσουμε τη συνάρτηση `NF`.

Το πρόγραμμα-PDA φαίνεται παρακάτω. Η συνάρτηση `f` επιστρέφει 1 αν η λέξη (που τη διαβάζουμε με διαδοχικές κλήσεις στη συνάρτηση `INP()`) ανήκει στην L_1 και 0 αλλιώς. Η στρατηγική της συνάρτησης είναι η εξής. Όσο διαβάζουμε μηδενικά τα σπρώχνουμε πάνω στη στοίβα. Για κάθε άσσο που διαβάζουμε πετάμε κάτι από την στοίβα. Επίσης προσέχουμε ποτέ να μη διαβάσουμε 0 μετά που έχουμε διαβάσει κάποιο 1. Αν στο τέλος η στοίβα καταλήξει άδεια δεχόμαστε τη λέξη.

Ότι ακολουθεί το σύμβολο // αποτελεί σχόλιο και δεν είναι τμήμα του προγράμματος.

```
int f()
{
    int i, SeenOne=0;
    a: i = INP();
    if(i==0) {  //Εδώ διαβάζουμε το επόμενο γράμμα
     if(SeenOne) return 0; //0 μετά από 1 απορρίπτεται
     PUSH(0); goto a; //Μηδενικά σπρώχνονται στη στοίβα
    }
    if(i==1) {
     SeenOne=1;
```

```
    if(-1 == READ()) return 0; //΄Αδειασε η στοίβα πριν την ώρα της
    POP(); goto a; //Για κάθε 1 πετάμε ένα 0 από τη στοίβα
   }
   if(-1 == READ()) return 1; //Αν έχει τελειώσει η λέξη και έχει αδειάσει η στοίβα, δεχόμαστε
   else return 0; //αλλιώς απορρίπτουμε
}
```

Παρατηρείστε ότι, πέρα από τη στοίβα, το πρόγραμμα αυτό χρησιμοποιεί πεπερασμένη και προκαθορισμένη μνήμη. Για την ακρίβεια χρησιμοποιεί όλες κι όλες δύο μεταβλητές, τις `i` (που χρησιμοποιείται για να κρατάει το επόμενο γράμμα ή -1) και `SeenOne` (που φροντίζουμε να είναι 0 όσο δεν έχουμε ακόμη δει άσσο και μετά να είναι 1). Η μεταβλητή `i` παίρνει τρεις διαφορετικές τιμές, αρκούν δηλ. 2 λογικά bits για να την αποθηκεύσουμε (μια και με αυτά κωδικοποιούμε 4 διαφορετικές τιμές), ενώ για τη μεταβλητή `SeenOne` που παίρνει δύο τιμές αρκεί ένα λογικό bit.

Η γλώσσα $L_2 = \left\{xx^R :\ \ x \in \{0,1\}^*\right\}$

Η γλώσσα $L_2 = \left\{xx^R :\ \ x \in \{0,1\}^*\right\}$ (x^R είναι η λέξη x γραμμένη ανάποδα) εύκολα φαίνεται ότι είναι context free. Μια CFG γι΄ αυτήν είναι απλούστατα η

$$S \to \epsilon \mid 0S0 \mid 1S1.$$

Παρακάτω δείχνουμε μια συνάρτηση `g` η οποία υλοποιεί ένα PDA για την αναγνώριση της γλώσσας L_2. Εδώ χρησιμοποιείται η συνάρτηση `NF`, πρόκειται δηλ. για ένα μη ντετερμινιστικό πρόγραμμα. Η στρατηγική είναι η εξής. Μέχρι τη μέση της ανάγνωσης της λέξης σπρώχνουμε τα γράμματα όπως τα διαβάζουμε πάνω στη στοίβα. Από τη μέση και πέρα για κάθε γράμμα που διαβάζουμε από το input πετάμε και ένα γράμμα από τη στοίβα και το συγκρίνουμε με αυτό που διαβάσαμε από το input. (Προσέξτε ότι τα γράμματα θα πεταχτούν από τη στοίβα με αντίστροφη σειρά από αυτή με την οποία γράφτηκαν.) Αν είναι ίδια συνεχίζουμε (μέχρι να τελειώσει το input) αλλιώς απορρίπτουμε τη λέξη.

Η σημαντική παρατήρηση εδώ είναι ότι δεν μπορούμε να ξέρουμε πότε έχουμε φτάσει τη μέση της λέξης εισόδου! Αυτή τη βοήθεια αναλαμβάνει να μας δώσει η συνάρτηση `NF()`, η οποία μας λέει ακριβώς αυτό. Στο παρακάτω πρόγραμμα κάθε κλήση στην `NF()` κατά τη διάρκεια της εκτέλεσης του προγράμματος ισοδυναμεί με μια ερώτηση (στο Θεό, αν προτιμάτε) για το αν φτάσαμε τη μέση της λέξης ή όχι. Αν ο Θεός μας βοηθήσει με τη σωστή απάντηση θα δεχτούμε τη λέξη, μόνο όμως αν πρέπει να τη δεχτούμε. Δε μπορεί δηλ. να μας κοροϊδέψει με τρόπο ώστε να δεχτούμε μια λέξη ενώ δε θα έπρεπε (επαναλαμβάνουμε ότι είναι πολύ σημαντική αυτή η ασυμμετρία στα μη ντετερμινιστικά προγράμματα).

```
int g()
{
   int i, r, mid=0;
   a:  i = INP(); //Εδώ διαβάζουμε το επόμενο γράμμα
   if(mid == 0) mid = NF(2); //΄Εχουμε περάσει τη μέση της λέξης;
   if(mid == 0) { //΄Οχι, δε την περάσαμε ακόμη
    PUSH(i); goto a;
   }
   else { //΄Εχουμε περάσει τη μέση
    r = READ(); //Διάβασε την κορυφή της στοίβας
    if(r == -1 && i != -1) return 0; //΄Αδεια στοίβα, απόρριψη
    POP();
    if(r != i) return 0; //Δεν ταιριάζουν, απόρριψη
    goto a; //Ταιριάζουν, συνέχισε
   }
   if(-1 == READ()) return 1; //Κανένα πρόβλημα και στοίβα άδεια, άρα δεχόμαστε
}
```

Η γραμμή του παραπάνω προγράμματος όπου χρησιμοποιείται ο μη ντετερμινισμός είναι η `if(mid == 0) mid = NF(2);`, όπου (αν η μεταβλητή `mid` δεν έχει ήδη γίνει μια φορά 1, οπότε και μένει για πάντα από κει και πέρα) θέτουμε τη μεταβλητή `mid` σε 0 ή 1 με μια κλήση στην `NF(2)`. Όταν η `NF` επιστρέψει 1 θεωρούμε από κει και πέρα ότι έχουμε περάσει τη μέση, και προσπαθούμε με το πρόγραμμα να δούμε αν, με αυτή την υπόθεση μπορούμε να δεχτούμε τη λέξη.

