

---

---

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών  
και Πληροφορικής

## Θεωρία Υπολογισμού

Διδακτικές Σημειώσεις

ΧΡΗΣΤΟΣ ΚΑΚΛΑΜΑΝΗΣ  
ΚΑΘΗΓΗΤΗΣ

Πάτρα, Ιούνιος 2021

---

---



# Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>7</b>
1.1	Εισαγωγή στη Θεωρία Υπολογισμού . . . . .	7
1.2	Σύνολα . . . . .	9
1.3	Πεπερασμένα και Άπειρα Σύνολα . . . . .	12
1.4	Συναρτήσεις και σχέσεις . . . . .	13
1.5	Γραφήματα, συμβολοσειρές, και γλώσσες . . . . .	15
1.5.1	Γραφήματα . . . . .	15
1.5.2	Συμβολοσειρές (Strings) και Γλώσσες (Languages) . . . . .	15
1.6	Δυαδική λογική και επισκόπηση όρων . . . . .	19
1.7	Ορισμοί, θεωρήματα, και αποδείξεις . . . . .	19
1.8	Αποδείξεις . . . . .	19
1.8.1	Απόδειξη με Επαγωγή (Proof by induction) . . . . .	20
1.8.2	Αρχή του Περιστεριώνα (The Pigeonhole Principle) . . . . .	21
1.8.3	Απόδειξη με Εις Άτοπον Απαγωγή (Proof by contradiction - reductio ad absurdum) . . . . .	23
1.8.4	Αρχή της Διαγωνοποίησης . . . . .	24
<b>2</b>	<b>Κανονικές γλώσσες</b>	<b>29</b>
2.1	Πεπερασμένα αυτόματα . . . . .	33
2.1.1	Κατασκευή FAs . . . . .	35
2.1.2	Αναπαράσταση FAs . . . . .	38
2.2	Κανονικές Εκφράσεις (Regular Expressions) . . . . .	43
2.2.1	Το θεώρημα του Kleene . . . . .	46
2.2.2	Εφαρμογές των REs . . . . .	46
2.3	Μη ντετερμινισμός . . . . .	49
2.3.1	Μη ντετερμινιστικά πεπερασμένα αυτόματα . . . . .	49
2.3.2	Η έννοια του μη ντετερμινισμού . . . . .	51
2.3.3	$\epsilon$ - μεταβάσεις . . . . .	52

2.3.4	Το θεώρημα του Kleene . . . . .	53
2.3.5	Μετατροπή κανονικής έκφρασης σε NFA . . . . .	53
2.3.6	Μετατροπή NFA σε DFA . . . . .	56
2.3.7	Μετατροπή FA σε κανονική έκφραση . . . . .	65
2.4	Ιδιότητες των κανονικών γλωσσών . . . . .	70
2.4.1	Πράξεις που ορίζονται με γλώσσες . . . . .	70
2.4.2	Ιδιότητες κλειστότητας . . . . .	73
2.4.3	Κατασκευαστικές αποδείξεις . . . . .	75
2.5	Μη κανονικές γλώσσες . . . . .	78
2.5.1	Διακρίσιμες συμβολοσειρές . . . . .	78
2.5.2	Το Pumping Lemma . . . . .	80
2.6	Εφαρμογές πεπερασμένων αυτομάτων . . . . .	89
2.6.1	Επεξεργασία συμβολοσειρών (String Processing) . . . . .	89
2.6.2	Μηχανές πεπερασμένων καταστάσεων (Finite - State Machines) . . . . .	90
2.6.3	Διαγράμματα καταστάσεων . . . . .	91
2.6.4	Λεκτική ανάλυση (Lexical Analysis) . . . . .	92
<b>3</b>	<b>Γραμματικές χωρίς συμφραζόμενα και αυτόματα στοίβας</b>	<b>95</b>
3.1	Γραμματικές και γλώσσες χωρίς συμφραζόμενα . . . . .	95
3.1.1	Γραφήματα Παραγωγής . . . . .	104
3.1.2	Αριστερότερες Παραγωγές (Leftmost derivations) . . . . .	105
3.1.3	Ιεραρχική Δόμηση Γραμματικών . . . . .	106
3.1.4	Γραμματικές Χωρίς Συμφραζόμενα . . . . .	111
3.1.5	Κανονικές Γλώσσες και Γραμματικές Χωρίς Συμφραζόμενα . . . . .	112
3.2	Αυτόματα στοίβας . . . . .	113
3.3	Ισοδυναμία αυτομάτων στοίβας με γραμματικές χωρίς συμφραζόμενα . . . . .	119
3.4	Ιδιότητες κλειστότητας των γλωσσών χωρίς συμφραζόμενα . . . . .	122
3.5	Το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα . . . . .	123
3.6	Εφαρμογές του Pumping Lemma . . . . .	126
3.7	Σύνοψη . . . . .	129
<b>4</b>	<b>Μηχανές Turing και υπολογιστική θεωρία</b>	<b>131</b>
4.1	Παραδείγματα μηχανών Turing . . . . .	135
4.2	Παραλλαγές του βασικού μοντέλου μηχανής Turing . . . . .	141
4.2.1	Μηχανές Turing πολλαπλών ταινιών . . . . .	141
4.2.2	Μη Ντετερμινιστικές Μηχανές Turing . . . . .	143
4.3	Ορισμός της έννοιας του αλγορίθμου . . . . .	145

4.4	Αποφασισιμότητα: Αποφασίσιμα προβλήματα σχετικά με κανονικές γλώσσες . . . .	146
4.5	Αποφασίσιμα προβλήματα σχετικά με γλώσσες χωρίς συμφραζόμενα . . . . .	149



# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Εισαγωγή στη Θεωρία Υπολογισμού

Η επεξεργασία πληροφορίας οποιασδήποτε μορφής επιτυγχάνεται μέσω υπολογισμών, η πολυπλοκότητα των οποίων μεταβάλλεται ανάλογα με το είδος της πληροφορίας. Η μελέτη των υπολογισμών αυτών οδηγεί στον εντοπισμό, την ανάλυση και την αξιοποίηση των ιδιαίτερων χαρακτηριστικών τους και παρέχει, κατά συνέπεια, τη δυνατότητα για πρόβλεψη της πολυπλοκότητας επιθυμητών υπολογισμών, για επιλογή κατάλληλης προσέγγισης πραγματοποίησής τους καθώς και για το σχεδιασμό και την ανάπτυξη εργαλείων διεκπεραίωσής τους. Υπάρχουν προβλήματα που δε μπορούν να λυθούν και από αυτά για τα οποία υπάρχει δυνατότητα επίλυσης, μερικά απαιτούν μη εφικτή ποσότητα πόρων (π.χ., εκατομμύρια χρόνια υπολογιστικού χρόνου). Η συστηματική και μεθοδευμένη μελέτη των προβλημάτων, γενικά, και των ενδεχόμενων τρόπων επίλυσής τους, ειδικότερα, αποτελεί μια διαδικασία “χαρτογράφησης” τους κι έχει σαν αποτέλεσμα την παρουσίαση προσεγγίσεων για την επιλυσιμότητα ή όχι τάξεων προβλημάτων. Η Θεωρία Υπολογισμού, αποτελεί ουσιαστικά ένα αυστηρά μαθηματικό εργαλείο για την εφαρμογή όλων όσων προαναφέρθηκαν. Βασικός της στόχος είναι η τυπική και αυστηρή έκφραση πραγματικών προβλημάτων με χρήση γενικευμένων, ισχυρών μαθηματικών μοντέλων μέσω των οποίων τα προβλήματα ταξινομούνται με βάση την υπολογιστική πολυπλοκότητα που απαιτείται για την επίλυσή τους, στις περιπτώσεις που αυτή είναι εφικτή, και παρέχονται συγκεκριμένες προσεγγίσεις επίλυσης για κάθε κατηγορία. Σε ό, τι ακολουθεί θα αναφερθούμε, αρχικά, σε βασικές έννοιες για την κατανόηση της Θεωρίας Υπολογισμού, όπως οι συμβολοσειρές (strings) και ο ρόλος τους στη διαδικασία αναπαράστασης της πληροφορίας. Στη συνέχεια θα μελετήσουμε την έννοια των γλωσσών και των γραμματικών, και θα παρουσιάσουμε μαθηματικά μοντέλα για την τυπική έκφρασή τους και τη μελέτη τους, εξετάζοντας τις έννοιες του ντετερμινισμού και μη-ντετερμινισμού. Τέλος, σε όλη την έκταση της συζήτησης, θα αναφερόμαστε στις έννοιες της τυπικής απόδειξης, μιας αυστηρά μαθηματικής διαδικασίας, που πετυχαίνεται με διάφορους τρόπους και οδηγεί στη διατύπωση συμπερασμάτων γενικής ισχύος με τη μορφή λημμάτων και θεωρημάτων. Στη συνέχεια συνοψίζονται οι πιο βασικές έννοιες που εισάγονται με τη Θεωρία

Υπολογισμού.

- Ένας πραγματικός υπολογιστής μπορεί να μοντελοποιηθεί με χρήση μαθηματικών, δίνοντας ένα θεωρητικό υπολογιστή. Ένας πραγματικός υπολογιστής πρόκειται για ένα αρκετά πολύπλοκο αντικείμενο που περιέχει Κεντρική Μονάδα Επεξεργασίας (CPU), μνήμη (memory), συσκευές εισόδου και εξόδου (Input/Output Devices), κ.τ.λ. Αν προσπαθήσουμε να δημιουργήσουμε σχέσεις - εντολές σχετικά με το τι μπορεί να υπολογίσει ένας συγκεκριμένος υπολογιστής, ενδεχομένως να μην έχουν εφαρμογή για άλλους υπολογιστές αφού είναι λογικό κάποιες παράμετροι, που αφορούν στο σχεδιασμό διαφορετικών μηχανών, να μεταβάλλονται. Το επιθυμητό είναι να καθορίσουμε αφηρημένες και γενικές αρχές, που να μην εξαρτώνται από τα ιδιαίτερα σχεδιαστικά χαρακτηριστικά του κάθε υπολογιστή, με βάση τις οποίες θα μπορούν να κατασκευαστούν αφηρημένα (λογικά) μοντέλα. Τότε, σχέσεις και εντολές που θα ορίζονται για τα μοντέλα αυτά θα έχουν γενική ισχύ και κάποιος περιορισμός τους θα είναι αυτός που θα ισχύει για κάθε τύπο πραγματικού υπολογιστή.
- Μια τυπική γλώσσα είναι ένα σύνολο συμβολοσειρών και μπορεί να αναπαραστήσει ένα υπολογιστικό πρόβλημα. Είναι κοινός τόπος ότι οι φυσικές γλώσσες χαρακτηρίζονται από ιδιαίτερη πολυπλοκότητα. Είναι δυνατή η ενοποίηση, με κάποιον τρόπο, όλων των φυσικών γλωσσών κάτω από ένα, ενιαίο σύνολο κανόνων; Είναι δυνατή η ανάπτυξη ενός λογικού μοντέλου το οποίο επιτρέπει την κατασκευή εργαλείων για την επεξεργασία διαφορετικών γλωσσών; Σκοπός είναι ο αποδοτικός σχεδιασμός γλωσσών για υπολογιστές που να είναι ευέλικτες και εύκολα μεταφράσιμες σε γλώσσα μηχανής.
- Μια τυπική γλώσσα μπορεί να περιγραφεί με διαφορετικούς τρόπους που τελικά αποδεικνύονται ισοδύναμοι. Υπάρχουν δύο θεμελιώδεις τεχνικές για την περιγραφή γλωσσών που έχουν υιοθετηθεί από μαθηματικούς αφενός και αφετέρου από επιστήμονες που ασχολούνται με τη θεωρία των υπολογιστών. Μια τεχνική είναι η κατασκευή αναγνωριστών γλωσσών (language recognizers), αφηρημένων, δηλαδή, μηχανών που δέχονται σαν είσοδο συμβολοσειρές και “ αποφασίζουν ” αν αυτές ανήκουν ή όχι σε δεδομένες γλώσσες. Σαν εναλλακτική λύση προτείνονται οι γεννήτριες γλωσσών (language generators), δηλαδή σύνολα κανόνων οι οποίοι όταν εφαρμοστούν “ γεννούν ” συμβολοσειρές που ανήκουν σε δεδομένη γλώσσα. Οι γραμματικές μπορούν να θεωρηθούν γεννήτριες γλωσσών.
- Εξομοίωση (simulation): η σχετική δύναμη των υπολογιστικών μοντέλων βασίζεται στην ευκολία εξομοίωσης ενός μοντέλου από κάποιο άλλο. Τα διάφορα υπολογιστικά μοντέλα δεν είναι κατ’ ανάγκη ισοδύναμα όσον αφορά στην υπολογιστική τους ισχύ. Ενδεχόμενη σύγκρισή τους θα μπορούσε να πραγματοποιηθεί με βάση τις κλάσεις των γλωσσών που αποδέχονται. Αυτό πετυχαίνεται με εξομοίωση (simulation): αν μπορούμε να δείξουμε ότι ένα υπολογιστικό μοντέλο  $M$  μπορεί να εξομοιώσει τις εντολές ενός υπολογι-



στικού μοντέλου  $M'$ , τότε είναι προφανές ότι κάθε γλώσσα που γίνεται αποδεκτή από το  $M'$  γίνεται επίσης αποδεκτή από το  $M$ .

- **Ισχύς (robustness) των υπολογιστικών μοντέλων.** Ένα μοντέλο είναι ισχυρό και φυσικό αν μικρές διαφοροποιήσεις του μοντέλου αφήνουν την υπολογιστική του ισχύ αμετάβλητη. Ένα ισχυρό μοντέλο αποτελεί ταυτόχρονα και ένα λογικό και αξιόπιστο μοντέλο για την επίλυση πραγματικών προβλημάτων.
- **Η θέση των Church–Turing: οτιδήποτε είναι υπολογίσιμο μπορεί να υπολογιστεί από μια μηχανή Turing.** Η μηχανή Turing πρόκειται για το ισχυρότερο υπολογιστικό μοντέλο που έχει βρεθεί μέχρι σήμερα.
- **Μη ντετερμινισμός (Non – Determinism):** οι γλώσσες μπορούν να περιγραφούν από την ύπαρξη ή τη μη ύπαρξη υπολογιστικών μονοπατιών. Ο μη ντετερμινιστικός υπολογισμός επιτρέπει επιλογή μεταξύ εναλλακτικών λύσεων σε κάθε σημείο του υπολογισμού και κάποια είσοδος γίνεται αποδεκτή αν και μόνον αν υπάρχει ακολουθία επιλογών που να οδηγεί σε κατάσταση αποδοχής.
- **Μη επιλυσιμότητα (unsolvability):** για κάποια υπολογιστικά προβλήματα δεν υπάρχει αντίστοιχος αλγόριθμος που θα τα επέλυε χωρίς λάθος. Η τάξη των προβλημάτων για τα οποία δεν έχει ακόμα βρεθεί λύση έχει γοητεύσει τους επιστήμονες για χιλιάδες χρόνια. Η θεωρία υπολογισμού παρέχει αυστηρό τρόπο καθορισμού της ιδέας ενός προβλήματος και απόδειξης της ύπαρξης καλά ορισμένων προβλημάτων για τα οποία όμως δεν υπάρχει γενική λύση. Η αυστηρή διατύπωση είναι σε μεγάλο βαθμό ανεξάρτητη από το υπολογιστικό μοντέλο και τον υπολογιστή που χρησιμοποιείται.

## 1.2 Σύνολα

**Ορισμός 1.2.1.** *Σύνολο είναι μια συλλογή αντικειμένων, τα οποία καλούνται μέλη ή στοιχεία του συνόλου.*

**Παράδειγμα 1.2.1.**  $L = \{a, b, c, d\}$  είναι ένα σύνολο που καλείται  $L$  και αποτελείται από τα γράμματα  $a, b, c, d$ .

Για να εκφράσουμε ότι το  $b$  είναι στοιχείο του συνόλου  $L$  μπορούμε να γράψουμε  $b \in L$  (το  $b$  ανήκει στο σύνολο  $L$ ). Διαφορετικά, όταν δηλ. το  $b$  δεν είναι στοιχείο του συνόλου  $L$ , γράφουμε  $b \notin L$  (το  $b$  δεν ανήκει στο σύνολο  $L$ ).

Σε ένα σύνολο **δε** διακρίνουμε επαναλήψεις στοιχείων.

**Παράδειγμα 1.2.2.** Τα σύνολα  $\{\text{κόκκινο, μπλε, κόκκινο}\}$  και  $\{\text{κόκκινο, μπλε}\}$  είναι ίδια.

Σε ένα σύνολο **δεν** έχει σημασία η σειρά των στοιχείων.

**Παράδειγμα 1.2.3.** Επίσης, τα σύνολα  $\{3, 1, 9\}$ ,  $\{9, 3, 1\}$ ,  $\{1, 3, 9\}$  είναι ίδια.

**Ορισμός 1.2.2.** Δύο σύνολα είναι ίσα αν και μόνον αν έχουν τα ίδια στοιχεία.

**Ορισμός 1.2.3.** Μονομελές είναι ένα σύνολο με ένα μόνο στοιχείο, π.χ.  $\{a\}$ .

**Ορισμός 1.2.4.** Κενό σύνολο είναι το σύνολο χωρίς στοιχεία και συμβολίζεται με  $\emptyset$ .

Σύνολα που δε μπορούν να οριστούν με καταγραφή όλων των στοιχείων τους χωρισμένων με κόμματα και μέσα σε αγκύλες είναι **άπειρα**, π.χ.  $N = \{0, 1, 2, \dots, \}$ . Σύνολα που δεν είναι άπειρα, είναι **πεπερασμένα**.

**Ορισμός 1.2.5.** Ένα σύνολο  $A$  είναι **υποσύνολο** ενός συνόλου  $B$ ,  $A \subseteq B$ , αν κάθε στοιχείο του  $A$  είναι και στοιχείο του  $B$ .