Το πρόγραμμα αυτό χρησιμοποιεί τρεις μεταβλητές κάθε μια από τις οποίες παίρνει τιμές μέσα στο σύνολο $\{0, 1, -1\}$, οπότε δύο λογικά bits αρκούν για την αποθήκευση κάθε μιας από αυτές.

5.7 Το Λήμμα Άντλησης για context free γλώσσες, και η εφαρμογή του

Όπως και στην περίπτωση των κανονικών γλωσσών υπάρχει και για τις context free γλώσσες, ένα 'Λήμμα Άντλησης' που είναι πολύ χρήσιμο στο να αποδεικνύουμε ότι ορισμένες γλώσσες δεν είναι context free. Η μορφή του είναι πολύ παρόμοια με το Λήμμα Άντλησης για κανονικές γλώσσες κα ο τρόπος χρήσης του επίσης.

Θεώρημα 14. (Λήμμα Άντλησης για context free γλώσσες)
Έστω ότι η γλώσσα L είναι context free. Τότε υπάρχει ένας φυσικός αριθμός n τέτοιος ώστε για κάθε λέξη $z \in L$ με $|z| \geq n$ να μπορούμε να γράψουμε

$$z = uvwxy,$$

όπου

(α) $|vx| \geq 1$,

(β) $|vwx| \leq n$, *και*

(γ) για κάθε $i \geq 0$ *ισχύει* $uv^iwx^iy \in L$.

Δε θα δώσουμε απόδειξη του θεωρήματος αυτού, αλλά θα δούμε πώς χρησιμοποιείται για να δείξουμε ότι μια γλώσσα δεν είναι context free.

Η γλώσσα $L_1 = \{a^nb^nc^n : \quad n \geq 0\}$ δεν είναι context free

Ας υποθέσουμε ότι η L_1 είναι context free. Έστω τότε n ο αριθμός που αναφέρεται στο Θεώρημα 14 και $z = a^{2n}b^{2n}c^{2n} \in L_1$. Από το Θεώρημα 14 η λέξη z γράφεται ως $z = uvwxy$ όπου ισχύουν τα συμπεράσματα του Θεωρήματος. Παρατηρούμε ότι η λέξη vwx, που σύμφωνα με το Θεώρημα έχει μήκος το πολύ n, δε μπορεί να περιέχει και a και c, ακριβώς επειδή το μήκος της δε φτάνει να 'γεφυρώσει' τα b της λέξης z. Χωρίς βλάβη της γενικότητας υποθέτουμε ότι από τη λέξη αυτή λείπουν τα a (και το επιχείρημα είναι εντελώς παρόμοιο αν λείπουν τα c). Από το Θεώρημα 14 έπεται ότι η λέξη $uwy \in L_1$ (εφαρμόσαμε το Θεώρημα με $i = 0$). Αλλά για να πάμε από τη λέξη z στην uwy σβήσαμε το u και το x, τα οποία δεν έχουν a μέσα, άρα το πλήθος των a στη λέξη uwy έχει παραμείνει $2n$. Το μήκος της uwy όμως είναι αυστηρά μικρότερο του $6n$ (εδώ χρησιμοποιούμε το (α) από τα συμπεράσματα του Θεωρήματος), άρα δε μπορεί η λέξη uwy να ανήκει στην L_1, όπως είχαμε υποθέσει. Άρα η L_1 δεν είναι context free.

Άσκηση 61. Δείξτε ότι η γλώσσα $\left\{xx : \quad x \in \{0,1\}^*\right\}$ δεν είναι context free.

Άσκηση 62. Δείξτε ότι η γλώσσα $\left\{(ab)^k(cd)^k : \quad k \in \mathbf{N}\right\}$ δεν είναι κανονική.

Κεφάλαιο 6

Υπολογισιμότητα

6.1 Υπολογίσιμες συναρτήσεις και αναδρομικά σύνολα

Μέχρι στιγμής έχουμε δει ουσιαστικά δύο κατηγορίες γλωσσών: τις κανονικές (regular) γλώσσες και τις context free γλώσσες. Η πρώτη κατηγορία περιέχεται στη δεύτερη. Από τη μέχρι στιγμής παρουσίαση πρέπει να έχει φανεί καθαρά ότι για κάθε μια από τις δύο κατηγορίες υπάρχει και ένα αντίστοιχο υπολογιστικό μοντέλο: τα πεπερασμένα αυτόματα (ντετερμινιστικά ή μη) για τις κανονικές γλώσσες και τα αυτόματα με στοίβα (push-down automata) για τις context free γλωσσες. Η αντιστοιχία στην οποία αναφερθήκαμε είναι ότι μια γλώσσα είναι στην κατηγορία που εξετάζουμε (κανονική ή context-free) αν και μόνο αν υπάρχει μια *μηχανή* του αντίστοιχου τύπου (πεπερασμένο αυτόματο ή αυτόματο με στοίβα) που αναγνωρίζει τη γλώσσα αυτή.

Είδαμε ότι το να υπάρχει ένα DFA για μια γλώσσα L είναι ισοδύναμο με το μπορεί κανείς να γράψει ένα πρόγραμμα σε μια γλώσσα προγραμματισμού (ας πούμε σε C) που να αναγνωρίζει τις λέξεις της L (να επιστρέφει δηλ. το πρόγραμμα αυτό 1 αν η λέξη που του δώσαμε ανήκει στην L και 0 αλλιώς), το οποίο δε χρησιμοποιεί απεριόριστη μνήμη αλλά μνήμη μεγέθους σταθερού και προκαθορισμένου, το ίδιο για όλες τις λέξεις της L. Ισοδύναμα, μιλάμε για ένα πρόγραμμα σε C που χρησιμοποιεί ένα στεθερό αριθμό μεταβλητών (όχι πίνακες) η καθεμία από τις οποίες χρησιμοποιεί προκαθορισμένη μνήμη (π.χ. ακέραιοι των 32 bits – δυαδικών ψηφίων).