- Κάθε σύνολο είναι υποσύνολο του εαυτού του.
- Το  $\emptyset$  είναι υποσύνολο κάθε συνόλου.

**Ορισμός 1.2.6.** Αν το  $A$  είναι υποσύνολο του  $B$ , αλλά το  $A$  δεν είναι ίδιο με το  $B$ , τότε το  $A$  είναι **γνήσιο υποσύνολο** του  $B$ ,  $A \subset B$ . Το κενό σύνολο,  $\emptyset$ , είναι υποσύνολο κάθε συνόλου.

Για να δείξουμε ότι δύο σύνολα  $A$  και  $B$  είναι ίσα, μπορούμε να δείξουμε ότι  $A \subseteq B$  και  $B \subseteq A$ .

Υποθέτουμε πάντα την ύπαρξη ενός υπερσυνόλου που καλείται **σύμπαν** (universal set) και συμβολίζεται  $U$ , σε σχέση με το οποίο ορίζεται η πράξη της συμπλήρωσης (complement). Ένα σύνολο αποτελείται από έναν αριθμό στοιχείων, που μπορεί να είναι μηδέν, πεπερασμένος, μετρήσιμα άπειρος ή μη μετρήσιμα άπειρος, και ορίζεται με έναν από τους παρακάτω τρόπους:

- i. καταγραφή των στοιχείων του με τη μορφή λίστας, αν είναι πεπερασμένο, π.χ.  $\{a, e, i, o, u\}$
- ii. περιγραφή ενός μηχανισμού παραγωγής ή αναγνώρισης των στοιχείων του,
- iii. παρουσίαση μιας σχέσης ορισμού του, π.χ.  $S = \{x \in A \mid Q(x)\}$ , που σημαίνει ότι το σύνολο  $S$  αποτελείται από όλα τα  $x$  που ανήκουν στο  $A$  έτσι ώστε η συνάρτηση  $Q$  με είσοδο  $x$  να είναι αληθής. Ο όρος  $Q(x)$  είναι ένα κατηγορήμα, μια συνθήκη, δηλαδή, που περιγράφει τι πρέπει να ισχύει για το  $x$  προκειμένου αυτό να ανήκει στο σύνολο  $S$ , π.χ.  $G = \{x : x \in N \text{ και } x \text{ είναι μεγαλύτερο από το } 2\}$ .

Έτσι, αναφέρουμε τα ακόλουθα (γνωστά) σύνολα.

- Το σύνολο των φυσικών αριθμών (Natural Numbers)  $N$  που συμβολίζεται ως:

$$\{1, 2, \dots, \}.$$

- Το σύνολο των ακεραίων αριθμών (Integer Numbers)  $\mathbb{Z}$  που συμβολίζεται ως:

$$\{\dots, -2, -1, 0, 1, 2, \dots\}.$$

Παρακάτω ορίζονται πράξεις με σύνολα.

- Ένωση (union) των συνόλων  $A$  και  $B$  ( $A \cup B$ ): ένα νέο σύνολο που περιλαμβάνει όλα τα στοιχεία του  $A$  και όλα τα στοιχεία του  $B$ .

**Συμβολίζουμε:**  $A \cup B = \{x \text{ ανήκει στο } U | x \text{ ανήκει στο } A \text{ ή } x \text{ ανήκει στο } B\}.$

- Τομή (intersection) των συνόλων  $A$  και  $B$  ( $A \cap B$ ): ένα νέο σύνολο που περιέχει μόνο τα κοινά στοιχεία των συνόλων  $A$  και  $B$ .

**Συμβολίζουμε:**  $A \cap B = \{x \text{ ανήκει στο } U | x \text{ ανήκει στο } A \text{ και } x \text{ ανήκει στο } B\}.$

- Διαφορά (difference) των συνόλων  $A$  και  $B$  ( $A - B$ ): ένα νέο σύνολο που περιέχει τα στοιχεία του  $A$  που δεν ανήκουν στο  $B$ .

**Συμβολίζουμε:**  $A - B = \{x \text{ ανήκει στο } U | x \text{ ανήκει στο } A \text{ και } x \text{ δεν ανήκει στο } B\}.$

- Συμπλήρωμα (complement) ενός συνόλου  $A$  ( $\bar{A}$  ή  $\Gamma$ ): ένα νέο σύνολο που περιέχει ό,τι δεν ανήκει στο  $A$ . Γίνεται σχεδόν πάντα ο συσχετισμός με ένα σύμπαν (universal set)  $U$  που υποτίθεται ότι περιέχει “τα πάντα” οπότε το  $\bar{A}$  είναι ουσιαστικά το  $U - A$ .

**Συμβολίζουμε:**  $S' = \{x \text{ ανήκει στο } U | x \text{ δεν ανήκει στο } S\}.$

Οι παραπάνω πράξεις έχουν τις εξής ιδιότητες:

<b>Αυτοδυναμία</b>	$A \cap A = A$ $A \cup A = A$
<b>Αντιμεταθετική</b>	$A \cap B = B \cap A$ $A \cup B = B \cup A$
<b>Προσεταιριστική</b>	$(A \cap B) \cap C = A \cap (B \cap C)$ $(A \cup B) \cup C = A \cup (B \cup C)$
<b>Επιμεριστική</b>	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
<b>Απορροφητικό Στοιχείο</b>	$A \cap (A \cup B) = A$ $A \cup (A \cap B) = A$
<b>Νόμοι του De Morgan</b>	$A - (B \cup C) = (A - B) \cap (A - C)$ $A - (B \cap C) = (A - B) \cup (A - C)$
<b>Νόμοι De Morgan</b>	$\overline{A \cup B} = \bar{A} \cap \bar{B}$ $\overline{A \cap B} = \bar{A} \cup \bar{B}$

**Πληθάριθμος (cardinality)** ενός συνόλου  $A$  ονομάζεται ο αριθμός των στοιχείων του  $A$  και συμβολίζεται  $|A|$ .

**Δυναμοσύνολο (powerset)** ενός συνόλου  $A$  ονομάζεται το σύνολο των υποσυνόλων του  $A$  και συμβολίζεται  $2^{|A|}$ . Δηλαδή ένα σύνολο με  $n$  στοιχεία (elements) έχει ένα δυναμοσύνολο με  $2^n$  στοιχεία.

Δύο σύνολα είναι **ξένα** μεταξύ τους (**disjoint**) αν δεν έχουν κανένα κοινό στοιχείο, αν δηλαδή  $A \cap B = \emptyset$ . Αν τα σύνολα  $A$  και  $B$  είναι ξένα μεταξύ τους, τότε ισχύουν τα εξής:

1.  $|A \cup B| = |A| + |B|$ .
2.  $|A \cap B| = 0$ .
3.  $|A \times B| = |A| * |B|$ .
4.  $|P(A)| = 2^{|A|}$

**Ορισμός 1.2.7.** Διαμέριση ενός μη κενού συνόλου  $A$ , είναι ένα υποσύνολο  $\Pi$  του  $2^{|A|}$  τέτοιο ώστε:

- το κενό,  $\emptyset$ , δεν είναι στοιχείο του  $\Pi$
- κάθε στοιχείο του  $A$  ανήκει σε ένα μόνο σύνολο που ανήκει στο  $\Pi$

**Ορισμός 1.2.8.** Το  $\Pi$  είναι διαμέριση ενός μη κενού συνόλου  $A$ , αν το  $\Pi$  είναι σύνολο υποσυνόλων του  $A$  τέτοιο ώστε:

1. Κάθε στοιχείο του  $\Pi$  είναι μη κενό
2. Τα μέλη του  $\Pi$  είναι ξένα μεταξύ τους
3.  $\cup_{\Pi} = A$

**Παράδειγμα 1.2.4.**  $\{\{a, b\}, \{c\}, \{d\}\}$  είναι μια διαμέριση του  $\{a, b, c, d\}$  αλλά  $\{\{a, b, c\}, \{c, d\}\}$  δεν είναι.

### 1.3 Πεπερασμένα και Άπειρα Σύνολα

**Ορισμός 1.3.1.** Δύο σύνολα  $A$  και  $B$  είναι **ισοδύναμα** αν υπάρχει αμφιμονοσήμαντη αντιστοιχία  $f : A \rightarrow B$ . Αν υπάρχει μια αμφιμονοσήμαντη αντιστοιχία  $f : A \rightarrow B$  τότε υπάρχει και μία άλλη  $f^{-1} : B \rightarrow A$ .

**Ορισμός 1.3.2.** Ένα σύνολο είναι πεπερασμένο αν είναι ισοδύναμο με το  $\{1, 2, 3, \dots, n\}$  για  $n \in \mathbb{N}$ . Για  $n = 0$ ,  $\{1, \dots, n\}$  είναι το  $\emptyset \Rightarrow \emptyset$  πεπερασμένο και ισοδύναμο με τον εαυτό του.

**Ορισμός 1.3.3.** Ένα σύνολο λέγεται **μετρήσιμο άπειρο** είναι ισοδύναμο του  $\mathbb{N}$  και **μετρήσιμο** αν είναι πεπερασμένο ή μετρήσιμο άπειρο. Ένα σύνολο που δεν είναι μετρήσιμο χαρακτηρίζεται ως **μη μετρήσιμο**

## 1.4 Συναρτήσεις και σχέσεις

Το διατεταγμένο ζεύγος των στοιχείων  $a$  και  $b$  γράφεται ως  $(a, b)$ .

- Έχει σημασία η σειρά: το διατεταγμένο ζεύγος  $(a, b)$  είναι διαφορετικό από το  $(b, a)$ .
- Τα δύο στοιχεία ενός διατεταγμένου ζεύγους δεν είναι ανάγκη να είναι διαφορετικά μεταξύ τους: το  $(a, a)$  είναι ένα διατεταγμένο ζεύγος.
- Δύο διατεταγμένα ζεύγη  $(a, b)$  και  $(c, d)$  είναι ίσα μόνο αν  $a = c$  και  $b = d$ .

**Ορισμός 1.4.1.** Το **καρτεσιανό γινόμενο** δύο συνόλων  $A$  και  $B$  συμβολίζεται με  $A \times B$  και είναι το σύνολο όλων των διατεταγμένων ζευγών  $(a, b)$  με  $a \in A$  και  $b \in B$ .

**Παράδειγμα 1.4.1.**  $\{1, 3, 9\} \times \{b, c, d\} = \{(1, b), (1, c), (1, d), (3, b), (3, c), (3, d), (9, b), (9, c), (9, d)\}$

**Ορισμός 1.4.2.** Μια **δυαδική σχέση** ανάμεσα σε δύο σύνολα  $A$  και  $B$  είναι ένα υποσύνολο του  $A \times B$ , π.χ.  $\{(1, b), (1, c), (3, d), (9, d)\}$  είναι μια δυαδική σχέση ανάμεσα στα σύνολα  $\{1, 3, 9\}$  και  $\{b, c, d\}$ .

- Μια σχέση  $R \subseteq A \times A$  είναι **ανακλαστική**  $(a, a) \in R$  για κάθε  $a \in A$ .
- Μια σχέση είναι **συμμετρική**  $(b, a) \in R$  όσο  $(a, b) \in R$ .
- Μια σχέση είναι **αντισυμμετρική** όποτε  $(a, b) \in R$  και τα  $a$  και  $b$  είναι διαφορετικά, τότε  $(b, a) \notin R$ .
- Μια σχέση είναι **μεταβατική** οποτεδήποτε  $(a, b) \in R$  και  $(b, c) \in R$ , τότε και  $(a, c) \in R$ .

Μια συνάρτηση (function) είναι ένα μαθηματικό αντικείμενο που ορίζει μια σχέση εισόδου - εξόδου. Πρόκειται για έναν ειδικό τύπο σχέσης όπου το δεύτερο στοιχείο κάθε ζεύγους καθορίζεται με μοναδικό τρόπο από το πρώτο στοιχείο του ζεύγους.

**Ορισμός 1.4.3.** Μια **συνάρτηση** από ένα σύνολο  $A$  σε ένα σύνολο  $B$  είναι μια δυαδική σχέση  $R$  πάνω στα  $A$  και  $B$  με την εξής ιδιότητα:  $\forall a \in A, \exists$  ακριβώς ένα διατεταγμένο ζεύγος στο  $R$  με πρώτο στοιχείο το  $a$ .

Γράφουμε  $f : A \rightarrow B$  για να δείξουμε ότι η  $f$  είναι μια συνάρτηση από το  $A$  στο  $B$ . Ονομάζουμε το  $A$  **πεδίο ορισμού** της  $f$ . Αν το  $a$  είναι ένα οποιοδήποτε στοιχείο του  $A$ , συμβολίζουμε με  $f(a)$  εκείνο το στοιχείο του  $B$  τέτοιο ώστε  $(a, b) \in f$ . Αφού η  $f$  είναι συνάρτηση, υπάρχει ένα μόνο στοιχείο με την ιδιότητα αυτή  $b \in B$  κι έτσι το  $f(a)$  συμβολίζει ένα μοναδικό αντικείμενο που ονομάζεται **εικόνα** του  $a$  ως προς την  $f$ . Το **πεδίο τιμών** της  $f$  είναι η εικόνα του πεδίου ορισμού της.

**Ορισμός 1.4.4.** Μια συνάρτηση  $f : A \rightarrow B$  είναι *ένα-προς-ένα* για κάθε δύο διαφορετικά στοιχεία  $a, a' \in A$ ,  $f(a) \neq f(a')$ .

**Ορισμός 1.4.5.** Μια συνάρτηση  $f : A \rightarrow B$  είναι *επί του*  $B$  αν κάθε στοιχείο του  $B$  είναι εικόνα μέσω της  $f$  κάποιου στοιχείου του  $A$ .

**Ορισμός 1.4.6.** Μια απεικόνιση  $f : A \rightarrow B$  είναι μια αμφιμονοσήμαντη αντιστοιχία ανάμεσα στα  $A$  και  $B$  αν είναι 1-1 και επί του  $B$ .

**Ορισμός 1.4.7.** Κάθε δυαδική σχέση  $R \subseteq A \times B$  έχει μια *αντίστροφη*  $R^{-1} \subseteq B \times A$  που ορίζεται σαν

$$(b, a) \in R^{-1} \text{ αν και μόνον αν } (a, b) \in R.$$

**ΠΡΟΣΟΧΗ!** Η αντίστροφη μιας συνάρτησης δεν είναι κατ' ανάγκη συνάρτηση. Μια συνάρτηση  $f : A \rightarrow B$  δεν έχει αντίστροφη αν

$$\exists b \in B : \nexists a \in A \text{ με } f(a) = b.$$

Αν  $f : A \rightarrow B$  είναι αμφιμονοσήμαντη αντιστοιχία και η  $f^{-1}$  είναι αμφιμονοσήμαντη αντιστοιχία ανάμεσα στα  $B$  και  $A$ .

$$f^{-1}(f(a)) = a, \forall a \in A,$$

$$f(f^{-1}(b)) = b, \forall b \in B.$$

**Ορισμός 1.4.8.** Μια σχέση  $\mu R \subseteq A \times B$  είναι *ανακλαστική* αν  $(a, a) \in R, \forall a \in A$ .

**Ορισμός 1.4.9.** Μια σχέση  $R \subseteq A \times B$  είναι *συμμετρική* αν  $(a, b) \in R$ , όσο  $(b, a) \in R$ .

**Ορισμός 1.4.10.** Μια σχέση  $R \subseteq A \times B$  είναι *αντισυμμετρική* αν όποτε  $(a, b) \in R$ , και  $a \neq b$  τότε  $(b, a) \notin R$ .

**Ορισμός 1.4.11.** Μια σχέση  $R$  είναι *μεταβατική* αν οποτεδήποτε  $(a, b) \in R$ , και  $(b, c) \in R$  τότε και  $(a, c) \in R$ .

Όταν το πεδίο ορισμού μιας συνάρτησης  $f$  είναι της μορφής  $A_1 \times \dots \times A_k$  για κάποια σύνολα  $A_1, \dots, A_k$ , η είσοδος της συνάρτησης  $f$  είναι μια  $k$ -άδα ( $k$ -tuple) της οποίας τα στοιχεία καλούνται ορίσματα (arguments) της  $f$ . Μια συνάρτηση με  $k$  ορίσματα καλείται  $k$ -ική συνάρτηση ( $k$ -ary function). Στην περίπτωση που  $k = 1$ , η  $f$  έχει ένα μόνο όρισμα και καλείται μοναδιαία συνάρτηση (unary function). Στην περίπτωση που  $k = 2$ , η  $f$  καλείται δυαδική συνάρτηση (binary function).

**Ορισμός 1.4.12.** Μια σχέση  $R$  είναι *σχέση ισοδυναμίας* αν είναι ανακλαστική, συμμετρική και μεταβατική. Οι σχέσεις ισοδυναμίας έχουν εξαιρετική σημασία και χωρίζουν ένα σύνολο σε ξένα μεταξύ τους τμήματα, που αναφέρονται σαν κλάσεις ισοδυναμίας. Αν το  $a$  είναι ένα στοιχείο του συνόλου βάσης  $A$  πάνω στο οποίο έχει οριστεί η σχέση, τότε  $[a]_R$  είναι η κλάση ισοδυναμίας του  $a$  κάτω από τη σχέση  $R$ , και συμβολίζουμε  $[a]_R = \{b \text{ ανήκει στο } A \mid aRb\}$ . Τέλος, για κάθε  $a, b$  που ανήκουν στο  $A$ , είναι είτε  $[a] = [b]$  είτε  $[a] \cap [b] = \emptyset$ .

## 1.5 Γραφήματα, συμβολοσειρές, και γλώσσες

### 1.5.1 Γραφήματα

Ένας μη κατευθυνόμενος γράφος (undirected graph) είναι, απλά, ένα σύνολο σημείων και ακμών που συνδέουν κάποια από τα σημεία αυτά. Συμβολίζεται συνήθως  $G = (V, E)$  όπου:

- $V$  : σύνολο κορυφών (vertices) ή κόμβων και,
- $E$  : σύνολο ακμών (edges).

Κάθε ακμή ορίζεται από ένα ζεύγος κορυφών. Αν οι ακμές σε ένα γράφο είναι κατευθυνόμενες, ο γράφος ονομάζεται κατευθυνόμενος γράφος (digraph που είναι σύντμηση του όρου “directed graph”). Ένας περίπατος (walk) είναι μια ακολουθία ακμών όπου η τελική κορυφή της μιας είναι αρχική κορυφή της επόμενης, π.χ.  $(a, e), (e, i), (i, o), (o, u)$ . Ο αριθμός των ακμών που πρόσκεινται σε έναν κόμβο είναι ο βαθμός (degree) του κόμβου αυτού. Στην περίπτωση κατευθυνόμενων γράφων, ορίζεται σαν έσω-βαθμός (in-degree) ενός κόμβου το πλήθος των εισερχόμενων σε αυτόν ακμών, ενώ σαν έξω-βαθμός (out-degree) ενός κόμβου το πλήθος των εξερχόμενων από αυτόν ακμών. Ένα μονοπάτι (path) σε ένα γράφο είναι μια ακολουθία κόμβων που συνδέονται με ακμές. Όταν σε ένα μονοπάτι κάθε κορυφή εμφανίζεται μια φορά, το μονοπάτι ονομάζεται απλό (simple path). Ένα μονοπάτι είναι κύκλος (cycle) αν η αφετηρία και ο προορισμός του είναι ο ίδιος κόμβος. Ένας κύκλος είναι απλός (simple cycle) όταν δεν περιέχει επαναλαμβανόμενους κόμβους εκτός από τον πρώτο και τον τελευταίο. Δεδομένου ενός γράφου  $G$ , ένας γράφος  $H$  καλείται υπογράφος (subgraph) του  $G$  αν το σύνολο των κόμβων του  $H$  είναι υποσύνολο του συνόλου των κόμβων του  $G$ . Ένας γράφος είναι συνεκτικός (connected) αν υπάρχει μονοπάτι μεταξύ δύο οποιωνδήποτε κόμβων του. Ένας κατευθυνόμενος γράφος είναι ισχυρά συνεκτικός (strongly connected) αν μεταξύ οποιωνδήποτε δύο κόμβων του υπάρχει κατευθυνόμενη ακμή. Ένας γράφος είναι ένα δένδρο (tree) αν είναι συνεκτικός και δεν περιέχει απλούς κύκλους. Οι κόμβοι ενός δένδρου που έχουν βαθμό 1 καλούνται φύλλα (leaves) του δένδρου.

### 1.5.2 Συμβολοσειρές (Strings) και Γλώσσες (Languages)

**Ορισμός 1.5.1.** Ένα αλφάβητο (alphabet), έστω  $\Sigma$ , είναι ένα πεπερασμένο, μη κενό σύνολο συμβόλων,  $\Sigma = \{a_1, a_2, \dots, a_k\}$ .

Τα σύμβολα ενδέχεται να αποτελούνται και από περισσότερους από έναν χαρακτήρες π.χ. η δεσμευμένη λέξη *while* της Pascal θεωρείται ένα απλό σύμβολο.

**Ορισμός 1.5.2.** Αν  $\Sigma$  είναι ένα αλφάβητο και  $L$  είναι ένα υποσύνολο του  $\Sigma^*$ , τότε το σύνολο  $L$  λέγεται γλώσσα ορισμένη με βάση το  $\Sigma$ . Κάθε στοιχείο της  $L$  αναφέρεται σαν πρόταση (sentence) ή λέξη (word) ή συμβολοσειρά (string) της γλώσσας.

**Ορισμός 1.5.3.** Μια συμβολοσειρά (*string*) είναι μια πεπερασμένη ακολουθία συμβόλων ενός αλφαβήτου  $\Sigma$ . Το μήκος μιας συμβολοσειράς  $s$  συμβολίζεται  $|s|$  και είναι το πλήθος των συμβόλων της.

Τυπικά, μια συμβολοσειρά μήκους  $k$  μπορεί να θεωρηθεί σα μια συνάρτηση από το σύνολο  $\{1, 2, \dots, k\}$  στο  $\Sigma$ . Όταν μια συμβολοσειρά έχει μήκος 0 ονομάζεται κενή συμβολοσειρά (empty string) συμβολίζεται συνήθως με το γράμμα  $\epsilon$  και μπορεί να θεωρηθεί σαν τη κενή συνάρτηση (ένα κενό σύνολο διατεταγμένων ζευγών). Με  $\Sigma^*$  συμβολίζουμε το σύνολο των συμβολοσειρών ενός αλφαβήτου, συμπεριλαμβανομένης και της κενής συμβολοσειράς  $\epsilon$ , ενώ με  $\Sigma^+$  το σύνολο όλων των μη κενών συμβολοσειρών ενός αλφαβήτου, είναι δηλαδή:  $\Sigma^+ = \Sigma^* - \{\epsilon\}$ . Μια γλώσσα είναι ουσιαστικά ένα υποσύνολο του  $\Sigma^*$ . Τα παραπάνω γράφονται πιο τυπικά ως εξής:

$$\Sigma^* = \{s \mid s \text{ μια συμβολοσειρά του } \Sigma\}$$

είναι το σύνολο όλων των συμβολοσειρών του αλφαβήτου  $\Sigma$ .

$$\Sigma^* = \{s = a_1 a_2 \dots a_k \mid k = 0 \text{ και για κάθε } i, 0 < i \leq k, a_i \text{ ανήκει στο } \Sigma\}.$$

### Παραδείγματα

1.  $\Sigma = \{a\}$
2.  $\Sigma = \{a, b, c\}$
3.  $\Sigma = \{0, 1\}$

Μια συμβολοσειρά είναι μια πεπερασμένη ακολουθία συμβόλων. Είναι ευκολότερο να ορίζει κανείς τις συμβολοσειρές σα συναρτήσεις. Για κάθε ακέραιο  $n \geq 1$  συμβολίζουμε  $[n] = \{1, 2, 3, \dots, n\}$  και για  $n = 0$ , συμβολίζουμε  $[0] = \emptyset$ .

**Ορισμός 1.5.4.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , μια συμβολοσειρά (*string*) ορισμένη στο αλφάβητο  $\Sigma$  (ή απλά μια συμβολοσειρά) μήκους  $n$  είναι κάθε συνάρτηση  $u : [n] \rightarrow \Sigma$ .

Ο ακέραιος  $n$  είναι το μήκος της συμβολοσειράς  $u$ , και συμβολίζεται  $|u|$ . Στην περίπτωση που  $n = 0$ , η ειδική συμβολοσειρά  $u : [0] \rightarrow \Sigma$  έχει μήκος 0, καλείται “κενή συμβολοσειρά” και συμβολίζεται  $\epsilon$ .

Δεδομένης μιας συμβολοσειράς  $u : [n] \rightarrow \Sigma$  μήκους  $n - 1$ , συμβολίζουμε με  $u(i)$  το  $i$ -στό σύμβολο στη συμβολοσειρά  $u$ . Για λόγους απλότητας, συμβολίζουμε τη συμβολοσειρά  $u$  σαν  $u = u_1 u_2 \dots u_n$  με κάθε  $u_i \in \Sigma$ .

Για παράδειγμα, αν  $\Sigma = \{a, b\}$  και  $u : [3] \rightarrow \Sigma$  ορίζεται έτσι ώστε  $u(1) = a$ ,  $u(2) = b$ , και  $u(3) = a$ , γράφουμε

$$u = aba.$$



Συμβολοσειρές μήκους 1 είναι συναρτήσεις  $u : [1] \rightarrow \Sigma$  και προκύπτουν απλά επιλέγοντας κάποιο στοιχείο  $u(1) = a_i \in \Sigma$ . Επομένως, κάθε σύμβολο  $a_i \in \Sigma$  ορίζεται σαν μια συμβολοσειρά μεγέθους 1. Το σύνολο όλων των συμβολοσειρών ενός αλφαβήτου  $\Sigma$ , συμπεριλαμβανομένης και της κενής συμβολοσειράς, συμβολίζεται σαν  $\Sigma^*$ .

Παρατηρούμε ότι όταν  $\Sigma = \emptyset$ , τότε είναι

$$\emptyset^* = \{\epsilon\}.$$

Όταν  $\Sigma \neq \emptyset$ , το σύνολο  $\Sigma^*$  είναι μετρήσιμα άπειρο. Οι συμβολοσειρές είναι δυνατόν να παρατεθούν (concatenated).

**Ορισμός 1.5.5.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , και δυο οποιονδήποτε συμβολοσειρών  $u : [m] \rightarrow \Sigma$  και  $v : [n] \rightarrow \Sigma$ , η παράθεση (concatenation)  $u \cdot v$  (που συμβολίζεται και  $uv$ ) της  $u$  και της  $v$  είναι η συμβολοσειρά  $uv : [m+n] \rightarrow \Sigma$ , που ορίζεται ως εξής:

$$uv(i) = \begin{cases} u(i), & \text{αν } 1 \leq i \leq m \\ v(i-m), & \text{αν } m+1 \leq i \leq m+n \end{cases}$$

Ειδικότερα,  $u\epsilon = \epsilon u = u$ . Εύκολα φαίνεται ότι

$$u(vw) = (uv)w.$$

Επομένως, η παράθεση είναι μια δυαδική πράξη στο σύνολο  $\Sigma^*$  που είναι προσεταιριστική και έχει την  $\epsilon$  σαν ουδέτερο στοιχείο.

Γενικά, πάντως, είναι  $uv \neq vu$ , για παράδειγμα για  $u = a$  και  $v = b$ .

Δεδομένης μιας συμβολοσειράς  $u \in \Sigma^*$  και  $n \geq 0$ , η  $u^n$  ορίζεται ως εξής:

$$u^n = \begin{cases} \epsilon, & \text{αν } n = 0 \\ u^{n-1}u, & \text{αν } n \geq 1 \end{cases}$$

Προφανώς, είναι  $u^1 u = u$  και εύκολα μπορεί ναδειχθεί ότι ισχύει  $u^n u = uu^n$  για κάθε  $n \geq 0$ .

**Ορισμός 1.5.6.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , και δύο οποιωνδήποτε συμβολοσειρών  $u, v \in \Sigma^*$  ορίζονται τα ακόλουθα:

Η συμβολοσειρά  $u$  είναι πρόθεμα (prefix) της  $v$  αν και μόνον αν υπάρχει συμβολοσειρά  $y \in \Sigma^*$  τέτοια ώστε

$$v = uy.$$

Η συμβολοσειρά  $u$  είναι ένα επίθεμα (κατάληξη - suffix) της  $v$  αν και μόνον αν υπάρχει συμβολοσειρά  $x \in \Sigma^*$  τέτοια ώστε

$$v = xu.$$

Η συμβολοσειρά  $u$  είναι υποσυμβολοσειρά (substring) μιας συμβολοσειράς  $v$  αν και μόνον αν υπάρχουν συμβολοσειρές  $x, y \in \Sigma^*$  τέτοιες ώστε

$$v = xuy.$$

Λέμε ότι η συμβολοσειρά  $u$  είναι γνήσιο (proper) πρόθεμα (επίθεμα, υποσυμβολοσειρά) μιας συμβολοσειράς  $v$  αν και μόνον αν η  $u$  είναι πρόθεμα (επίθεμα, υποσυμβολοσειρά) της  $v$  και  $u \neq v$ .

Υπενθυμίζουμε ότι μια μερική διάταξη (partial ordering)  $\leq$  σε ένα σύνολο  $S$  είναι μια δυαδική σχέση (binary relation)  $\leq \subseteq S \times S$  η οποία είναι ανακλαστική (reflexive), μεταβατική (transitive) και αντισυμμετρική (antisymmetric). Οι έννοιες πρόθεμα, επίθεμα, υποσυμβολοσειρά ορίζουν δυαδικές σχέσεις στο  $\Sigma^*$  με προφανή τρόπο. Μπορεί να αποδειχθεί ότι αυτές οι σχέσεις είναι μερικές διατάξεις. Μια άλλη σημαντική διάταξη στις συμβολοσειρές είναι η λεξικογραφική (lexicographic) ή λεξικό (dictionary).

**Ορισμός 1.5.7.** Δεδομένου ενός αλφαβήτου  $\Sigma = \{a_1, \dots, a_k\}$  που θεωρούμε ότι είναι πλήρως διατεταγμένο έτσι ώστε  $a_1 \leq a_2 \leq \dots \leq a_k$  και δύο συμβολοσειρών  $u, v \in \Sigma^*$ , η λεξικογραφική διάταξη  $\leq$  ορίζεται ως εξής:

$$u \leq v = \begin{cases} \text{αν } v = uy, \text{ για κάποιο } y \in \Sigma^*, \text{ ή} \\ \text{αν } u = xa_iy, v = xa_iz, \\ \text{και } a_i < a_j, \text{ για κάποια } x, y, z \in \Sigma^*. \end{cases}$$

Η λεξικογραφική διάταξη είναι στην πραγματικότητα μια μερική διάταξη και μάλιστα μια γνήσια διάταξη, γεγονός που σημαίνει ότι για οποιεσδήποτε συμβολοσειρές  $u, v \in \Sigma^*$ , είτε  $u \leq v$ , ή  $v \leq u$ . Η αντίστροφη συμβολοσειρά  $w^R$  μιας συμβολοσειράς  $w$  ορίζεται επαγωγικά ως εξής:

$$\epsilon^R = \epsilon,$$

$$(ua)^R = au^R,$$

όπου  $a \in \Sigma$  και  $u \in \Sigma^*$ .

Εύκολα αποδεικνύεται ότι  $(uv)^R = v^R u^R$ . Επομένως,

$$(u_1, \dots, u_n)^R = u_n^R, \dots, u_1^R.$$

Όταν  $u_i \in \Sigma$ , είναι

$$(u_1, \dots, u_n)^R = u_n, \dots, u_1.$$

5αλφάβητο  $\Sigma$  (ή απλά μια γλώσσα) είναι κάθε υποσύνολο  $L$

## 1.6 Δυαδική λογική και επισκόπηση όρων

Η λογική πραγματεύεται με σχέσεις (κανόνες) και τα σύνολα αληθείας τους. Μια σχέση μπορεί να έχει μεταβλητές δεσμευμένες (bound) ή ελεύθερες (free). Η δέσμευση ακολουθεί τους κανόνες εμβέλειας με την έννοια που συναντώνται στις γλώσσες προγραμματισμού, και με την επιπλέον ύπαρξη δύο ποσοδεικτών (quantifiers), του “για κάθε” - καθολικού ποσοδείκτη (forall) που συμβολίζεται “ $\forall$ ” και του “υπάρχει” - υπαρξιακού ποσοδείκτη (exists) που συμβολίζεται “ $\exists$ ”. Η διάταξη των ποσοδεικτών έχει σημασία: οι σχέσεις  $S_1$  και  $S_2$  είναι διαφορετικές. Έστω  $H = \{ \text{άνθρωποι, ζωντανοί και νεκροί} \}$  και έστω  $m : H \rightarrow H$  με  $m(h) =$  η βιολογική μητέρα του  $h$ .

$S_1$  : Για κάθε  $x$  που ανήκει στο  $H$  υπάρχει  $y$  που ανήκει στο  $H$  τέτοιο ώστε  $y = m(x)$ .

$S_2$  : Υπάρχει  $y$  που ανήκει στο  $H$  τέτοιο ώστε για κάθε  $x$  που ανήκει στο  $H$  να είναι  $y = m(x)$ .

Η σχέση  $S_1$  δηλώνει ότι κάθε άνθρωπος έχει μια (μοναδική) μητέρα, ενώ η σχέση  $S_2$  δηλώνει ότι υπάρχουν άνθρωποι που είναι “μητέρες” κάθε ανθρώπου ακόμα και του εαυτού τους. Οι σχέσεις σχετίζονται με τα σύνολα αλήθειας τους. Για την  $S_1$ , το σύνολο αλήθειας είναι  $T(S_1) = \{(x, m(x)) | x \text{ ανήκει στο } H\}$ . Για την  $S_2$  το σύνολο αλήθειας είναι το κενό. Μια σχέση  $A$  ικανοποιείται από μια ανάθεση, αν η τελευταία ανήκει στο σύνολο αλήθειας της σχέσης, δηλαδή, αν απόδοση των τιμών της ανάθεσης στις μεταβλητές της σχέσης κάνει τη σχέση αληθή.

## 1.7 Ορισμοί, θεωρήματα, και αποδείξεις

Οι ορισμοί, τα θεωρήματα και οι αποδείξεις είναι θεμελιώδεις οντότητες για κάθε μαθηματικό αντικείμενο.

Οι ορισμοί περιγράφουν τα αντικείμενα και τις έννοιες που χρησιμοποιούμε. Η ακρίβεια είναι μια βασική ιδιότητα που πρέπει να χαρακτηρίζει έναν ορισμό, δηλαδή όταν ορίζεται ένα αντικείμενο να πρέπει ρητά να διευκρινίζεται τι συνθέτει αυτό το αντικείμενο και τι όχι. Μια μαθηματική πρόταση (mathematical statement) δηλώνει το γεγονός ότι ένα αντικείμενο έχει μια συγκεκριμένη ιδιότητα. Μια πρόταση μπορεί να είναι αληθής ή ψευδής, αλλά επίσης, επιβάλλεται να είναι ακριβής. Απόδειξη (proof) είναι μια αλληλουχία λογικών επιχειρημάτων που έχει ως σκοπό να πείσει ότι μια πρόταση είναι αληθής. Μια μαθηματική πρόταση που αποδείχτηκε αληθής αποτελεί ένα θεώρημα (theorem). Μια πρόταση (που έχει αποδειχτεί αληθής) που έχει ενδιαφέρον γιατί βοηθά στην απόδειξη άλλων σημαντικότερων καλείται λήμμα (lemma).