Αντίστοιχα, ένα *ντετερμινιστικό* αυτόματο με στοίβα είναι ακριβώς ένα πρόγραμμα σε C που χρησιμοποιεί μεν πεπερασμένη μνήμη μόνο, αλλά έχει και πρόσβαση σε μια άπειρη στοίβα με ένα περιορισμένο όμως τρόπο, που του επιτρέπει να διαβάζει και να τροποποιεί πάντα μόνο την κορυφή της στοίβας. Τα *ντετερμινιστικά* αυτόματα με στοίβα δεν αρκούν για να αναγνωρίσουν όλες τις context free γλώσσες όμως, αλλά μπορεί κανείς με λίγη προσοχή να δείξει ότι και τα *μη ντετερμιστικά* αυτόματα με στοίβα έχουν ισοδύναμα C προγράμματα.

Είναι πλέον φυσιολογικός ο ακόλουθος ορισμός.

Ορισμός 37. Έστω Σ ένα πεπερασμένο αλφάβητο. Μια συνάρτηση $f : \Sigma^* \to \Sigma^*$ λέγεται *υπολογίσιμη συνάρτηση* αν υπάρχει ένα C πρόγραμμα P που με είσοδο $x \in \Sigma^*$ επιστρέφει τη λέξη $f(x)$.

Τονίζουμε εδώ ότι δε μας ενδιαφέρει πόσο χρόνο θα χρειαστεί να *τρέξει* αυτό το πρόγραμμα για να υπολογίσει την απάντησή του, αλλά απαιτούμε πάντα από αυτό να μας βρεί τη σωστή την απάντηση.

Ας παρατηρήσουμε εδώ ότι δεν υπάρχει κάτι μαγικό που να μας επιβάλλει το Σ^* σα πεδίο ορισμού και πεδίο τιμών στον παραπάνω ορισμό. Στη θέση του θα μπορούσε να είναι οποιοδήποτε σύνολο που τα στοιχεία του να μπορούν να παρασταθούν, με κάποιο τρόπο, στον υπολογιστή (ένα τέτοιο σύνολο *δεν* είναι το σύνολο $\mathbf{R}$ των πραγματικών αριθμών). Αυτό όμως σημαίνει ουσιαστικά ότι μπορούμε να γράψουμε κάτω μια λέξη από 0 ή 1 που να αντστοιχεί μοναδικά σε ένα στοιχείο του συνόλου αυτού. Πολλές φορές βλέπει κανείς τον ορισμό της υπολογίσιμης συνάρτησης να δίνεται π.χ. για συναρτήσεις από τους φυσικούς αριθμούς στους φυσικούς.

Εμείς με τον όρο υπολογίσιμη συνάρτηση θα καταλαβαίνουμε οποιαδήποτε απεικόνιση την οποία μπορούμε να υπολογίζουμε χρησιμοποιώντας ένα πρόγραμμα.

Για παράδειγμα οι παρακάτω συναρτήσεις είναι υπολογίσιμες αφού εύκολα βλέπει κανείς ότι υπάρχουν προγράμματα που τις υπολογίζουν.

$$\begin{aligned} f_1(x) &= x+1 \quad (\mathbf{N} \to \mathbf{N}) \\ f_2(x) &= x^2 \quad (\mathbf{N} \to \mathbf{N}) \\ f_3(x) &= |x| \quad (\Sigma^* \to \mathbf{N} \end{aligned}$$

Ορισμός 38. Έστω Σ ένα πεπερασμένο αλφάβητο. Ένα σύνολο $L \subseteq \Sigma^*$ λέγεται *αναδρομικό* αν η χαρακτηριστική του συνάρτηση

$$\chi_L(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$$

είναι υπολογίσιμη.

Και πάλι τη θέση του Σ^* μπορεί να πάρει οποιοδήποτε σύνολο του οποίου τα στοιχεία μπορούν να παρασταθούν στον υπολογιστή, π.χ. το σύνολο $\mathbf{N}$ των φυσικών αριθμών. Ένα σύνολο λοιπόν λέγεται αναδρομικό αν μπορούμε να γράψουμε ένα πρόγραμμα που να αποφασίζει πότε ένα στοιχείο x ανήκει στο σύνολο ή όχι. Για παράδειγμα, το σύνολο των πρώτων αριθμών (εκείνοι οι φυσικοί αριθμοί που δεν έχουν κανένα φυσικό αριθμό ως διαιρέτη, εκτός από τον εαυτό τους και τη μονάδα) είναι ένα αναδρομικό σύνολο, αφού μπορούμε να γράψουμε ένα πρόγραμμα πουα να αποφασίζει αν ένας δοσμένος φυσικός αριθμός είναι πρώτος ή όχι, εξετάζοντας κάθε μικρότερό του φυσικό αριθμό για το αν διαιρεί το δοσμένο φυσικό ή όχι. Αυτό σίγουρα δεν είναι το 'καλύτερο' πρόγραμμα που μπορεί κανείς να γράψει για αυτό το σκοπό, αλλά πάντως δουλεύει.