## 1.8 Αποδείξεις

Ως απόδειξη θα μπορούσαμε να ορίσουμε μια αρκετά λεπτομερή και λογική διαδικασία διατύπωσης μιας ακολουθίας επιχειρημάτων που έχει σκοπό να πείσει για την ορθότητα ή όχι μιας δήλωσης -

θέσης. Οι αποδείξεις παρουσιάζονται συνήθως μέσω μιας ιεραρχίας βημάτων πλήρως τεκμηριωμένων. Οι τυπικές αποδείξεις (formal proofs) δομούνται με βάση αξιώματα (αληθείς προτάσεις που δεν απαιτούν απόδειξη) και κανόνες. Βασικό σημείο στην εκάστοτε αποδεικτική διαδικασία είναι η σαφήνεια στη διατύπωση τόσο της θέσης που επιθυμούμε να αποδείξουμε, όσο και των υποθέσεων που λαμβάνουμε υπόψη. Γενικά οι θέσεις που επιθυμούμε να αποδείξουμε διατυπώνονται με τη μορφή  $P(x)$  ή  $Q(y)$ .

**Παράδειγμα 1.8.1.** Για οποιαδήποτε δύο σύνολα  $A$  και  $B$ , είναι  $\overline{A \cup B} = \overline{A} \cap \overline{B}$ . (Νόμος De Morgan)

*Απόδειξη.* Το παραπάνω θεώρημα ισχυρίζεται ότι δύο σύνολα, τα  $\overline{A \cup B}$  και  $\overline{A} \cap \overline{B}$  είναι ίσα. Θα δείξουμε ότι κάθε στοιχείο του ενός συνόλου είναι επίσης και στοιχείο του άλλου. Έστω  $x \in \overline{A \cup B}$ . Τότε  $x \notin A \cup B$  με βάση τον ορισμό του συμπληρωματικού ενός συνόλου. Επομένως, το στοιχείο  $x$  δεν ανήκει ούτε στο σύνολο  $A$  ούτε στο σύνολο  $B$ , με βάση τον ορισμό της ένωσης δύο συνόλων. Δηλαδή, ισχύει  $x \in \overline{A}$  και  $x \in \overline{B}$ , και με βάση τον ορισμό της τομής δύο συνόλων είναι  $x \in \overline{A} \cap \overline{B}$ .

Για την απόδειξη του αντίστροφου, υποθέτουμε ότι  $x \in \overline{A} \cap \overline{B}$ . Τότε το  $x$  είναι στοιχείο του συνόλου  $\overline{A}$  και του συνόλου  $\overline{B}$ . Άρα, είναι  $x \notin A$  και  $x \notin B$  και επομένως το  $x$  δεν ανήκει ούτε στην ένωση των δύο αυτών συνόλων. Κατά συνέπεια, το στοιχείο  $x$  ανήκει στο συμπληρωματικό σύνολο της ένωσης των δύο προηγούμενων συνόλων. Δηλαδή  $x \in \overline{A \cup B}$ , γεγονός που ολοκληρώνει την απόδειξη του θεωρήματος.  $\square$

Οι απευθείας αποδείξεις ξεκινούν με την υπόθεση ότι η πρόταση  $P(x)$  είναι αληθής, και καταλήγουν στο ότι η αλήθεια της πρότασης  $Q(x)$  προκύπτει ως επακόλουθο της υπόθεσης. Στις κατασκευαστικές αποδείξεις, αναθέτουμε συγκεκριμένες τιμές στη μεταβλητή  $x$  της πρότασης  $P(x)$  και κατόπιν δείχνουμε πώς να βρεθεί κάποιο  $y$  που να ικανοποιεί την πρόταση  $Q$ , με βάση τις αναθέσεις της  $x$ . Στις έμμεσες αποδείξεις υποθέτουμε ότι η  $Q$  είναι ψευδής, και δείχνουμε ότι τότε πρέπει και η  $P$  να είναι επίσης ψευδής. Δείχνουμε δηλαδή την αντιθετοαντιστροφή της αρχικής συνεπαγωγής:  $NOT(Q(y))$  ή  $NOT(P(x))$ . Η επαγωγή (induction) επιτρέπει την απόδειξη της ισχύος μιας θέσης για όλα τα στοιχεία ενός μετρήσιμου άπειρου συνόλου ξεκινώντας με την απόδειξη της αλήθειας της θέσης για μία ή περισσότερες βασικές περιπτώσεις. Στη συνέχεια διατυπώνεται η επαγωγική υπόθεση, μια πρόταση, δηλαδή, που υποτίθεται ότι είναι αληθής για κάποια στοιχεία του συνόλου. Τέλος, υπάρχει το επαγωγικό βήμα, κατά το οποίο δείχνουμε ότι με βάση την επαγωγική υπόθεση, η ίδια πρόταση που είναι αληθής για κάποιο στοιχείο είναι αληθής και για το επόμενο του.

### 1.8.1 Απόδειξη με Επαγωγή (Proof by induction)

**Διατύπωση:** Έστω ένα σύνολο φυσικών αριθμών τέτοιο ώστε:

1.  $0 \in A$ , και

2.  $\forall n \in N$ , αν  $\{0, 1, \dots, n\} \in A, \Rightarrow n + 1 \in A$ .

Τότε  $A = N$ .

Απλούστερα, κάθε σύνολο φυσικών αριθμών που περιέχει το μηδέν και έχει την ιδιότητα να περιέχει το  $n + 1$ , αν περιέχει τους άλλους αριθμούς έως το  $n$  συμπεριλαμβανομένου και του  $n$ , είναι στην πραγματικότητα το σύνολο όλων των φυσικών αριθμών. Στην πράξη η επαγωγή χρησιμοποιείται για να αποδείξουμε συμπεράσματα του τύπου: “Για κάθε φυσικό αριθμό  $n$ , ισχύει η ιδιότητα  $P$ ”.

**Εφαρμογή:** Η παραπάνω αρχή εφαρμόζεται στο σύνολο  $A = \{n : P \text{ αληθής για } n\}$  με τον ακόλουθο τρόπο:

1. **Βασικό Βήμα:** Δείχνουμε ότι  $0 \in A$ , δηλαδή ότι η  $P$  ισχύει για 0. Άρα, αποδεικνύουμε ότι κάποια υπόθεση ισχύει για την πρόταση  $P_1$  (που ονομάζεται και βάση της επαγωγής).
2. **Επαγωγική Υπόθεση:** Υποθέτουμε ότι για κάποιο ορισμένο αλλά αυθαίρετο  $n \geq 0$ , η  $P$  ισχύει για κάθε φυσικό αριθμό  $0, 1, \dots, n$ . Αποδεικνύουμε, δηλαδή, ότι αν η ίδια υπόθεση είναι αληθής για την πρόταση  $P_n$ , τότε είναι επίσης αληθής και για την πρόταση  $P_{n+1}$  (το βήμα αυτό καλείται και επαγωγική υπόθεση - inductive assumption).
3. **Επαγωγικό βήμα:** Δείχνουμε, χρησιμοποιώντας την επαγωγική υπόθεση, ότι η  $P$  ισχύει και για  $n + 1$ . Συμπεραίνουμε, δηλαδή, ότι η υπόθεση είναι αληθής για κάθε  $P$ . Κατά συνέπεια το  $A$  ισούται με το  $N$ , κι επομένως η  $P$  ισχύει για κάθε φυσικό αριθμό.

### 1.8.2 Αρχή του Περιστεριώνα (The Pigeonhole Principle)

Αν  $m$  περιστέρια τοποθετηθούν σε  $m$  περιστεροφωλιές (περιστεριώνες), υπάρχει άδεια περιστεροφωλιά αν και μόνον αν υπάρχει φωλιά με περισσότερα από δύο περιστέρια. Η Αρχή του Περιστεριώνα που είναι γνωστή και σαν Αρχή του Dirichlet δηλώνει την ισοδύναμη εξίσωση: Αν  $n > m$  περιστέρια τοποθετηθούν σε  $m$  φωλιές, υπάρχει μια φωλιά με περισσότερα από ένα περιστέρια.

**Θεώρημα 1.8.1.** Αν  $A$  και  $B$  μη κενά πεπερασμένα σύνολα και  $|A| > |B|$  τότε δεν υπάρχει συνάρτηση “ένα – προς – ένα” από το  $A$  στο  $B$ .

Απόδειξη. Για την απόδειξη χρησιμοποιούμε επαγωγή.

**Βασικό βήμα:** Υποθέτουμε ότι  $|B| = 1$ , ότι δηλαδή το  $B$  είναι το μονομελές σύνολο  $\{b\}$  και  $|A| >$

1. Αν  $f : A \rightarrow B$ , τότε υπάρχουν τουλάχιστον δύο διαφορετικά στοιχεία  $a_1, a_2 \in A$ , με  $f(a_1) = f(a_2) = b$ . Άρα η  $f$  δεν είναι “ένα – προς – ένα”.

**Επαγωγική Υπόθεση:** Υποθέτουμε ότι η  $f$  δεν είναι “ένα – προς – ένα”, αν  $f : A \rightarrow B, |A| > |B|$  και  $|B| \leq n$ , όπου  $n \geq 1$ .

**Επαγωγικό Βήμα:** Έστω

$$f : A \rightarrow B \text{ και } |A| > |B| = n + 1.$$

Επιλέγουμε ένα  $b \in B$ .

Αν  $|f^{-1}(b)| \leq 2$ , τότε η  $f$  δεν είναι “ένα – προς – ένα” γιατί δύο στοιχεία του  $A$  έχουν την ίδια εικόνα ως προς  $f$ , δηλαδή το  $b$ .

Αν  $|f^{-1}(b)| \leq 1$ , θεωρούμε τη συνάρτηση  $g : A - f^{-1}(b) \rightarrow B - \{b\}$  έτσι ώστε  $g(a) = f(a)$  για κάθε  $a \in A - f^{-1}(b)$ . Τότε

$$|A - f^{-1}(b)| \geq |A| - 1,$$

$$|B - b| = n < |A| - 1.$$

Τότε, σύμφωνα με την επαγωγική υπόθεση η  $g$  δεν είναι “ένα – προς – ένα”. Άρα

$$g(a_1) = g(a_2) \text{ για κάποια } a_1, a_2 \in A - f^{-1}(b)$$

που είναι διαφορετικά μεταξύ τους. Τότε όμως  $f(a_1) = f(a_2)$  και ούτε η  $f$  δεν είναι “ένα – προς – ένα”.  $\square$

Η Αρχή του Περιστεριώνα χρησιμοποιείται για να αποδειχτεί ότι κάποιες μη πεπερασμένες γλώσσες δεν είναι κανονικές. (**Προσοχή:** Κάθε πεπερασμένη γλώσσα είναι κανονική.) Εύκολα παρατηρείται ότι τα DFAs “δεν μπορούν να μετρήσουν”. Αυτό μπορεί να αποδειχθεί με χρήση της Αρχής του Περιστεριώνα.

**Παράδειγμα 1.8.2.** Η γλώσσα  $L = \{a^n b^n : n > 0\}$  δεν είναι κανονική.

**Παρατήρηση** Η μελέτη του παραδείγματος να γίνει σε συνδυασμό με το βιβλίο - κεφάλαιο των κανονικών γλωσσών.

**Απόδειξη.** Η απόδειξη γίνεται με εις άτοπον απαγωγή. Έστω ότι η  $L$  είναι κανονική. Υπάρχουν άπειρες τιμές που μπορεί να πάρει το  $n$  αλλά το αυτόματο που δέχεται τη γλώσσα  $L$ ,  $M(L)$ , έχει πεπερασμένο αριθμό καταστάσεων. Από την Αρχή του Περιστεριώνα, θα πρέπει να υπάρχουν διαφορετικές τιμές για  $i$  και  $j$  έτσι ώστε τα  $a^i$  και  $a^j$  να καταλήγουν στην ίδια κατάσταση. Από αυτή την κατάσταση, το  $b^i$  πρέπει να καταλήγει σε τελική κατάσταση, αφού η  $a^i b^i$  ανήκει στη  $L$ , και το  $b^j$  πρέπει να μη μεταβαίνει σε τελική κατάσταση, αφού η  $a^j b^j$  δεν ανήκει στη  $L$ . Όμως η επόμενη κατάσταση δεν μπορεί να είναι ταυτόχρονα τελική και μη τελική, επομένως φτάνουμε σε άτοπο. Άρα η υπόθεση, ότι δηλαδή η γλώσσα  $L$  είναι κανονική, είναι λανθασμένη.  $\square$

**Παράδειγμα 1.8.3.** Έστω  $a$  ένας άρρητος αριθμός. Τότε υπάρχουν άπειροι ρητοί αριθμοί  $r = \frac{p}{q}$  τέτοιοι ώστε  $|a - r| < \frac{1}{q^2}$ .

Απόδειξη. Έστω  $Q$  ένας θετικός ακέραιος αριθμός. Υποθέτουμε ότι  $a > 0$ . Θεωρούμε τα κλασματικά μέρη  $\{0\}, \{a\}, \{2a\}, \dots, \{Qa\}$  των πρώτων  $(Q+1)$  πολλαπλασίων του  $a$ . Από την Αρχή του Περιστεριώνα, δύο από αυτά πρέπει να βρίσκονται στο ίδιο  $Q$  διάστημα

$$[0, \frac{1}{Q}), [\frac{1}{Q}, \frac{2}{Q}), \dots, [\frac{Q-1}{Q}, 1),$$

όπου  $[A, B) = \{x : A \leq x < B\}$ . Με άλλα λόγια, υπάρχουν ακέραιοι  $s, q_1$ , και  $q_2$  τέτοιοι ώστε:

$$\{q_1 a\} \in [\frac{s}{Q}, \frac{s+1}{Q}) \text{ και } \{q_2 a\} \in [\frac{s}{Q}, \frac{s+1}{Q}).$$

Θεωρώντας  $q = |q_1 - q_2|$  παίρνουμε για κάποιον ακέραιο  $p$ :

$$|qa - p| < \frac{1}{Q}.$$

Διαιρώντας με  $q$  έχουμε

$$|a - \frac{p}{q}| < \frac{1}{Qq} \leq q^{-2}$$

αφού, εξ' ορισμού,  $0 < qQ$ . Τώρα πρέπει να δείξουμε ότι ο αριθμός τέτοιων ζευγών  $(p, q)$  είναι άπειρος. Κάνουμε την αντίθετη υπόθεση, ότι δηλαδή υπάρχει πεπερασμένος αριθμός από  $r_i = \frac{p_i}{q_i}$ ,  $i = 1, \dots, N$ ,  $|a - r_i| < q_i^{-2}$ . Αφού καμία διαφορά δεν είναι ακριβώς 0, υπάρχει ακέραιος  $Q$  τέτοιος ώστε  $|a - r_i| > \frac{1}{Q}$  για κάθε  $i = 1, \dots, N$ . Εφαρμόζοντας τις αρχικές υποθέσεις για τον  $Q$  προκύπτει ότι  $r = \frac{p}{q}$  τέτοιο ώστε

$$|a - r| < Qq \leq \frac{1}{Q}.$$

Κατά συνέπεια ο  $r$  δεν μπορεί να είναι κάποιος από τους  $r_i$ ,  $i = 1, \dots, N$ . Από την άλλη πλευρά, όπως και παραπάνω, ισχύει  $|a - r| < q^{-2}$  που είναι άτοπο με βάση την υπόθεση ότι τα κλάσματα  $r_i$ ,  $i = 1, \dots, N$  αποτελούν τα μόνα κλάσματα που φέρουν αυτή την ιδιότητα.  $\square$

### 1.8.3 Απόδειξη με Εις Άτοπον Απαγωγή (Proof by contradiction - reductio ad absurdum)

Υποθέτουμε ότι κάποιο γεγονός  $P$  είναι ψευδές. Δείχνουμε ότι η παραπάνω υπόθεση οδηγεί σε άτοπο. Συμπεραίνουμε, ως εκ τούτου, ότι το  $P$  είναι αληθές.

**Παράδειγμα 1.8.4.** Ο αριθμός  $\sqrt{2}$  είναι άρρητος.

Απόδειξη. Ξεκινάμε με την υπόθεση ότι ο αριθμός  $\sqrt{2}$  είναι ρητός. Επομένως μπορούμε να γράψουμε

$$\sqrt{2} = \frac{m}{n},$$

όπου οι  $m, n$  είναι ακέραιοι. Αν τα  $m, n$  είναι διαιρετά από τον ίδιο ακέραιο μεγαλύτερο του 1, διαιρούμε και τους δύο με αυτόν τον αριθμό. Η διαίρεση δεν αλλάζει την τιμή του κλάσματος, αλλά

πλέον οι  $m, n$  δεν μπορεί να είναι άρτιοι αριθμοί. Πολλαπλασιάζουμε και τα δύο μέλη της παραπάνω σχέσης επί  $n$  κι έχουμε:

$$n\sqrt{2} = m.$$

Υψώνουμε και τα δύο μέλη στο τετράγωνο:

$$2n^2 = m^2.$$

Είναι προφανές, ότι ο αριθμός  $m^2$  είναι άρτιος, και κατά συνέπεια και ο  $m$  είναι άρτιος (αφού το τετράγωνο ενός περιττού αριθμού είναι πάντα περιττός). Μπορούμε επομένως να γράψουμε  $m = 2k$  για κάποιον ακέραιο  $k$ . Αντικαθιστώντας το  $m$  με  $2k$  στην παραπάνω σχέση προκύπτει:

$$\begin{aligned} 2n^2 &= (2k)^2 \\ &= 4k^2. \end{aligned}$$

Διαιρώντας και τα δύο μέλη με 2, έχουμε:

$$n^2 = 2k^2.$$

Με βάση αυτό το αποτέλεσμα προκύπτει ότι ο  $n^2$  είναι άρτιος, και κατά συνέπεια και ο  $n$  είναι άρτιος. Καταλήξαμε δηλαδή στο συμπέρασμα ότι και ο  $m$  και ο  $n$  είναι άρτιοι. Αυτό όμως είναι άτοπο μιας και η αρχική υπόθεση ήταν ότι οι  $m, n$  δεν είναι άρτιοι. Κατά συνέπεια, η αρχική υπόθεση, ότι δηλαδή ο αριθμός  $\sqrt{2}$  είναι ρητός, είναι λάθος. Άρα ο  $\sqrt{2}$  είναι άρρητος.  $\square$

#### 1.8.4 Αρχή της Διαγωνοποίησης

**Διατύπωση:** Έστω  $R$  μια σχέση σε ένα σύνολο  $A$  και έστω  $D$  το διαγώνιο σύνολο για τη  $R$ ,  $\{\alpha : \alpha \in A \text{ και } (\alpha, \alpha) \in R\}$ . Για κάθε  $\alpha \in A$ , έστω  $R_\alpha = \{b : b \in A \text{ και } (\alpha, b) \in R\}$ . Τότε το  $D$  είναι διαφορετικό από όλα τα  $R_\alpha$ .

Αν  $A$  είναι ένα περασμένο σύνολο τότε το  $R$  μπορεί να απεικονιστεί ως ένας τετραγωνικός πίνακας. Οι γραμμές και οι στήλες επιγράφονται με τα ονόματα των στοιχείων του  $A$  και υπάρχει ένας σταυρός στο τετραγωνίδιο της γραμμής που επιγράφεται με  $\alpha$  και της στήλης που επιγράφεται με  $b$  στην περίπτωση που  $(\alpha, b) \in R$ . Το διαγώνιο σύνολο  $D$  αντιστοιχεί στο συμπλήρωμα της ακολουθίας των τετραγωνιδίων κατά μήκος της κυρίας διαγωνίου που προκύπτει αν εναλλάξουμε τα τετραγωνίδια με σταυρούς σε τετραγωνίδια χωρίς σταυρούς και αντίστροφα. Τα σύνολα  $R_\alpha$  αντιστοιχούν στις σειρές του πίνακα. Η αρχή της διαγωνοποίησης μπορεί τότε να διατυπωθεί διαφορετικά ως εξής: το συμπλήρωμα της διαγωνίου διαφέρει από κάθε σειρά.







**Παράδειγμα 1.8.5.** Έστω το σύνολο  $A = \{\alpha, b, c, d, e, f\}$  και

$$R = \{(\alpha, b), (\alpha, d), (b, b), (b, c), (c, c), (d, b), (d, c), (d, e), (d, f), (e, e), (e, f), (f, a), (f, c), (f, d), (f, e)\}.$$

Η  $R$  μπορεί να απεικονιστεί ως εξής:

Η ακολουθία των τετραγωνιδίων κατά μήκος της διαγωνίου είναι:



	$\alpha$	$b$	$c$	$d$	$e$	$f$	
$\alpha$		<b>x</b>		<b>x</b>			$R_\alpha = \{b, d\}$
$b$			<b>x</b>				$R_b = \{b, c\}$
$c$							$R_c = \{c\}$
$d$		<b>x</b>	<b>x</b>		<b>x</b>	<b>x</b>	$R_d = \{b, c, e, f\}$
$e$						<b>x</b>	$R_e = \{e, f\}$
$f$	<b>x</b>		<b>x</b>	<b>x</b>	<b>x</b>		$R_f = \{\alpha, c, d, e\}$

	<b>x</b>	<b>x</b>		<b>x</b>	
--	----------	----------	--	----------	--

και το συμπλήρωμά της είναι:

<b>x</b>			<b>x</b>		<b>x</b>
----------	--	--	----------	--	----------

που αντιστοιχεί στο διαγώνιο σύνολο  $D = \{\alpha, d, f\}$  και διαφέρει από κάθε γραμμή του πίνακα. Εξαιτίας του τρόπου κατασκευής της, αυτή η γραμμή διαφέρει από την πρώτη γραμμή στην πρώτη θέση, από τη δεύτερη γραμμή στη δεύτερη θέση κ.ο.κ.

Η αρχή της διαγωνοποίησης ισχύει και για τα άπειρα σύνολα για παρόμοιους λόγους: το διαγώνιο σύνολο  $D$  πάντα διαφέρει από το σύνολο  $R_\alpha$  στην ερώτηση αν το  $\alpha$  είναι στοιχείο του κι έτσι δεν μπορεί να ταυτίζεται με το  $R_\alpha$  για κανένα  $\alpha$ .

Στη συνέχεια παρουσιάζουμε ένα παράδειγμα χρήσης της διαγωνοποίησης με εφαρμογή στην απόδειξη του θεωρήματος του Georg Cantor.

**Παράδειγμα 1.8.6.** Το σύνολο  $2^N$  είναι μη μετρήσιμο.

Απόδειξη. Υποθέτουμε ότι το σύνολο  $2^N$  είναι μετρήσιμα άπειρο, δηλαδή υπάρχει μια αμφιμονοσή-

μαντη αντιστοιχία  $f : N \rightarrow 2^N$ . Τότε το  $2^N$  μπορεί να αριθμηθεί ως

$$2^N = \{S_0, S_1, S_2, \dots\},$$

όπου  $S_i = f(i)$  για κάθε  $i \in N$ . Θεωρούμε το σύνολο

$$D = \{n \in N\}.$$

Το  $D$  είναι το σύνολο των φυσικών αριθμών και γι' αυτό θα έπρεπε να είναι  $S_k$  για κάποιο φυσικό αριθμό  $k$ . Εξετάζουμε αν ισχύει  $k \in S_k$ :

1. Έστω ότι  $k \in S_k$ . Αφού το  $D = \{n : n \notin S_n\}$ , έχουμε ότι  $k \in D$ , όμως  $D = S_k$ , τότε  $k \notin S_k$ , πράγμα άτοπο.
2. Έστω ότι  $k \notin S_k$ . Τότε  $k \in D$ . Όμως  $D = S_k$ , οπότε  $k \in S_k$ , που είναι επίσης αντίφαση.

Αφού ούτε το (1) ούτε το (2) είναι δυνατόν, η υπόθεση ότι  $d = S_k$  για κάποιο  $k$  είναι λάθος. Άρα, το  $2^N$  είναι μη μετρήσιμο.  $\square$

Το θεώρημα του Cantor δηλώνει ουσιαστικά ότι ανεξάρτητα από το πόσο μεγάλο είναι ένα δοσμένο σύνολο, πάντα υπάρχει ένα μεγαλύτερο από αυτό σύνολο.

Αυτό είναι προφανές αν το αρχικό σύνολο είναι πεπερασμένο, αλλά αυτό δεν ισχύει στην περίπτωση που το αρχικό σύνολο είναι άπειρο.

Θεωρούμε ότι δύο άπειρα σύνολα έχουν το ίδιο μέγεθος, δηλ. τον ίδιο πληθικό αριθμό, αν μπορούμε να σχηματίσουμε μία ένα-προς-ένα αντιστοιχία ανάμεσα στα στοιχεία των δύο συνόλων (χωρίς να μείνει αταίριαστο κανένα στοιχείο). Αν δείξουμε ότι μεταξύ δύο άπειρων συνόλων μια τέτοια αντιστοιχία δεν μπορεί ποτέ να κατασκευαστεί, τότε συμπεραίνουμε ότι το ένα σύνολο είναι μεγαλύτερο από το άλλο. Αποδεικνύουμε χρησιμοποιώντας την αρχή της Διαγωνοποίησης.

Υποθέτουμε ότι υπάρχει το μέγιστο άπειρο σύνολο και δείχνουμε ότι πάντα υπάρχει ένα άπειρο σύνολο ακόμα μεγαλύτερο. Έστω, λοιπόν,  $X$  ένα άπειρο σύνολο με  $X = \{a, b, c, d, \dots\}$ . Υποθέτουμε ότι το  $X$  είναι το μεγαλύτερο σε μέγεθος σύνολο που υπάρχει και ότι κανένα άλλο άπειρο σύνολο δεν μπορεί να είναι μεγαλύτερο.

Το δυναμοσύνολο ενός συνόλου  $X$  είναι το σύνολο των υποσυνόλων του  $X$  και το συμβολίζουμε  $P(X)$ , με  $P(X) = \{\{a\}, \{a, b\}, \{b, c, e\}, \{a, c\}, \{c\}, \dots\}$ .

- Το  $P(X)$  είναι ένα σύνολο.
- Σύμφωνα με την αρχική υπόθεση, το σύνολο  $P(X)$  δεν μπορεί να είναι μεγαλύτερο από το σύνολο  $X$ , αφού το  $X$  υποθέσαμε ότι είναι το μέγιστο άπειρο σύνολο που υπάρχει.
- Επίσης, το  $P(X)$ , δεν μπορεί να είναι μικρότερο από το  $X$ , αφού περιέχει όλα τα απλά στοιχεία του  $X$  με τη μορφή μονοσυνόλων  $\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \dots\}$ , για κάθε στοιχείο  $a, b, c, \dots$  του  $X$ .

Άρα, θα πρέπει τα σύνολα  $X$  και  $P(X)$  να έχουν το ίδιο μέγεθος, που σημαίνει ότι θα μπορεί να σχηματιστεί αμφιμονοσήμαντη αντιστοιχία ανάμεσα στα στοιχεία των συνόλων  $X$  και  $P(X)$ .

$X$	$\leftrightarrow$	$P(x)$
$a$	$\leftrightarrow$	$\{c, d\}$
$b$	$\leftrightarrow$	$\{a\}$
$c$	$\leftrightarrow$	$\{a, b, c, d\}$
$d$	$\leftrightarrow$	$\{b, e\}$
$e$	$\leftrightarrow$	$\{a, c, e\}$
$\dots$		

Παρατηρούμε, ότι κάποια από τα στοιχεία του  $Q$  αντιστοιχίζονται με υποσύνολα που τα περιέχουν, π.χ. το  $e$  αντιστοιχίζεται με το  $\{a, c, e\}$ , ενώ άλλα στοιχεία του  $Q$  αντιστοιχίζονται με υποσύνολα που δεν τα περιέχουν, π.χ. το  $a$  αντιστοιχίζεται με το  $\{c, d\}$ . Θεωρούμε το σύνολο των στοιχείων του  $Q$  που ΔΕΝ αντιστοιχίζονται με υποσύνολα που τα περιέχουν, το οποίο καλούμε  $F$ . Άρα, το  $F$  είναι ένα υποσύνολο του  $Q$ , οπότε πρέπει να εμφανίζεται στην παραπάνω λίστα των υποσυνόλων του  $Q$ .

Το ερώτημα τώρα είναι το εξής: ποιο θα μπορούσε να είναι το στοιχείο του  $Q$  με το οποίο αντιστοιχίζεται το  $F$ .

1. Δεν μπορεί να είναι κάποιο στοιχείο του  $F$ , αφού το  $F$  κατασκευάστηκε ώστε να περιέχει στοιχεία του  $Q$  που ΔΕΝ αντιστοιχίζονται με υποσύνολα του  $Q$  που τα περιέχουν.
2. Αν το στοιχείο του  $Q$  που αντιστοιχίζεται με το  $F$  δεν ανήκει στο  $F$ , τότε πρέπει να ανήκει στο  $F$  με βάση τον ορισμό του  $F$ , που είναι αντίφαση κι επομένως καταλήγουμε σε άτοπο.

Συμπεραίνουμε, επομένως, ότι δεν υπάρχει στοιχείο του  $Q$  που να αντιστοιχίζεται με το  $F$ . Άρα δεν μπορούμε να έχουμε ένα-προσ-ένα αντιστοιχία ανάμεσα στα στοιχεία των συνόλων  $Q$  και  $F$ , που σημαίνει ότι τα δύο σύνολα δεν έχουν το ίδιο μέγεθος. Επιπλέον, λόγω της παρατήρησης, που κάναμε προηγουμένως, ότι το  $P(X)$  δεν μπορεί να είναι μικρότερο σε μέγεθος από το  $Q$ , συμπεραίνουμε ότι το σύνολο  $P(X)$  είναι μεγαλύτερο σε μέγεθος από το  $X$ .



## Κεφάλαιο 2

# Κανονικές γλώσσες

Στη συνέχεια ασχολούμαστε με θεμελιώδεις δυνατότητες και σημαντικούς περιορισμούς του υπολογισμού: τι μπορεί να γίνει και με ποιες προϋποθέσεις. Θα μελετήσουμε τρία μοντέλα υπολογισμού: τα Πεπερασμένα αυτόματα, τα αυτόματα Στοίβας και τις μηχανές Turing. Παράλληλα, θα δούμε κι άλλους τρόπους περιγραφής υπολογισμών και αλγορίθμων, μέσω της γλώσσας των μαθηματικών, συμπεριλαμβανομένων των Κανονικών Εκφράσεων και των Γραμματικών. Έννοιες όπως τα Πεπερασμένα αυτόματα είναι εξαιρετικά σημαντικές στην Επιστήμη των Υπολογιστών, αλλά ακόμα και το να αποδείξουμε ότι κάτι είναι αδύνατον είναι χρήσιμο, αφού έτσι γνωρίζουμε που να μην αναζητήσουμε λύσεις, οι οποίες θα απαιτούν κάποιου είδους συμβιβασμό. Οι είσοδοι στους υπολογιστές μας θα είναι πάντοτε συμβολοσειρές μιας και τα πάντα μπορούν να μετατραπούν σε ερωτήσεις σχετικά με συμβολοσειρές. Αρχικά, παρουσιάζουμε την απλούστερη μορφή ενός υπολογιστή ή ίσως μιας μηχανής. Για παράδειγμα, μια αυτόματη πόρτα. Είναι συνέχεια είτε ανοιχτή είτε κλειστή. Ο σχεδιασμός είναι απλός. Ανοιχτή, Κλειστή. Ή ίσως Να ανοίγει, Ανοιχτή, Να κλείνει, Κλειστή. Ίσως να υπάρχει και επικάλυψη. Αυτή είναι η απλούστερη μορφή μηχανής: μόνο εσωτερική μνήμη, τίποτα εξωτερικό, μόνο αντίδραση σε γεγονότα. Αυτό είναι το πεπερασμένο αυτόματο.

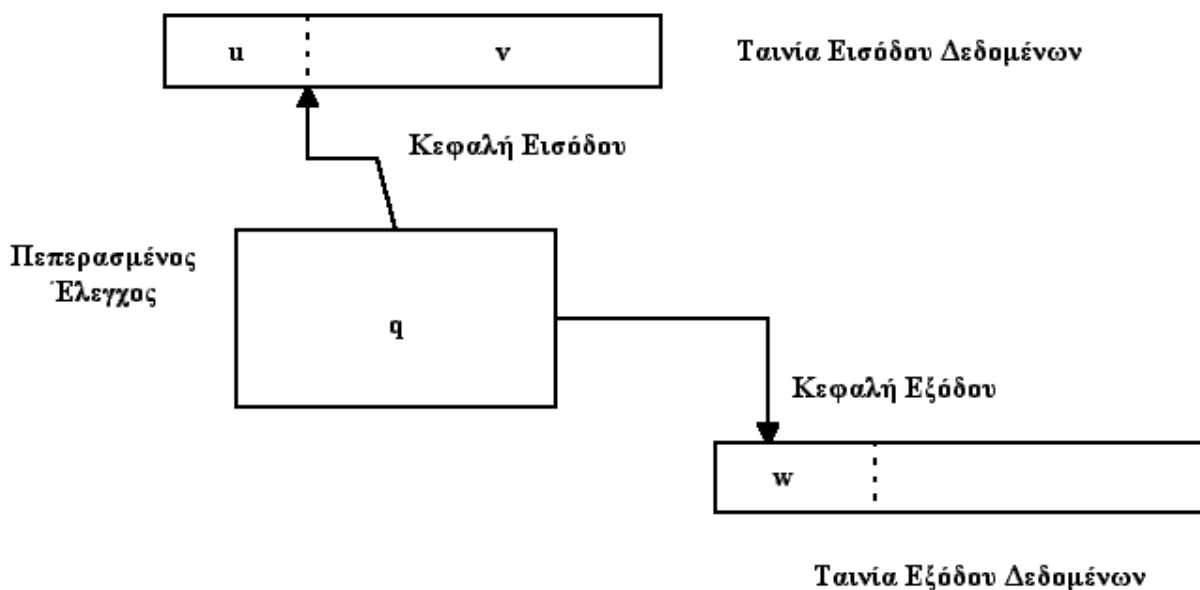
Το σύνολο των συμβολοσειρών είναι ένα πολύ χρήσιμο μέσο για την αναπαράσταση πληροφορίας αρκεί να υπάρχει συνάρτηση που να μεταφράζει τη μεταφερόμενη πληροφορία. Μετάφραση είναι η αντίστροφη διαδικασία της απεικόνισης (mapping), δηλαδή, μετάφραση είναι μια συνάρτηση  $g$  από το  $\Sigma^*$  στο  $D$ , όπου  $\Sigma^*$  είναι κάποιο αλφάβητο και  $D$  είναι κάποιο σύνολο. Για παράδειγμα, η συμβολοσειρά 111 μπορεί να μεταφραστεί σαν τον αριθμό εκατόν έντεκα αν αναπαριστάνει μια συμβολοσειρά στο δεκαδικό σύστημα αρίθμησης, ή σαν τον αριθμό επτά αν το σύστημα αρίθμησης είναι το δυαδικό. Ένας αποστολέας κι ένας παραλήπτης που επικοινωνούν κάνουν και αναπαράσταση και μετάφραση. Η αναπαράσταση γίνεται από τον αποστολέα και η μετάφραση από τον παραλήπτη και ακολουθείται η ίδια διαδικασία ανεξάρτητα από το αν ο αποστολέας και ο παραλήπτης είναι άνθρωποι ή προγράμματα. Επομένως, από την πλευρά των μερών που επικοινωνούν, μια γλώσσα πρόκειται για μια συλλογή συμβολοσειρών και τα εκάστοτε δύο μέρη διαθέτουν τις συναρτήσεις αναπαράστασης και μετάφρασης τις οποίες και γνωστοποιούν. Σε ό,τι ακολουθεί θα μελετήσουμε τις σχέσεις μεταξύ

γλωσσών, μηχανών και γραμματικών.