Είναι τώρα φανερό ότι οι κανονικές και οι context free γλώσσες είναι αναδρομικά σύνολα, αφού υπάρχουν προγράμματα που τις 'αποφασίζουν'. Η ιεραρχία των γλωσσών που έχουμε μέχρι τώρα δεί λοιπόν είναι η

$$(\text{κανονικές γλώσσες}) \subset (\text{context free γλώσσες}) \subset (\text{αναδρομικές γλώσσες}).$$

Και οι δύο παραπάνω εγκλεισμοί είναι γνήσιοι, υπάρχει δηλ. context free γλώσσα που δεν είναι κανονική (π.χ. η γλώσσα των παλίνδρομων λέξεων xx^R πάνω από ένα αλφάβητο) και αναδρομική γλώσσα που δεν είναι context free. Ένα παράδειγμα για το τελευταίο αποτελεί η γλώσσα $\{a^n b^n c^n : n \in \mathbf{N}\}$. Γι' αυτήν έχουμε δει στην §5.7 ότι δεν είναι context free γλώσσα, ενώ είναι αρκετά απλό να γράψει κανείς ένα C πρόγραμμα που να αναγνωρίζει πότε μια λέξη είναι της μορφής $a^n b^n c^n$. Ένα τέτοιο πρόγραμμα απλά μετράει το πλήθος των a, b και c και ελέγχει ότι είναι ίδια, καθώς και ότι όλα τα b έρχονται μετά από όλα τα a κι όλα τα c μετά τα b.

Είναι σημαντικό να τονίσουμε εδώ ότι η μεταβλητή του C προγράμματος όπου κρατιέται, π.χ., το πλήθος των a, είναι μια μεταβλητή για την οποία χρειαζόμαστε απεριόριστη μνήμη. Δεν είναι δηλ. δυνατόν να προκαθορίσουμε ένα άνω όριο για το πλήθος αυτό. Και επειδή ο φυσικός αριθμός x χρειάζεται περίπου $\log_2 x$ δυαδικά ψηφία για να παρασταθεί στον υπολογιστή, κι επειδή η συνάρτηση $\log_2 x$ αυξάνει (αν και αργά) χωρίς όριο όταν $x \to \infty$, αυτό το C πρόγραμμα δεν χρησιμοποιεί σταθερή μνήμη (αν αυτό συνέβαινε τότε η γλώσσα αυτή θα ήταν κανονική, ενώ δεν είναι καν context free).

Άσκηση 63. Αν τα σύνολα $A, B \subseteq \mathbf{N}$ είναι αναδρομικά δείξτε ότι το ίδιο ισχύει και για τα σύνολα $A \cap B, A \cup B, A^c$.

6.2 Μη υπολογίσιμες συναρτήσεις

Είναι πολύ φυσιολογικό το ερώτημα αν υπάρχουν συναρτήσεις που να μην είναι υπολογίσιμες. Για να συγκεκριμενοποιήσουμε τα πράγματα, ένα φυσιολογικό ερώτημα είναι αν υπάρχει συνάρτηση $f : \mathbf{N} \to \mathbf{N}$ για την οποία να μη μπορούμε να γράψουμε ένα πρόγραμμα που να την υπολογίζει.

Θεώρημα 15. *Υπάρχει συνάρτηση $f : \mathbf{N} \to \mathbf{N}$ για την οποία δεν υπάρχει πρόγραμμα που να την υπολογίζει.*

Απόδειξη: Η πρώτη παρατήρηση που κάνουμε είναι ότι το σύνολο των υπολογισίμων συναρτήσεων είναι αριθμήσιμο, μπορεί δηλ. κάποιος να γράψει κάτω όλες τις υπολογίσιμες συναρτήσεις $\mathbf{N} \to \mathbf{N}$ σε μια ακολουθία

$$f_1, f_2, \ldots, f_n, \ldots$$

(δεν υποθέτουμε εδώ ότι $f_i \neq f_j$ για $i \neq j$, αλλά επιτρέπουμε τις επαναλήψεις στην λίστα αυτή όλων των υπολογισίμων συναρτήσεων). Αυτό είναι άμεση συνέπεια του ότι το σύνολο όλων των προγραμμάτων είναι αριθμήσιμο. Πράγματι, αν το υποθέσουμε αυτό ως γνωστό δεν έχουμε παρά να απαριθμήσουμε όλα τα προγράμματα και για κάθε ένα από αυτά γράφουμε κάτω ποια συνάρτηση υπολογίζει (αν υπολογίζει συνάρτηση, αλλιώς δε γράφουμε τίποτα). Κάθε υπολογίσιμη συνάρτηση θα εμφανιστεί τότε στη λίστα μια και κάποια στιγμή θα εξετάσουμε ένα από τα προγράμματα που την υπολογίζουν.

Γιατί είναι τώρα το σύνολο όλων των προγραμμάτων αριθμήσιμο; Απλούστατα, κάθε πρόγραμμα δεν είναι παρά μια, συνήθως μεγάλη, λέξη πάνω από ένα συγκεκριμένο αλφάβητο Σ. Για παράδειγμα, όλα τα προγράμματα σε C γράφονται στο αλφάβητο που χρησιμοποιούν οι σύγχρονοι υπολογιστές και που περιλαμβάνει όλα τα λατινικά γράμματα, μικρά και κεφαλαία, δίαφορα σημεία στίξης, αριθμητικούς και άλλους τελεστές, ψηφία, κλπ. Απαριθμούμε πρώτα λοιπόν όλες τις λέξεις μήκους 1, μετά όλες τις λέξεις μήκους 2, όλες τις λέξεις μήκους 3, κ.ο.κ., και για κάθε λέξη που παράγουμε την αναφέρουμε στη λίστα των προγραμμάτων, αν αυτή είναι πρόγραμμα (πληροί δηλ. τους συντακτικούς κανόνες της γλώσσας προγραμματισμού που χρησιμοποιούμε). Μπορούμε να το κάνουμε αυτό ακριβώς επειδή όλες οι λέξεις πάνω από το Σ μήκους k είναι φυσικά πεπερασμένες το πλήθος, και άρα κάποια στιγμή τελειώνουμε την απαρίθμηση των λέξεων μήκους k και προχωράμε στην απαρίθμηση των λέξεων μήκους $k+1$. Κάθε πρόγραμμα λοιπόν θα εμφανιστεί στη λίστα μας κάποια στιγμή ανάλογα και με το ποιο είναι το μήκος του. Έχουμε λοιπόν μέχρι στιγμής δείξει ότι το πλήθος όλων των υπολογισίμων συναρτήσεων είναι αριθμήσιμο.