Ως γλώσσα θεωρούμε ένα σύνολο συμβολοσειρών που προκύπτουν από συνδυασμούς μηδέν ή περισσότερων συμβόλων ενός πεπερασμένου αλφαβήτου. Οι μηχανές περιλαμβάνουν κάποιον τρόπο ανάγνωσης συμβόλων εισόδου, ένα κάθε φορά, μέσω κάποιου μηχανισμού ανάγνωσης της εισόδου, π.χ. ταινία εισόδου, καθώς επίσης και κάποιον πεπερασμένο έλεγχο, λειτουργίες δηλαδή που μπορούν να επιτελεστούν στα σύμβολα εισόδου. Οι αποθηκευτικοί χώροι, μία ή περισσότερες κεφαλές ανάγνωσης και τρόποι μετακίνησης των κεφαλών αυτών είναι στοιχεία που διαφοροποιούν τα διάφορα μοντέλα μηχανών και παρέχουν τη δυνατότητα εκτέλεσης όχι μόνο λειτουργιών διαφορετικής πολυπλοκότητας από κάθε μοντέλο αλλά και της ίδιας λειτουργίας σε διαφορετικό χρόνο. Μηχανές που έχουν επιπλέον δυνατότητα εξαγωγής αποτελεσμάτων θεωρούνται γεννήτριες γλωσσών, αφού μπορούν να παράγουν με κατάλληλο πεπερασμένο έλεγχο συμβολοσειρές μιας συγκεκριμένης γλώσσας. Μπορούν, επίσης, να θεωρηθούν σαν υπολογιστές αφού με δεδομένη κάποια συμβολοσειρά εισόδου, η μηχανή μπορεί να παράγει μια συμβολοσειρά εξόδου και να τερματίσει τη λειτουργία της ή, στην περίπτωση που δεν έχει οριστεί κάποια λειτουργία, η μηχανή δεν τερματίζει. Σε ό,τι ακολουθεί, οι μηχανές θεωρούνται αναγνωριστές γλωσσών με την έννοια ότι με δεδομένη κάποια συμβολοσειρά εισόδου, η μηχανή θα εκτελεί έναν αριθμό βημάτων καταναλώνοντας την είσοδο και θα τερματίζει τη λειτουργία της μόνο στην περίπτωση που η είσοδος γίνεται αποδεκτή. Άλλα σημαντικά στοιχεία είναι:

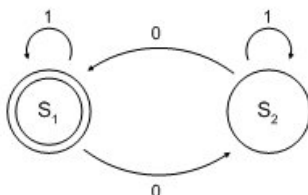
- αν μια γλώσσα είναι υπολογίσιμη (computable), δηλαδή αν υπάρχει μηχανή (υπολογιστικό μοντέλο) που να μπορεί να αναγνωρίσει τη γλώσσα αυτή,
- αν μια γλώσσα είναι υπολογίσιμη από μηχανή που πάντα τερματίζει τη λειτουργία μετά από πεπερασμένο αριθμό βημάτων (decidability),
- αν μια γλώσσα είναι υπολογίσιμη από μια “αποδοτική” μηχανή, μια μηχανή, δηλαδή, που λειτουργεί σε πολυωνυμικό χρόνο (tractability).

Διαφορετικές τάξεις γλωσσών διακρίνονται με βάση την υπολογιστική πολυπλοκότητα που απαιτείται για να αποφασιστεί αν μια δεδομένη ακολουθία χαρακτήρων (string=αλφαριθμητικό) ανήκει ή όχι στη γλώσσα. Κανονικές είναι οι γλώσσες αυτές που μπορούν να περιγραφούν από κανονικές εκφράσεις (regular expressions). Πρόκειται για την απλούστερη τάξη γλωσσών μιας και απαιτείται η ελάχιστη υπολογιστική πολυπλοκότητα για την αναγνώρισή τους. Οι μηχανές που τις αναγνωρίζουν καλούνται πεπερασμένα αυτόματα. Τα πεπερασμένα αυτόματα, που ανήκουν στη γενικότερη τάξη των μηχανών πεπερασμένου αριθμού καταστάσεων, μπορούν σε κάθε χρονική στιγμή να βρίσκονται σε μια κατάσταση από ένα πεπερασμένο σύνολο καταστάσεων. Όταν ένα αυτόματο βρίσκεται σε κάποια κατάσταση, διαβάζοντας το επόμενο σύμβολο εισόδου μεταβαίνει σε μια νέα κατάσταση, δηλαδή κάθε φορά η επόμενη κατάσταση είναι συνάρτηση της παρούσας κατάστασης και του συμβόλου εισόδου. Η συνάρτηση αυτή καλείται συνάρτηση μετάβασης του αυτομάτου (transition function).



Όταν αρχίζει ο υπολογισμός, το αυτόματο βρίσκεται σε μια μονοσήμαντα ορισμένη αρχική κατάσταση. Στη συνέχεια, πραγματοποιεί μια σειρά από μεταβάσεις που καθορίζονται από τη συμβολοσειρά εισόδου και τη συνάρτηση μετάβασης που του αντιστοιχεί. Ολοκλήρωση της ανάγνωσης της συμβολοσειράς εισόδου σημαίνει ότι το αυτόματο βρίσκεται σε κάποια τελική κατάσταση από ένα προκαθορισμένο σύνολο τελικών καταστάσεων, δηλαδή η υπό εξέταση συμβολοσειρά ανήκει στη γλώσσα. Αν δεν φτάσει το αυτόματο σε τελική κατάσταση, δηλαδή αν για συγκεκριμένο συνδυασμό παρούσας κατάστασης και συμβόλου εισόδου δεν ορίζεται κάποια μετάβαση, η συμβολοσειρά δεν γίνεται δεκτή από το αυτόματο και κατά συνέπεια δεν ανήκει στη γλώσσα.

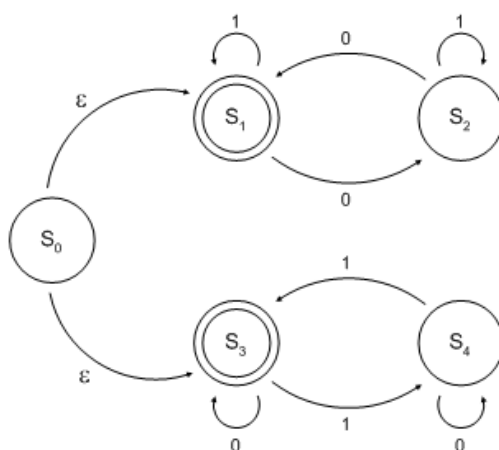
Οι μη τελικές καταστάσεις του αυτομάτου αναπαρίστανται με κύκλους ενώ οι τελικές με διπλούς κύκλους. Οι μεταβάσεις από μια κατάσταση σε μια άλλη αναπαρίστανται με βέλη πάνω στα οποία αναγράφονται τα σύμβολα εισόδου.



Σχήμα 2.1: Παράδειγμα διαγράμματος καταστάσεων ενός Ντετερμινιστικού Πεπερασμένου Αυτομάτου - DFA

Υπάρχουν δύο τύποι πεπερασμένων αυτομάτων, τα Ντετερμινιστικά Πεπερασμένα Αυτόματα (Deterministic Finite Automata – DFAs) και τα Μη Ντετερμινιστικά Πεπερασμένα Αυτόματα (Non

– deterministic Finite Automata – NFAs). Η συμπεριφορά των πρώτων καθορίζεται πλήρως κατά το σχεδιασμό τους. Για κάθε κατάσταση υπάρχει μόνο μια δυνατή μετάβαση για δεδομένο σύμβολο εισόδου, ενώ δεν επιτρέπονται μεταβάσεις χωρίς κατανάλωση εισόδου. Για ένα NFA που βρίσκεται σε κάποια κατάσταση, είναι δυνατόν να υπάρχουν εναλλακτικές μεταβάσεις για δεδομένο σύμβολο εισόδου ενώ δίνεται επιπλέον η δυνατότητα μετάβασης μεταξύ καταστάσεων χωρίς την κατανάλωση εισόδου. Οι τελευταίες μεταβάσεις συμβολίζονται με βέλη που επιγράφονται με  $\epsilon$ , όπου  $\epsilon$  είναι η κενή συμβολοσειρά. Αφού επομένως μια συμβολοσειρά εισόδου μπορεί να οδηγήσει ένα NFA σε διαφορετικές ακολουθίες υπολογισμών, μια είσοδος γίνεται αποδεκτή αν υπάρχει αποδεκτή ακολουθία μεταβάσεων του αυτομάτου που το οδηγεί σε τελική κατάσταση. Τα NFAs σχεδιάζονται ευκολότερα και είναι μικρότερα και απλούστερα από τα DFAs.



Σχήμα 2.2: Παράδειγμα διαγράμματος καταστάσεων ενός Μη-Ντετερμινιστικού Πεπερασμένου Αυτομάτου - NFA

Τα πεπερασμένα αυτόματα αποτελούνται από ένα σύνολο καταστάσεων, και από κανόνες για το πώς εκτελούνται οι μεταβάσεις μεταξύ των καταστάσεων αυτών με βάση την είσοδο. Επίσης, υπάρχει ένα σύνολο από αναγνωρίσιμους χαρακτήρες εισόδου, που καλείται αλφάβητο, και προκαθορισμένη αρχική κατάσταση καθώς και ένα προκαθορισμένο σύνολο τελικών καταστάσεων.

Μια γλώσσα είναι κανονική αν αναγνωρίζεται από κάποιο ντετερμινιστικό πεπερασμένο αυτόματο (DFA).

Ένα μη ντετερμινιστικό πεπερασμένο αυτόματο (Nondeterministic Finite Automaton (NFA)) είναι ίδιο με ένα DFA, με τη μόνη διαφορά ότι η  $\delta$  δεν είναι πλέον συνάρτηση, αλλά σχέση μετάβασης. Σε ένα NFA, επιτρέπεται να υπάρχουν μεταβάσεις από μία κατάσταση σε πολλές άλλες επόμενες καταστάσεις ή και σε καμία κατάσταση για κάποιο/α σύμβολα εισόδου, επιτρέπεται να υπάρχουν μεταβάσεις σε διαφορετικές επόμενες καταστάσεις από μία κατάσταση με συγκεκριμένο σύμβολο εισόδου, και επίσης επιτρέπεται να υπάρχουν μεταβάσεις χωρίς την κατανάλωση εισόδου (δηλ. με



την κενή συμβολοσειρά που συμβολίζεται  $\epsilon$ .)

Για να μετατρέψουμε αυτό το διάγραμμα καταστάσεων σε ένα που να μην έχει ελεύθερες μεταβάσεις, θα πρέπει να θεωρήσουμε ένα NFA που να μπορεί να βρίσκεται σε περισσότερες από μία καταστάσεις κάθε φορά. Αυτό συμβαίνει γιατί μια ελεύθερη μετάβαση ισοδυναμεί με την ιδέα πολλών διαφορετικών καταστάσεων συνδεδεμένων μεταξύ τους, που αποτελούν μία νέα σύνθετη κατάσταση. Αφού καταναλωθεί όλη η συμβολοσειρά εισόδου, αν το αυτόματο βρίσκεται σε σύνθετη κατάσταση που περιέχει κάποια τελική κατάσταση, η συμβολοσειρά γίνεται αποδεκτή. Προκειμένου να κατανοήσουμε το διάγραμμα καταστάσεων, υποθέτουμε ότι οι τρεις χαμηλότερες καταστάσεις είναι αρχικές αφού μπορεί το αυτόματο να βρεθεί σε αυτές χωρίς κατανάλωση εισόδου. Στη συνέχεια, ξαναφτιάχνουμε το διάγραμμα ώστε να περιλαμβάνονται όλες οι πιθανές μεταβάσεις από μια κατάσταση σε άλλη και υποθέτουμε ότι ελεύθερες μεταβάσεις είναι πάντα επιτρεπτές.

Χρησιμοποιώντας τέτοια NFA, γίνονται ευκολότερα συγκεκριμένες αποδείξεις. Μια τέτοια απόδειξη αφορά στην ισοδυναμία μεταξύ NFA και DFA. Αυτό σημαίνει ότι τα NFA και τα DFA αναγνωρίζουν το ίδιο σύνολο γλωσσών.

## 2.1 Πεπερασμένα αυτόματα

Το βασικότερο μοντέλο ενός υπολογιστή είναι το πεπερασμένο αυτόματο. Πρόκειται για έναν υπολογιστή χωρίς μνήμη, ή καλύτερα με προκαθορισμένη μνήμη, ανεξάρτητα από το μέγεθος της εισόδου.

Μια συμβολοσειρά είναι μια ακολουθία χαρακτήρων ή συμβόλων. Μια μηχανή με πεπερασμένο πλήθος καταστάσεων ή αλλιώς ένα πεπερασμένο αυτόματο (FA) είναι ένα κατασκευάσμα που αναγνωρίζει κάποιο σύνολο συμβολοσειρών. Ένα FA αποτελείται από τα εξής τρία στοιχεία:

1. Μια ταινία εισόδου, που περιλαμβάνει μία μόνο συμβολοσειρά,
2. Ένας ασινητήρας ή κεφαλή που διαβάζει τη συμβολοσειρά εισόδου σύμβολο - σύμβολο, και
3. Μνήμη, που μπορεί να είναι σε μία από πεπερασμένο αριθμό καταστάσεων • άρα ουσιαστικά, η τρέχουσα κατάσταση του αυτομάτου.

Το 'πρόγραμμα' του FA περιγράφει πώς τα σύμβολα που διαβάζονται επηρεάζουν την παρούσα κατάσταση. Η τελική κατάσταση για μια συμβολοσειρά είναι η κατάσταση στην οποία βρίσκεται το αυτόματο όταν ολοκληρώνεται η ανάγνωση της εισόδου.

### Λειτουργία ενός FA

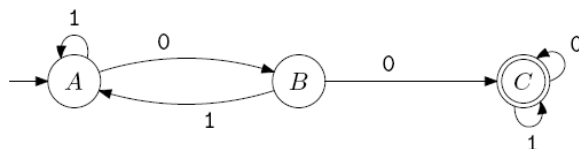
1. Θέτουμε τη μηχανή στην αρχική κατάσταση.
2. Αν έχουμε φτάσει στο τέλος της συμβολοσειράς εισόδου τότε η μηχανή σταματά.

3. Διαβάζουμε ένα σύμβολο.
4. Ανανεώνουμε την κατάσταση της μηχανής με βάση την τρέχουσα κατάσταση και το σύμβολο που διαβάσαμε.
5. Πηγαίνουμε στο βήμα 2.

Ένα FA μπορεί να περιγραφεί από ένα διάγραμμα στο οποίο κάθε κατάσταση αναπαριστάνεται από έναν κύκλο και το όνομά της γράφεται μέσα στον κύκλο. Κάθε κατάσταση έχει, για κάθε σύμβολο, ένα βέλος που δείχνει στην επόμενη κατάσταση. Η αρχική κατάσταση υποδεικνύεται με ένα απλά εισερχόμενο σε αυτή βέλος. Ο σκοπός ενός FA είναι να λειτουργεί σαν ένας αναγνωριστής - στην ουσία λειτουργεί σα μία λογική συνάρτηση. Για κάθε FA, συγκεκριμένες καταστάσεις έχουν σχεδιασθεί σαν καταστάσεις αποδοχής (accept states) ενώ οι υπόλοιπες είναι καταστάσεις μη αποδοχής. Μια κατάσταση αποδοχής δηλώνεται με έναν διπλό κύκλο στο διάγραμμα.

**Ορισμός 2.1.1.** Ένα FA αποδέχεται τη συμβολοσειρά εισόδου αν η κατάσταση στην οποία καταλήγει είναι κατάσταση αποδοχής, διαφορετικά απορρίπτει τη συμβολοσειρά εισόδου.

**Παράδειγμα 2.1.1.** Το παρακάτω είναι ένα FA με 3 καταστάσεις, τις A, B, και C. Αρχική κατάσταση είναι η A, ενώ η C είναι η μόνη τελική κατάσταση (κατάσταση αποδοχής).



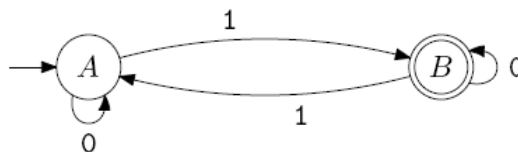
Ας εξετάσουμε τι κάνει το αυτόματο όταν σαν είσοδος δίνεται η συμβολοσειρά 101001:

Παρούσα κατάσταση	Σύμβολο που διαβάζουμε	Επόμενη κατάσταση
A	1	A
A	0	B
B	1	A
A	0	B
B	0	C
C	1	C

Η τελική κατάσταση είναι η C. Όμοια, η τελική κατάσταση για τη συμβολοσειρά 11101 είναι η A, ενώ για τη συμβολοσειρά 0001 τελική κατάσταση είναι η C. Τι πρέπει να γίνει για να φτάσουμε σε τελική κατάσταση; Η συγκεκριμένη μηχανή αποδέχεται όλες τις συμβολοσειρές που περιέχουν 0 και 1 και περιέχουν δύο συνεχόμενα μηδέν.

**Παράδειγμα 2.1.2.** Θεωρήστε το παρακάτω FA.

Αυτό το FA αποδέχεται συμβολοσειρές όπως οι 100, 0101001, 11111, και απορρίπτει συμβολοσειρές όπως οι 000 και 0110. Μπορούμε να περιγράψουμε ακριβώς ποιες συμβολοσειρές αποδέχεται



το FA; Το FA αγνοεί το σύμβολο 0 (δεν αλλάζει κατάσταση). Ασχολείται μόνο με το σύμβολο 1; και εδώ αλλάζει κατάσταση. Το πρώτο 1 οδηγεί το αυτόματο στην κατάσταση B, το δεύτερο 1 το οδηγεί στην κατάσταση A, το τρίτο στην κατάσταση B, και ούτω καθεξής. Επομένως, το αυτόματο είναι στην κατάσταση B όταν έχει διαβαστεί περιττός αριθμός από 1. Δηλαδή, η μηχανή αποδέχεται όλες τις συμβολοσειρές από 0 και 1 που περιέχουν περιττό πλήθος 1.

Σε αυτά τα παραδείγματα, χρησιμοποιούμε απλά γράμματα σαν ονόματα για τις καταστάσεις. Εννοείται ότι μπορούν να χρησιμοποιηθούν και περισσότερο περιγραφικά ονόματα.

### 2.1.1 Κατασκευή FAs

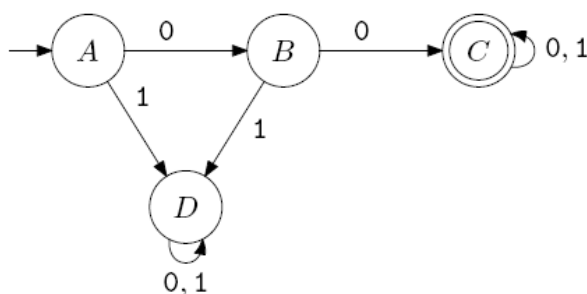
Δεν υπάρχει κάποιος συγκεκριμένος τρόπος για να σχεδιάζουμε πεπερασμένα αυτόματα. Χρειάζεται απλά εξάσκηση και σκέψη. Στη συνέχεια, θα δούμε πώς να κατασκευάζουμε ένα FA για συγκεκριμένο σκοπό. Παραθέτουμε μερικούς αναγκαίους ορισμούς: Ένα αλφάβητο είναι ένα σύνολο συμβόλων. Μια γλώσσα είναι ένα σύνολο συμβολοσειρών, όπου οι συμβολοσειρές έχουν σύμβολα από ένα συγκεκριμένο αλφάβητο. Γλώσσα ενός FA είναι το σύνολο των συμβολοσειρών που αυτό αποδέχεται. Για παράδειγμα, η γλώσσα του πρώτου FA από το παράδειγμα 2.1.1 είναι το σύνολο των συμβολοσειρών του αλφαβήτου  $\{0, 1\}$  που περιέχουν την υποσυμβολοσειρά 00. Συνήθως, χρησιμοποιούμε το γράμμα  $\Sigma$  για να συμβολίσουμε το αλφάβητο και συχνά το αλφάβητο είναι το  $\{a, b\}$  ή το  $\{0, 1\}$ . Επίσης, αν και γίνεται κατάχρηση του όρου, αναφέρουμε συνήθως συμβολοσειρές στο αλφάβητο  $\{0, 1\}$  σα δυαδικές συμβολοσειρές. Μια μοναδιαία γλώσσα είναι ορισμένη σε κάποιο αλφάβητο που περιέχει ένα μόνο σύμβολο. Μήκος της συμβολοσειράς είναι ο αριθμός των συμβόλων που περιέχει. Η κενή συμβολοσειρά έχει μήκος 0: πρόκειται για συμβολοσειρά χωρίς σύμβολα και συμβολίζεται με  $\epsilon$ .

**Παράδειγμα 2.1.3.** Όλες οι δυαδικές συμβολοσειρές που ξεκινούν με 0. Η ιδέα είναι να διαβάζουμε τα δύο πρώτα σύμβολα, κι έτσι έχουμε την απάντηση. Η υπόλοιπη συμβολοσειρά αγνοείται (που αντιστοιχεί στο ότι το αυτόματο δεν αλλάζει ποτέ κατάσταση).

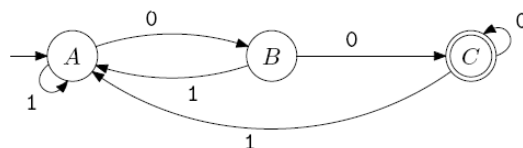
Για να γλυτωσουμε χώρο στο διάγραμμα, μπορούμε να επιγράψουμε κάποιο βέλος με πολλά σύμβολα μαζί χωρισμένα με κόμματα.

Αν μάς ενδιαφέρουν τα δύο τελευταία σύμβολα της συμβολοσειράς, τότε ο σχεδιασμός είναι δυσκολότερος.

**Παράδειγμα 2.1.4.** Όλες οι δυαδικές συμβολοσειρές που καταλήγουν με 0. Η πρώτη σκέψη μπορεί να είναι τι γίνεται αν τα δύο πρώτα σύμβολα είναι 0. Αυτό θα πρέπει να οδηγεί σε



τελική κατάσταση. Επομένως, θα πρέπει να υπάρχουν τουλάχιστον τρεις καταστάσεις:  $A$ ,  $B$  και  $C$ ,  $A$  θα είναι η αρχική κατάσταση,  $C$  θα είναι μια τελική κατάσταση, και ένα 0 θα οδηγεί από την κατάσταση  $A$  στη  $B$ , και από τη  $B$  στη  $C$ . Τι γίνεται όμως αν διαβάσουμε ένα 1; Κανονικά, θα έπρεπε να πηγαίνουμε πίσω στην αρχική κατάσταση, ανεξάρτητα από την κατάσταση που είμαστε. Μια άλλη ερώτηση είναι, τι γίνεται αν διαβάσουμε 0 όταν είμαστε στην κατάσταση  $C$ ; Μάλλον να παραμείνουμε στη  $C$ . Έτσι, καταλήγουμε στο παρακάτω FA:



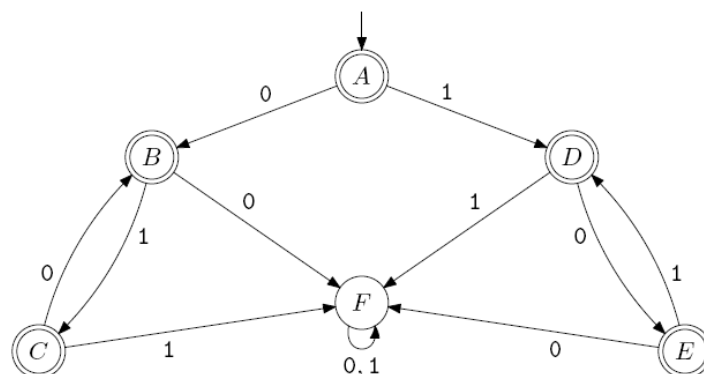
Για να διασπιστώσουμε ότι πράγματι δουλεύει, αρκεί να καταλάβουμε τι σημαίνει το να είμαστε σε κάθε κατάσταση. Η κατάσταση  $C$  σημαίνει ότι τα προηγούμενα δύο σύμβολα ήταν 00, η κατάσταση  $A$  σημαίνει ότι το προηγούμενο σύμβολο ήταν 1, ενώ η κατάσταση  $B$  σημαίνει ότι τα τελευταία δύο σύμβολα ήταν 10.

Ένας κοινός τύπος κατάστασης είναι η αποκαλούμενη παγίδα. Αυτή είναι μια κατάσταση από την οποία, αν βρεθούμε, δεν μπορούμε να φύγουμε. Για παράδειγμα, η κατάσταση  $C$  του παραδείγματος 2.1.1 είναι παγίδα. Πώς μπορούμε να αναγνωρίζουμε τις καταστάσεις - παγίδες από το διάγραμμα; Δεν έχουν εξερχόμενα βέλη. Οι παγίδες μπορούν να χρησιμοποιηθούν με δύο τρόπους:

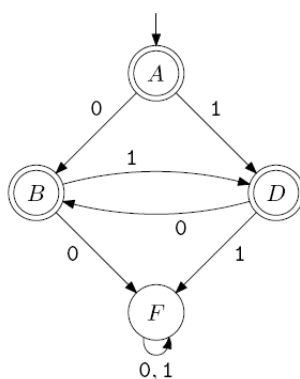
- (i) σαν καταστάσεις απόρριψης για συμβολοσειρές που έχουν διαβαστεί μερικώς αλλά ποτέ δεν θα γίνουν αποδεκτές, ή
- (ii) σαν τελικές καταστάσεις για συμβολοσειρές που έχουν διαβαστεί μερικώς αλλά σίγουρα θα γίνουν αποδεκτές.

**Παράδειγμα 2.1.5.** Κατασκευάστε ένα FA που να αποδέχεται όλες τις δυαδικές συμβολοσειρές στις οποίες τα 0 και τα 1 εμφανίζονται εναλλάξ. Η βασική ιδέα είναι ένα ζεύγος καταστάσεων μεταξύ των οποίων κινείται το αυτόματο: ένα 0 μεταφέρει το αυτόματο στη μία από αυτές τις καταστάσεις ενώ ένα 1 το οδηγεί πίσω στην άλλη κατάσταση. Επίσης, χρειαζόμαστε μία κατάσταση παγίδα για

την περίπτωση που δύο διαδοχικά σύμβολα είναι ίδια. Μία προσέγγιση είναι να έχουμε μία αρχική κατάσταση η οποία οδηγεί σε δύο επιμέρους αυτόματα. Η αντίστοιχη λύση είναι:



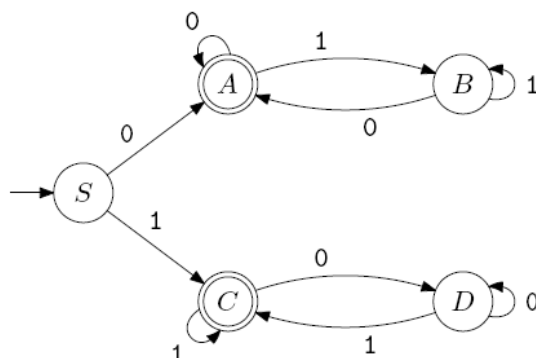
Φυσικά μπορούμε να κατασκευάσουμε ένα αυτόματο με λιγότερες καταστάσεις:



Άρα, ενώ ένα FA αναγνωρίζει μία μόνο γλώσσα, για δεδομένη γλώσσα μπορούν να υπάρχουν αρκετά FA που την αναγνωρίζουν. Πρέπει πάντα να δίνουμε προσοχή στην κενή συμβολοσειρά: θα έπρεπε το FA να αποδέχεται την  $\epsilon$  ή όχι. (ή αλήθεια είναι ότι αυτό δεν προκύπτει σαφώς από τη δοσμένη περιγραφή της γλώσσας.) Στο προηγούμενο παράδειγμα, η κενή συμβολοσειρά είναι αποδεκτή αφού η αρχική κατάσταση είναι και τελική. Μια άλλη χρήσιμη τεχνική είναι να θυμόμαστε συγκεκριμένα σύμβολα. Στο επόμενο παράδειγμα, πρέπει να θυμόμαστε το πρώτο σύμβολο. Επομένως, το FA χωρίζεται σε δύο επιμέρους μετά την κατανάλωση του πρώτου συμβόλου.

**Παράδειγμα 2.1.6.** Δώστε ένα FA που να αποδέχεται όλες τις δυαδικές συμβολοσειρές που ξεκινούν και καταλήγουν με το ίδιο σύμβολο. Προφανώς, το αυτόματο πρέπει να πηγαίνει σε δύο διαφορετικές καταστάσεις με βάση το πρώτο σύμβολο που θα διαβάσει. Μετά, πρέπει να ελέγχουμε το τρέχον σύμβολο που διαβάζεται, για την περίπτωση που είναι το τελευταίο σύμβολο της συμβολοσειράς.

Παρατηρούμε ότι δεν πρέπει να υπάρχει σύνδεση μεταξύ του πάνω και του κάτω μέρους, γιατί διαφορετικά το FA μπορεί να μη θυμάται το πρώτο σύμβολο που διάβασε.



**Για εξάσκηση:** Κατασκευάστε FA για κάθε μία από τις παρακάτω τρεις γλώσσες:

1. Όλες οι δυαδικές συμβολοσειρές που περιέχουν τουλάχιστον ένα 0
2. Όλες οι δυαδικές συμβολοσειρές που περιέχουν το πολύ ένα 0
3. Όλες οι δυαδικές συμβολοσειρές που ξεκινούν και καταλήγουν σε 0 (συμπεριλαμβάνεται και η συμβολοσειρά που περιέχει ένα μόνο 0)

### 2.1.2 Αναπαράσταση FAs

Ένα διάγραμμα δεν είναι ο μόνος τρόπος για να αναπαραστήσουμε ένα FA. Μάλιστα, για να δώσουμε ένα FA σαν είσοδο σε κάποιο πρόγραμμα πρέπει να έχουμε έναν άλλο τρόπο αναπαράστασής του. Ένα τέτοιος τρόπος είναι ο πίνακας μεταβάσεων: ένας πίνακας που δείχνει ποια είναι η επόμενη κατάσταση με βάση την τρέχουσα κατάσταση και το σύμβολο που διαβάζεται.

**Παράδειγμα 2.1.7.** Συνέχεια παραδείγματος 2.1.6. Παρακάτω δίνεται ο πίνακας μεταβάσεων για το FA που αποδέχεται όλες τις δυαδικές συμβολοσειρές που ξεκινούν και καταλήγουν στο ίδιο σύμβολο.

	Είσοδος	
	0	1
S	A	C
A	A	B
B	A	B
C	D	C
D	D	C

Στη συνέχεια, θα δούμε μια πιο γενική μορφή ενός FA. Τα FA που έχουμε δει μέχρι τώρα είναι ντετερμινιστικά (και συνήθως αποκαλούνται για συντομία DFA). Μπορούμε να ορίσουμε τα ντετερμινιστικά πεπερασμένα αυτόματα σαν μια 5-άδα  $(Q, \Sigma, \delta, q_0, F)$  όπου:

- $Q$  είναι ένα πεπερασμένο σύνολο καταστάσεων
- $\Sigma$  είναι το αλφάβητο εισόδου
- $\delta$  είναι η συνάρτηση μεταβάσεων που απεικονίζει μια κατάσταση και ένα σύμβολο σε μια κατάσταση. (Συμβολίζουμε  $\delta : Q \times \Sigma \rightarrow Q$ .) Για παράδειγμα,  $\delta(r, 1) = s$  σημαίνει ότι το αυτόματο είναι στην κατάσταση  $r$  και διαβάσει το σύμβολο 1 μεταβαίνει στην κατάσταση  $s$ .
- $q_0$  είναι η αρχική κατάσταση
- $F$  είναι ένα υποσύνολο του  $Q$  που περιέχει τις τελικές καταστάσεις

Για κάθε κατάσταση  $p \in K$  και κάθε είσοδο  $\alpha \in \Sigma$ , η επόμενη κατάσταση  $q = \delta(p, \alpha)$  ορίζεται μοναδικά. Επομένως, είναι δυνατόν να οριστεί η κατάσταση στην οποία θα μεταβεί το αυτόματο από κάθε παρούσα κατάσταση  $p \in K$  όταν δεχτεί σαν είσοδο τη συμβολοσειρά  $w \in \Sigma^*$ , ακολουθώντας το μονοπάτι που καθορίζεται από τη  $w$ . Πρακτικά, αυτό γίνεται με τον ορισμό της συνάρτησης  $\delta^* : K \times \Sigma^* \rightarrow K$ .

**Ορισμός 2.1.2.** Δεδομένου ενός DFA  $D = (K, \Sigma, \delta, q_0, F)$ , η συνάρτηση μετάβασης  $\delta^* : K \times \Sigma^* \rightarrow K$  ορίζεται ως εξής:

$$\delta^*(p, \epsilon) = p,$$

$$\delta^*(p, u\alpha) = \delta(\delta^*(p, u), \alpha),$$

όπου  $\alpha \in \Sigma$  και  $u \in \Sigma^*$ .

Άμεσα προκύπτει από τον παραπάνω ορισμό ότι  $\delta^*(p, \alpha) = \delta(p, \alpha)$  για  $\alpha \in \Sigma$ . Η σημασία της έκφρασης  $\delta^*(p, w)$  είναι ότι πρόκειται για την κατάσταση στην οποία μπορεί να μεταβεί το αυτόματο από την κατάσταση  $p$  ακολουθώντας το μονοπάτι που ξεκινάει από την  $p$  και καθορίζεται από την  $w$ . Εύκολα επίσης προκύπτει ότι

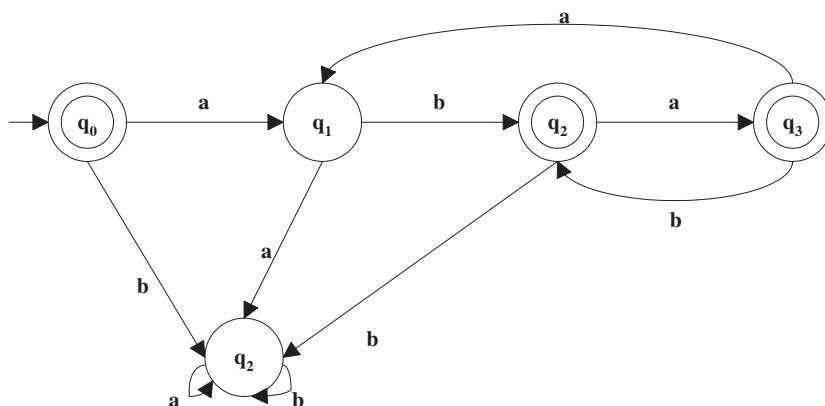
$$\delta^*(p, uv) = \delta^*(\delta^*(p, u), v).$$

Στη συνέχεια ορίζεται ο τρόπος με τον οποίο ένα DFA δέχεται ή απορρίπτει συμβολοσειρές.

**Ορισμός 2.1.3.** Δεδομένου ενός DFA  $D = (K, \Sigma, \delta, q_0, F)$ , η γλώσσα  $L(D)$  που γίνεται δεκτή (ή αναγνωρίζεται) από το αυτόματο  $D$  είναι η

$$L(D) = \{w \in \Sigma^* | \delta^*(q_0, w) \in F\}.$$

Επομένως, μια συμβολοσειρά  $w \in \Sigma^*$  είναι αποδεκτή αν και μόνον αν το μονοπάτι από την  $q_0$  με είσοδο  $w$  καταλήγει σε τελική κατάσταση.