Η απόδειξη του θεωρήματος συμπληρώνεται από το ότι το πλήθος *όλων* των συναρτήσεων από το $\mathbf{N}$ στο $\mathbf{N}$ δεν είναι αριθμήσιμο, δε μπορούμε δηλ. να διατάξουμε όλες αυτές τις συναρτήσεις σε μια ακολουθία

$$g_1, g_2, \ldots, g_n, \ldots$$

Αν μπορούσαμε τότε ας κοιτάξουμε τη συνάρτηση

$$f(k) = g_k(k) + 1.$$

Αυτή είναι προφανές μια συνάρτηση από το $\mathbf{N}$ στο $\mathbf{N}$ άρα, αφού υποθέσαμε ότι όλες οι συναρτήσεις αυτές βρίσκονται στη λίστα g_i, υπάρχει κάποιο $i \in \mathbf{N}$ τέτοιο ώστε $f(k) = g_i(k)$ για κάθε $k \in \mathbf{N}$. Αυτό σημαίνει $g_k(k) + 1 = g_i(k)$. Διαλέγοντας $k = i$ παίρνουμε άτοπο. (Το παραπάνω επιχείρημα ονομάζεται 'διαγώνιο επιχείρημα'.)
□

Στην παραπάνω απόδειξη δείξαμε ότι υπάρχουν μη υπολογίσιμες συναρτήσεις δείχνοντας ότι το σύνολο των υπολογισίμων συναρτήσεων είναι αριθμήσιμο ενώ, αντίθετα, το σύνολο όλων των συναρτήσεων δεν είναι, είναι αυτό που λέμε *υπεραριθμήσιμο* σύνολο. Άρα, δεν μπορούν τα δύο σύνολα να είναι ίσα.

Αυτός είναι ένας κλασικός τρόπος απόδειξης στα μαθηματικά, η 'υπαρξιακή απόδειξη', που αν και είναι αρκετά εύκολος και αποδεικνύει αυτό που θέλουμε, δεν είναι τόσο ικανοποιητικός επειδή αποδεικνύεται η ύπαρξη ενός αντικειμένου με ορισμένες ιδιότητες, αλλά, συνήθως, δεν προσδιορίζεται με κάποιο τρόπο κάποιο τέτοιο αντικείμενο. Στην προκειμένη περίπτωση γνωρίζουμε πλέον, μετά την απόδειξη του Θεωρήματος 15, ότι υπάρχουν συναρτήσεις από τους φυσικούς στους φυσικούς που δεν είναι υπολογίσιμες, αλλά δε γνωρίζουμε καμία συγκεκριμένη τέτοια συνάρτηση. Θα δούμε αργότερα συγκεκριμένα παραδείγματα που για το καθένα θα έχουμε και ιδιαίτερο τρόπο απόδειξης του ότι δεν είναι υπολογίσιμα.

Άσκηση 64. Δεν υπάρχει κάποιος ιδιαίτερος λόγος που στο Θεώρημα 15 αναφερόμαστε σε συναρτήσεις $\mathbf{N} \to \mathbf{N}$. Δείξτε, για παράδειγμα, ότι υπάρχουν συναρτήσεις $\mathbf{N} \to \{0,1\}$ και $\Sigma^* \to \Sigma^*$, που δεν είναι υπολογίσιμες (Σ είναι ένα πεπερασμένο αλφάβητο).

6.3 Το Halting Problem. Άλλα μη αποφασίσιμα προβλήματα στα Μαθηματικά.

Τα προγράμματα ως αριθμοί, και το αντίστροφο Για τη συζήτηση από δω και πέρα θα χρειαστούμε να βλέπουμε το τυχόν C πρόγραμμα ως ένα φυσικό αριθμό, και τον τυχόντα φυσικό αριθμό ως ένα C πρόγραμμα. Πώς γίνεται αυτό; Πολύ απλά, ένα C πρόγραμμα δεν είναι παρά ένα κομμάτι κειμένου, μια λέξη δηλ. πάνω από ένα συγκεκριμένο αλφάβητο. Το αλφάβητο που χρησιμοποιείται στους υπολογιστές σήμερα είναι το

$$\Sigma = \{b_7 b_6 \cdots b_1 b_0 : \ b_i \in \{0,1\}, i = 0, \ldots, 7\}.$$

Αυτές είναι όλες οι λέξεις (bytes) με οκτώ δυαδικά ψηφία (bits) η κάθε μία και πολλές φορές ταυτίζουμε το συγκεκριμένο αλφάβητο με τους αριθμούς $0, \ldots, 255$. Ένα πρόγραμμα P λοιπόν είναι μια λέξη

$$P = w_1 \cdots w_N, \quad w_i \in \Sigma$$

και δεν έχουμε παρά να αντικαταστήσουμε κάθε w_i με τα δυαδικά του ψηφία ώστε να γράψουμε το P σα μια δυαδική λέξη με $8N$ δυαδικά ψηφία

$$P = b_{8N-1} b_{8N-2} \cdots b_1 b_0$$

την οποία λέξη μπορούμε να ερμηνεύσουμε ως ένα φυσικό αριθμό, που τον συμβολίζουμε στο εξής ως $\alpha(P)$. Αντίστροφα, αν n είναι ένας φυσικός αριθμός τον γράφουμε στο δυαδικό σύστημα και συμπληρώνουμε με μηδενικά (από τα αριστερά) το πλήθος των ψηφίων του αν χρειάζεται ώστε να είναι πολλαπλάσιο του 8, έστω $8k$. Κάθε μια από τις k οκτάδες τώρα τη βλέπουμε σαν ένα στοιχείο του Σ και ο αριθμός μας γίνεται τώρα μια λέξη του Σ^*. Αν αυτή η λέξη είναι ένα συντακτικά σωστό πρόγραμμα σε C τότε ονομάζουμε αυτό το πρόγραμμα $\pi(n)$, αν όχι, θέτουμε $\pi(n)$ να είναι το πρόγραμμα `main() { return 1; }`. (Η επιλογή αυτού του τελευταίου προγράμματος είναι τελείως τυχαία. Θα μπορούσαμε αντί γι' αυτό να επιστρέφουμε οποιοδήποτε άλλο.)

Η απεικόνιση $P \to \alpha(P)$ είναι εύκολο να δεί κανείς ότι είναι ένα προς ένα (αλλά όχι επί, αφού δεν είναι όλοι οι αριθμοί κωδικοποιήσεις συντακτικά σωστών προγραμμάτων), και ότι η απεικόνιση $n \to \pi(n)$ δεν είναι ένα προς ένα, αφού μπορεί δύο προγράμματα να μοιάζουν διαφορετικά αλλά να κάνουν την ίδια δουλειά, να υπολογίζουν δηλαδή την ίδια συνάρτηση. Είναι επίσης σημαντικό το ότι και οι δύο απεικονίσεις είναι υπολογίσιμες. Το ότι η $\alpha(P)$ είναι υπολογίσιμη είναι φανερό, ενώ το μόνο λεπτό σημείο όσον αφορά την υπολογισιμότητα της $\pi(n)$ είναι στο κομμάτι όπου πρέπει να ελέγξουμε αν η λέξη του Σ^* που προκύπτει είναι συντακτικά σωστό C πρόγραμμα ή όχι. Το να αποδείξουμε ότι κάτι τέτοιο είναι υπολογστικά εφικτό σημαίνει ουσιαστικά να γράψουμε ένα πρόγραμμα που να αναγνωρίζει αν ένα κομμάτι κειμένου (μια λέξη του Σ^*) είναι ένα σωστό συντακτικά C πρόγραμμα ή όχι. Τέτοια προγράμματα όμως υπάρχουν και χρησιμοποιούνται ευρέως αφού αποτελούν το πρώτο τμήμα των complilers (μεταφραστών) που παίρνουν ένα πρόγραμμα σε C και παράγουν από αυτό ένα ισοδύναμο πρόγραμμα σε 'γλώσσα μηχανής'.

Προσομοίωση. Θα χρειαστούμε επίσης το εξής: Υπάρχει ένα πρόγραμμα $S(n, x, t)$ το οποίο (α) βρίσκει το πρόγραμμα $P = \pi(n)$ και (β) 'Τρέχει' το πρόγραμμα $P(x)$ για τόσα βήματα όσα λέει ο αριθμός t ή επ άπειρον αν ο αριθμός t είναι αρνητικός. Αν και όταν το πρόγραμμα $P(x)$ τελειώσει το $S(n, x, t)$ επιστρέφει ότι επέστρεψε το $P(x)$. Δε θα δείξουμε την ύπαρξη αυτού του προγράμματος που προσομοιώνει άλλα προγράμματα, μια και τέτοια προγράμματα υπάρχουν και χρησιμοποιούνται ευρέως στον προγραμματισμό (π.χ. interpreters, debuggers, κλπ).

Είμαστε έτοιμοι τώρα να αναφερθούμε στο Halting problem (πρόβλημα τερματισμού). Αυτό ρωτάει απλά αν υπάρχει ένα πρόγραμμα που να μπορεί να αποφασίζει αν ένα τυχόν άλλο πρόγραμμα που εμείς του προσδιορίζουμε θα τερματίσει ή όχι. (Είναι φανερό ότι υπάρχουν προγράμματα που δεν τερματίζουν ποτέ, π.χ. το `main() { while(1) 1; }`). Ένα τέτοιο πρόγραμμα θα ήταν εξαιρετικά χρήσιμο αν υπήρχε. Φανταστείτε να έχετε ένα C compiler ο οποίος όχι μόνο θα μετάφραζε το πρόγραμμά σας σε γλώσσα μηχανής αλλά θα μπορούσε και να σας πει εκ των προτέρων αν το πρόγραμμά σας θα έπεφτε σε άπειρο βρόγχο (loop) ή όχι. Δυστυχώς όμως τέτοιο πρόγραμμα δεν υπάρχει.

Στο εξής, για τυχόν πρόγραμμα P, θα γράφουμε $P \downarrow$ αν το πρόγραμμα P τερματίζει και $P \uparrow$ αν το P τρέχει επ' άπειρον.

Θεώρημα 16. *Η συνάρτηση*

$$H(n,x) = \begin{cases} 1 & \alpha\nu \quad (\pi(n))(x) \downarrow \\ 0 & \alpha\nu \quad (\pi(n))(x) \uparrow \end{cases}$$

δεν είναι υπολογίσιμη.

Απόδειξη. Ας υποθέσουμε ότι η συνάρτηση $H(n,x)$ είναι υπολογίσιμη και ας συμβολίσουμε με P το πρόγραμμα που την υπολογίζει. Φτιάχνουμε τώρα το πρόγραμμα $Q(x)$ που χρησιμοποιεί το P ως υποπρόγραμμα:

```
int Q(x)
{
    if (P(x, x) == 0) return 1;
    else {
     while(1) 1;
    }
}
```

Το πρόγραμμα $Q(x)$ που ορίσαμε έχει την ιδιότητα ότι τερματίζει αν και μόνο αν το $(\pi(x))(x)$ δεν τερματίζει.

Ποια είναι η συμπεριφορά του προγράμματος $Q(\alpha(Q))$; Τερματίζει ή όχι; Σύμφωνα με τη προηγούμενη παρατήρηση τερματίζει αν και μόνο αν το πρόγραμμα $(\pi(\alpha(Q)))(\alpha(Q))$ δεν τερματίζει. Αλλά $\pi(\alpha(Q)) = Q$, οπότε δείξαμε ότι το πρόγραμμα $Q(\alpha(Q))$ τερματίζει αν και μόνο αν δεν τερματίζει, πράγμα προφανώς αδύνατο. Το συμπέρασμα είναι ότι η συνάρτηση H δεν είναι υπολογίσιμη.
□

Έχουμε λοιπόν δείξει ότι η, πολύ φυσιολογική, συνάρτηση $H(n,x)$ που διακρίνει πότε το πρόγραμμα $\pi(n)$ τερματίζει με input x ή όχι δεν είναι υπολογίσιμη. Έχουμε λοιπόν τώρα μια απόδειξη του ότι υπάρχουν μη υπολογίσιμες συναρτήσεις που είναι αρκετά πιο συγκεκριμένη από την προηγούμενη υπαρξιακή απόδειξη του Θεωρήματος 15. Μπορεί βέβαια και πάλι να πει κανείς ότι η $H(x)$ δεν έχει ίσως και τόσο ενδιαφέρον από μαθηματική άποψη μια και είναι μια συνάρτηση 'κομμένη και ραμμένη' στα μέτρα της Θεωρίας Υπολογισμών, και άρα όχι, ίσως, τόσο φυσιολογική.

Υπάρχουν όμως πολλά παραδείγματα αρκετά φυσιολογικών μαθηματικών προβλημάτων που δεν επιδέχονται λύση αλγοριθμική, προβλήματα που προυπήρχαν της Θεωρίας Υπολογισμών.

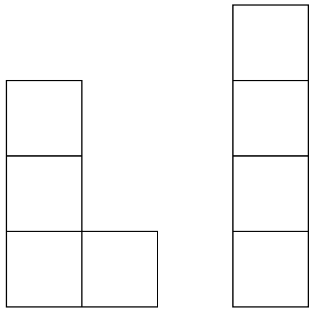
1. Δεν υπάρχει αλγόριθμος που να αποφασίζει αν ένα δοσμένο πολυώνυμο με ακεραίους συντελεστές (και οποιοδήποτε πλήθος από μεταβλητές)

   $$P(x_1,\ldots,x_n) = \sum_{0\le j_1,\ldots,j_n\le D} c_{j_1,\ldots,j_n} x_1^{j_1}\cdots x_n^{j_n}, \quad c_{j_1,\ldots,j_n} \in \mathbf{Z},$$

   έχει λύση (ρίζα) ανάμεσα στους ακεραίους, αν υπάρχουν δηλ. $x_1,\ldots,x_n \in \mathbf{Z}$ τ.ώ. $P(x_1,\ldots,x_n) = 0$.

2. Είναι ενδιαφέρον ότι αν ρωτήσουμε αν υπάρχει ρίζα πραγματική (και όχι αναγκαστικά ακέραια), αν δηλ. υπάρχουν $x_1,\ldots,x_n \in \mathbf{R}$ τ.ώ. $P(x_1,\ldots,x_n) = 0$, τότε υπάρχει αλγόριθμος που να απαντάει στο ερώτημα.

3. Ένα *πολυόμινο* είναι μια πεπερασμένη ένωση από μη αλληλοεπικαλυπτόμενα, μοναδιαία (πλευράς 1) τετράγωνα στο επίπεδο, τέτοια ώστε οι συντεταγμένες των κορυφών τους είναι ακέραιες. Στο Σχήμα 19 φαίνεται ένα σύνολο από δύο πολυόμινα. Ένα πολύ ενδιαφέρον ερώτημα είναι αν ένα δεδομένο πεπερασμένο σύνολο από πολυόμινα μπορεί να χρησιμοποιηθεί για να 'πλακοστρώσει' ολόκληρο το επίπεδο. Αυτό σημαίνει ότι καλύπτουμε το επίπεδο, χωρίς αλληλοεπικαλύψεις, από κομμάτια καθ' ένα από τα οποία είναι μια παράλληλη μεταφορά κάποιου από τα δεδομένα πολυόμινα. Για παράδειγμα, είναι σχετικά εύκολο να δει κανείς πως τα δύο πολυόμινα του Σχήματος 19 μπορούν να πλακοστρώσουν (αγγλικά: tiling) το επίπεδο, και είναι επίσης εύκολο να φτιάξει κανείς παραδείγματα απο σύνολο πολυομίνων που δε μπορούν, όπως και να μεταφερθούν, να πλακοστρώσουν το επίπεδο. Αποδεικνύεται λοιπόν ότι το να



Σχήμα 19. Ένα σύνολο από δύο πολυόμινα

αποφασίσει κανείς αν ένα δεδομένο σύνολο από πολυόμινα μπορεί να πλακοστρώσει το επίπεδο είναι αλγοριθμικά ανεπίλυτο πρόβλημα. Δεν υπάρχει δηλ. πρόγραμμα που να του δίνουμε ένα τυχόν πεπερασμένο σύνολο από πολυόμινα και να μας απαντά αν επιδέχεται πλακόστρωση ή όχι.

6.4 Αναδρομικά απαριθμήσιμα (α.α.) σύνολα.

Ορισμός 39. Ένα μη κενό σύνολο $A \subseteq \mathbf{N}$ ονομάζεται *αναδρομικά απαριθμήσιμο* (α.α.) αν υπάρχει πρόγραμμα $P(x)$ που να απαριθμεί τα στοιχεία του A, δηλ. να έχουμε

$$A = \{P(1), P(2), P(3), \ldots\},$$

με ενδεχόμενη επανάληψη των στοιχείων του A στο δεξί μέλος. Με άλλα λόγια, για κάθε $a \in A$ υπάρχει $n \in \mathbf{N}$, όχι κατ΄ ανάγκη μοναδικό, τέτοιο ώστε $a = P(n)$.

Παρατήρηση: Και πάλι, δεν υπάρχει, στον Ορισμό 39, τίποτα το μαγικό με το σύνολο $\mathbf{N}$ και θα μπορούσαμε στη θέση του να έχουμε, π.χ. το σύνολο Σ^*, όπου Σ ένα πεπερασμένο αλφάβητο.

Είναι σχεδόν άμεσο ότι κάθε αναδρομικό σύνολο είναι και α.α. Πράγματι, αν $\emptyset \neq A \subseteq \mathbf{N}$ είναι αναδρομικό τότε υπάρχει πρόγραμμα $Q(x)$ τέτοιο ώστε $Q(x) = 1 \Leftrightarrow x \in A$. Τότε το ακόλουθο πρόγραμμα απαριθμεί τα στοιχεία του A (m είναι, π.χ., το μικρότερο στοιχείο του A):

```
int P(x)
{
    if(Q(x)) return x;
    else return m;
}
```

Άσκηση 65. Αν $A, B \subseteq \mathbf{N}$ είναι α.α. σύνολα δείξτε ότι το ίδιο ισχύει και για τα σύνολα $A \cap B, A \cup B$.

Από την άλλη μεριά είναι εύκολο να κατασκευάσουμε α.α. σύνολα που δεν είναι αναδρομικά. Για παράδειγμα, το σύνολο

$$A = \{x \in \mathbf{N} : \ \pi(x) \downarrow\}$$

δεν είναι αναδρομικό (αν ήταν τότε το πρόβλημα του τερματισμού θα ήταν υπολογίσιμο, ενώ έχουμε δείξει ότι δεν είναι) αλλά είναι α.α. Για να δούμε αυτό τον τελευταίο ισχυρισμό κάνουμε το εξής. Χρειαζόμαστε ένα πρόγραμμα που να απαριθμεί τα $x \in \mathbf{N}$ για τα οποία το πρόγραμμα $\pi(x)$ τερματίζει. Ισοδύναμα, φτιάχνουμε ένα πρόγραμμα που λειτουργεί επ΄ άπειρον και τυπώνει συνέχεια στην έξοδό του στοιχεία του A, χωρίς να παραλείψει κανένα. Το πρόγραμμα αυτό

- **Βήμα 1:** Πρώτα τρέχει (προσομοιώνει) το πρόγραμμα $\pi(1)$ για 1 βήμα.

- **Βήμα 2:** Έπειτα τρέχει (προσομοιώνει) τα προγράμματα $\pi(1), \pi(2)$ για 2 βήματα το καθένα
- ...
- **Βήμα k:** Τρέχει τα προγράμματα $\pi(1), \pi(2), \ldots, \pi(k)$ για k βήματα το καθένα,
- Συνεχίζει έτσι επ΄ άπειρον.

Στο τέλος του βήματος k το πρόγραμμά μας ελέγχει ποια από τα προγράμματα $\pi(1), \ldots, \pi(k)$ τελείωσαν μέσα στα πρώτα k τους βήματα, και τυπώνει τα αντίστοιχα x για αυτά που τελείωσαν στην έξοδό του. Είναι έτσι φανερό ότι κάθε x για το οποίο $\pi(x) \downarrow$ θα εμφανιστεί στην έξοδο του προγράμματός μας, και ότι σε αυτή την έξοδο εμφανίζονται μόνο τέτοια x (δηλ. στοιχεία του A). Αν $x \in A$ και το πρόγραμμα $\pi(x)$ τερματίζει μετά από t βήματα τότε ο αριθμός x εμφανίζεται για πρώτη φορά στην έξοδο του προγράμματός μας μετά το βήμα υπ΄ αριθμόν $\max\{x, t\}$, και εμφανίζεται και σε όλα τα μεταγενέστερα βήματα.

Έχουμε λοιπόν δείξει:

Θεώρημα 17. *Κάθε αναδρομικό σύνολο είναι αναδρομικά απαριθμήσιμο αλλά το αντίστροφο δεν ισχύει.*

Η ιεραρχία των γλωσσών τώρα γίνεται

$$(\text{κανονικές γλώσσες}) \subset (\text{context free γλώσσες}) \subset (\text{αναδρομικές γλώσσες}) \subset (\text{α.α. γλώσσες})$$

και όλοι οι εγκλεισμοί είναι γνήσιοι.

Η σχέση αναδρομικών και α.α. συνόλων φαίνεται πιο καθαρά στα επόμενα δύο θεωρήματα.

Θεώρημα 18. *$A \subseteq \mathbf{N}$ είναι αναδρομικό αν και μόνο αν τα A, A^c είναι και τα δύο α.α.*

Απόδειξη: Αν το A είναι αναδρομικό τότε αναδρομικό είναι και το A^c, άρα και τα δύο είναι α.α. Αντίστροφα τώρα, έστω ότι τα A, A^c είναι και τα δύο α.α., και ότι τα προγράμματα P και Q τα απαριθμούν, έχουμε δηλ.

$$A = \{P(1), P(2), \ldots\},\ A^c = \{Q(1), Q(2), \ldots\}.$$

Το παρακάτω πρόγραμμα $R(x)$, που χρησιμοποιεί τα P και Q ως υποπρογράμματα, επιστρέφει 1 αν $x \in A$ και 0 αλλιώς.

```
int R(x)
{
    int i=1;
    label:
    if(P(i) == x) return 1;
    if(Q(i) == x) return 0;
    i = i+1; goto label;
}
```

Το πρόγραμμα R απλούστατα παράγει όλους τους αριθμούς $P(1), Q(1), P(2), Q(2), \ldots$ και μόλις εμφανιστεί το x επιστρέφει 0 ή 1 ανάλογα με το αν το x εμφανίστηκε στην έξοδο του Q ή του P. Είναι σίγουρο ότι το x κάποτε θα εμφανιστεί (μια και είτε θα ανήκει στο A είτε στο A^c) και άρα για κάθε x το πρόγραμμα $R(x)$ τερματίζει.
□

Θεώρημα 19. *Ένα σύνολο $A \subseteq \mathbf{N}$ είναι α.α. αν και μόνο αν υπάρχει πρόγραμμα $Q(x)$ τέτοιο ώστε*

- *Αν $x \in A$ τότε $Q(x) = 1$, και*
- *Αν $x \notin A$ τότε $Q(x) = 0$ ή $Q(x) \uparrow$.*

Το πρόγραμμα Q στο θεώρημα λέει την αλήθεια για το αν $x \in A$, αν τελειώνει, μπορεί όμως και να τρέχει επ' άπειρον (αλλά αυτό μπορεί να συμβαίνει μόνο αν $x \in A$. Παρατηρείστε ότι υπάρχει ασυμμετρία στις δύο περιπτώσεις $x \in A$ και $x \notin A$, και αυτή η ασυμμετρία εκδηλώνεται και με το ότι υπάρχουν α.α. σύνολα που τα συμπληρώματά τους δεν είναι α.α.

Απόδειξη: Έστω ότι το πρόγραμμα P απαριθμεί τα στοιχεία του A:

$$A = \{P(1), P(2), \ldots\}.$$

Ορίζουμε το πρόγραμμα $Q(x)$ ως εξής:

```
int Q(x)
{
    int i=1;
    label:
    if(P(i) == x) return 1;
    i = i+1; goto label;
}
```

Το πρόγραμμα $Q(x)$ επιστρέφει 1 αν και μόνο αν το x εμφανιστεί στη λίστα των τιμών του P, αλλιώς τρέχει επ' άπειρον.

Η άλλη κατεύθυνση του Θεωρήματος 19 αφήνεται ως άσκηση.
□

Άσκηση 66. Αποδείξτε την κατεύθυνση του Θεωρήματος 19 που λείπει από την απόδειξη που δόθηκε παραπάνω.

Ευρετήριο