**Παράδειγμα 2.1.8.** Θεωρούμε τη γλώσσα  $L = (ab \cup aba)^*$ . Η γλώσσα αυτή αναγνωρίζεται από το παρακάτω DFA:

Έτσι είναι σαφές τι πρέπει να ορίζουμε προκειμένου να περιγράψουμε πλήρως ένα FA.

**Παράδειγμα 2.1.9.** Συνέχεια παραδείγματος 2.1.6. Για το προηγούμενο παράδειγμα, η 5-άδα είναι  $(Q, \Sigma, \delta, q_0, F)$  με

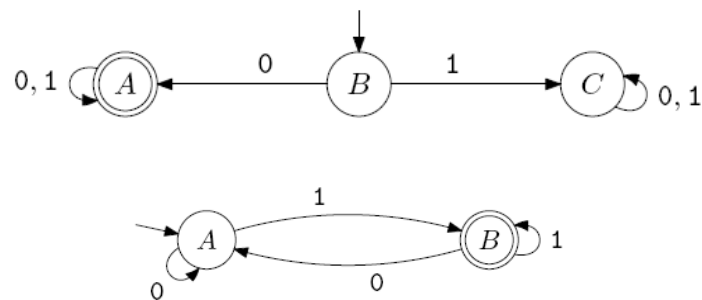
- $Q = \{S, A, B, C, D\}$
- $\Sigma = \{0, 1\}$
- η συνάρτηση  $\delta$  φαίνεται στον προηγούμενο πίνακα μεταβάσεων
- $q_0 = S$
- $F = \{A, C\}$

### Ασκήσεις

1. Κατασκευάστε FA που να δέχεται μόνο τη συμβολοσειρά 0110.
2. Κατασκευάστε FA που να αποδέχεται δυαδικές συμβολοσειρές μήκους 3.
3. Αν στο παράδειγμα 2.1.1 κάνουμε τελικές καταστάσεις και τη  $B$  και τη  $C$ , ποια θα είναι η γλώσσα που αποδέχεται το FA;
4. Για τα ακόλουθα δύο FA, πείτε αν οι συμβολοσειρές 0110, 1, 1011010 και 00000 γίνονται αποδεκτές.

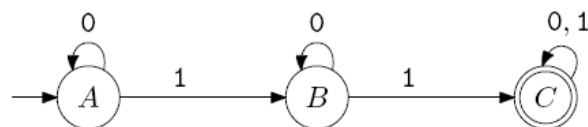
(α')





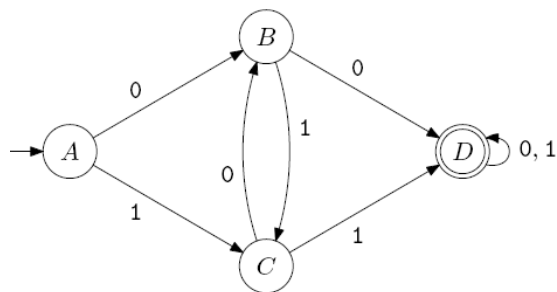
(β')

5. Περιγράψτε τη γλώσσα που δέχεται καθένα από τα προηγούμενα δύο FA.
6. Κατασκευάστε ένα FA που να αποδέχεται κάθε συμβολοσειρά με 0 και 1.
7. Για το παρακάτω FA, πείτε αν οι συμβολοσειρές 0110, 1, 1011010 και 00000 γίνονται αποδεκτές.

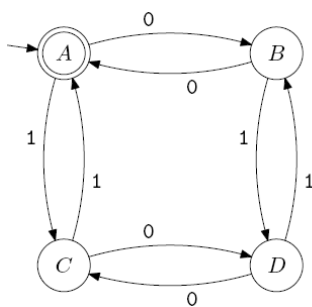


8. Περιγράψτε τη γλώσσα που δέχεται το παραπάνω FA.
9. Κατασκευάστε FA για κάθε μια από τις παρακάτω γλώσσες:
  - (α') Όλες οι δυαδικές συμβολοσειρές που περιέχουν τουλάχιστον τρία 1
  - (β') Όλες οι δυαδικές συμβολοσειρές που περιέχουν περιττό αριθμό από 1
  - (γ') Όλες οι δυαδικές συμβολοσειρές που δεν περιέχουν την υποσυμβολοσειρά 111
  - (δ') Όλες οι δυαδικές συμβολοσειρές στις οποίες κάθε περιττή θέση είναι 1
10. Κατασκευάστε ένα FA που να αποδέχεται όλες τις δυαδικές συμβολοσειρές που περιέχουν ακριβώς τρία 1.
11. Κατασκευάστε ένα FA που να αποδέχεται όλες τις συμβολοσειρές στο αλφάβητο  $\{a, b, c\}$  που περιέχουν περιττό αριθμό  $a$ .
12. Κατασκευάστε ένα FA για τη γλώσσα όλων των δυαδικών συμβολοσειρών που έχουν τουλάχιστον τρία σύμβολα και το πρώτο και τελευταίο σύμβολό τους είναι διαφορετικά.
13. Κατασκευάστε ένα FA του οποίου η γλώσσα είναι το σύνολο των συμβολοσειρών που αποτελούνται από  $a$ ,  $b$  και  $c$  και περιέχουν την υποσυμβολοσειρά  $abc$ .

14. Κατασκευάστε ένα FA που να αποδέχεται όλες τις συμβολοσειρές στο αλφάβητο  $\{a, b\}$  που περιέχουν τις υποσυμβολοσειρές  $ab$  ή  $bba$  (ή και τις δύο).
15. Κατασκευάστε ένα FA που να αποδέχεται όλες τις συμβολοσειρές στο αλφάβητο  $\{a, b, c\}$  των οποίων τα σύμβολα είναι αλφαβητικά διαταγμένα. (Για παράδειγμα, οι συμβολοσειρές  $aaabcc$  και  $ac$  είναι αποδεκτές ενώ οι  $abca$  και  $cb$  δεν είναι.)
16. Κατασκευάστε ένα FA που να αποδέχεται όλες τις δυαδικές συμβολοσειρές με άρτιο αριθμό 0 και αριθμό 1 που να είναι πολλαπλάσιο του 3.
17. Εξηγήστε τι αποδέχεται το παρακάτω FA:



18. Εξηγήστε τι αποδέχεται το παρακάτω FA:



## 2.2 Κανονικές Εκφράσεις (Regular Expressions)

Συχνά πρέπει να αναγνωρίζουμε συμβολοσειρές συγκεκριμένου τύπου. Για παράδειγμα, υποθέστε ότι αναζητούμε μία μέθοδο ή διαδικασία για να αναγνωρίζουμε αριθμούς με δεκαδικό μέρος. Για να το κάνουμε αυτό, πρέπει να είμαστε σαφείς με το τι εννοούμε λέγοντας ‘αριθμούς με δεκαδικό μέρος’. Μπορούμε να τους ορίσουμε ως εξής: ‘Ψηφία ακολουθούμενα ίσως από μία τελεία και άλλα ψηφία.’ Ή καλύτερα: ‘προαιρετικά το σύμβολο μείον, οποιαδήποτε ακολουθία ψηφίων, προαιρετικά μια τελεία, και τέλος προαιρετικά μια ακολουθία ψηφίων.’ Για να είμαστε ακριβείς, μπορούμε να γράψουμε την περιγραφή ενός αριθμού με δεκαδικό μέρος με μια κανονική έκφραση:

$$(- + \epsilon)DD^*(\epsilon + .D^*)$$

όπου το  $D$  είναι κάποιο ψηφίο.

Μια κανονική έκφραση (regular expression (RE)) αντιστοιχεί σε ένα σύνολο συμβολοσειρών, δηλαδή περιγράφει μια γλώσσα. Κατασκευάζεται με χρήση τριών κανονικών πράξεων, της ένωσης, της παράθεσης και της πράξης star, που περιγράφονται στη συνέχεια. Μια κανονική έκφραση περιέχει συνηθισμένα σύμβολα και επιπλέον τέσσερα ειδικά σύμβολα:

$$+ * ()$$

Μια κανονική έκφραση ερμηνεύεται με βάση τους παρακάτω κανόνες:

- Οι παρενθέσεις ( και ) χρησιμοποιούνται για ομαδοποίηση, όπως στα συνηθισμένα μαθηματικά.
- Το σύμβολο  $+$  σημαίνει ένωση. Επομένως γράφοντας

$$0 + 1$$

εννοούμε είτε το σύμβολο μηδέν είτε το σύμβολο ένα. Θα αναφερόμαστε στην πράξη  $+$  και σαν *ή* δηλαδή διάζευξη.

- Η παράθεση δύο εκφράσεων προκύπτει αν γράψουμε τη μία έκφραση μετά την άλλη χωρίς κενό μεταξύ τους. Για παράδειγμα, η έκφραση

$$(0 + 1)0$$

περιγράφει τις συμβολοσειρές 00 και 10.

- Το σύμβολο  $\epsilon$  αναπαριστά την κενή συμβολοσειρά. Επομένως, η κανονική έκφραση

$$(0 + 1)(0 + \epsilon)$$

περιγράφει τις τέσσερις συμβολοσειρές 00, 0, 10, 1

- Το σύμβολο  $*$  σημαίνει κανένα ή περισσότερα αντίγραφα. Για παράδειγμα, η κανονική έκφραση

$$a^*$$

περιγράφει οποιαδήποτε συμβολοσειρά που περιέχει  $a$ :  $\{\epsilon, a, aa, aaa, \dots\}$ . Επίσης, η κανονική έκφραση

$$(0 + 1)^*$$

περιγράφει όλες τις δυαδικές συμβολοσειρές.

**Παράδειγμα 2.2.1.** Η κανονική έκφραση

$$(01)^*$$

περιγράφει το σύνολο που περιέχει τις συμβολοσειρές  $\epsilon, 01, 0101, 010101, \dots$

**Παράδειγμα 2.2.2.** Ποια είναι μια κανονική έκφραση για τη γλώσσα όλων των δυαδικών συμβολοσειρών με μήκος τουλάχιστον 2 που ξεκινούν και καταλήγουν με το ίδιο σύμβολο; Φαίνεται παρακάτω:

$$0(0 + 1)^*0 + 1(0 + 1)^*1$$

Παρατηρήστε ότι η σειρά προτεραιότητας των κανονικών πράξεων είναι πρώτα η πράξη star, μετά η παράθεση και μετά η ένωση. Δηλαδή, η πράξη star αναφέρεται πάντα στο μικρότερο δυνατό τμήμα ενώ η ένωση στο μεγαλύτερο.

**Παράδειγμα 2.2.3.** Θεωρήστε την κανονική έκφραση

$$(0 + 1)^*1 + \epsilon)(00)^*00$$

Η γλώσσα που περιγράφει αυτή η κανονική έκφραση είναι το σύνολο των δυαδικών συμβολοσειρών που καταλήγουν με άρτιο, μη μηδενικό αριθμό από 0.

Γενικά, αν κατασκευάσουμε μια κανονική έκφραση από την ένωση δύο άλλων, έστω  $R$  και  $S$ , τότε η γλώσσα που προκύπτει είναι η ένωση των γλωσσών που περιγράφουν οι κανονικές εκφράσεις  $R$  και  $S$ . Γι' αυτό και η πράξη  $+$  λέγεται ένωση.

Αν κατασκευάσουμε μια κανονική έκφραση από την παράθεση δύο άλλων, έστω  $R$  και  $S$ , τότε η γλώσσα που προκύπτει περιέχει όλες τις συμβολοσειρές που μπορούν να κατασκευαστούν παίρνοντας μία συμβολοσειρά από τη γλώσσα της κανονικής έκφρασης  $R$  και μια συμβολοσειρά από τη γλώσσα της  $S$  και παραθέτοντάς τες.

Αν κατασκευάσουμε μια κανονική έκφραση από το star μιας κανονικής έκφρασης  $R$ , τότε η γλώσσα που προκύπτει περιέχει όλες τις συμβολοσειρές που μπορούν να κατασκευαστούν παίρνοντας αυθαίρετο αριθμό συμβολοσειρών από τη γλώσσα της  $R$  (δεν είναι απαραίτητα διαφορετικές αλλά ούτε απαραίτητα ίδιες), και παραθέτοντάς τες. Επομένως, μπορούμε πράγματι να αναφερόμαστε σε ένωση, παράθεση και star γλωσσών. Για παράδειγμα, ένας αναδρομικός ορισμός του star μιας γλώσσας είναι ο εξής:

**Ορισμός 2.2.1.** Αναδρομικός ορισμός της  $L^*$ . (1)  $\epsilon \in L^*$ . (2) Αν  $x \in L^*$  και  $y \in L$  τότε και  $xy \in L^*$ .

**Παράδειγμα 2.2.4.** Αν η γλώσσα  $L$  είναι η  $\{ma, pa\}$  και η γλώσσα  $M$  η  $\{be, bor\}$ , τότε η ένωσή τους  $L + M$  είναι η γλώσσα  $\{ma, pa, be, bor\}$ , η παράθεσή τους  $LM$  είναι η γλώσσα  $\{mabe, mabor, pabe, pabor\}$  και η  $L^*$  είναι η γλώσσα  $\{\epsilon, ma, pa, mama, \dots, pamama, \dots\}$ .

Επιπλέον, θα χρησιμοποιήσουμε τον παρακάτω συμβολισμό:

Αν  $\Sigma$  είναι κάποιο αλφάβητο, τότε  $\Sigma^*$  είναι το σύνολο των συμβολοσειρών που χρησιμοποιούν το αλφάβητο αυτό.

**Παράδειγμα 2.2.5.** 1. Όλες οι συμβολοσειρές που αποτελούνται από 0 και 1.

$$(0 + 1)^*$$

2. Όλες οι συμβολοσειρές που αποτελούνται από  $a$  και  $b$  και έχουν άρτιο μήκος.

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

3. Όλες οι συμβολοσειρές που αποτελούνται από  $a$  και  $b$  και έχουν περιττό μήκος.

$$(a + b)((a + b)(a + b))^* \\ ((a + b)(a + b))^*(a + b)$$

4. Όλες οι συμβολοσειρές που αποτελούνται από  $a$  και  $b$  και έχουν μήκος 3.

$$(a + b + \epsilon)(a + b + \epsilon)(a + b + \epsilon)$$

**Παρατήρηση** ο συμβολισμός  $(a + b + \epsilon)^3$  είναι σωστός αλλά δεν είναι κανονική έκφραση.

5. Όλες οι συμβολοσειρές που αποτελούνται από  $a$  και  $b$  και περιέχουν το  $aa$ .

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

6. Όλες οι συμβολοσειρές που αποτελούνται από  $a$  και  $b$  και δεν περιέχουν το  $aa$ .

$$(a + \epsilon)(b + ba)^* \\ (a + \epsilon)(bb^*a)^*b^*$$

7. Όλες οι συμβολοσειρές που αποτελούνται από  $a, b$  και  $c$  και στις οποίες τα  $a$  προηγούνται των  $b$  και τα  $b$  προηγούνται των  $c$ .

$$a^*b^*c^*$$

8. Όλες οι μη κενές συμβολοσειρές που αποτελούνται από  $a, b$  και  $c$  και στις οποίες τα  $a$  προηγούνται των  $b$  και τα  $b$  προηγούνται των  $c$ .

$$aa^*b^*c^* + a^*bb^*c^* + a^*b^*cc^* \\ ((aa^+b)b^* + c)c^*$$

### 2.2.1 Το θεώρημα του Kleene

Εύκολα μπορούμε να κατασκευάσουμε ένα αυτόματο που να αναγνωρίζει μια συμβολοσειρά αν και μόνον αν η συμβολοσειρά αυτή είναι στη μορφή που απαιτείται από τις κανονικές εκφράσεις των προηγούμενων παραδειγμάτων. Συμβαίνει το ίδιο όμως με περισσότερο πολύπλοκες κανονικές εκφράσεις; Ναι, υπάρχει συγκεκριμένος αλγόριθμος που κάνει αυτή ακριβώς τη δουλειά και επομένως πάντα μπορούμε να κατασκευάσουμε ένα πεπερασμένο αυτόματο για δεδομένη κανονική έκφραση.

**Παράδειγμα 2.2.6.** Κατασκευάστε FA για τη γλώσσα που περιγράφεται από την κανονική έκφραση

$$(0 + 1)^*00(0 + 1)^*$$

Το FA είναι αυτό του παραδείγματος 2.1.1.

Στη συνέχεια, θα αποδείξουμε το ακόλουθο θεώρημα:

**Θεώρημα 2.2.1.** (Θεώρημα του Kleene). Υπάρχει FA για μια γλώσσα αν και μόνον αν υπάρχει κανονική έκφραση για τη γλώσσα αυτή.

Το θεώρημα αυτό δεν έχει μόνο θεωρητική σημασία: γιατί, είναι συχνά απλό να περιγράψουμε ένα πρόβλημα αναγνώρισης με μια κανονική έκφραση, και είναι απλό να γράψουμε ένα πρόγραμμα που να εξομοιώνει ένα FA. Το θεώρημα, και ο αντίστοιχος αλγόριθμος, δίνει τη δυνατότητα να μετατρέπουμε μια κανονική έκφραση σε ένα FA. Θα χρησιμοποιούμε τον όρο κανονική γλώσσα για κάθε γλώσσα που γίνεται αποδεκτή από κάποιο FA ή περιγράφεται από κάποια κανονική έκφραση.

### 2.2.2 Εφαρμογές των REs

Αναφέρουμε στη συνέχεια μερικές χαρακτηριστικές εφαρμογές των πεπερασμένων αυτομάτων και των κανονικών εκφράσεων.

Οι κανονικές εκφράσεις χρησιμοποιούνται στο σχεδιασμό μεταγλωττιστών για να περιγράψουν τμήμα της γλώσσας. Για παράδειγμα, στην Pascal ή τη Java ένας ακέραιος αριθμός έχει συγκεκριμένη μορφή, ένας πραγματικός αριθμός έχει συγκεκριμένη μορφή, κτλ. Η κανονική έκφραση για

κάθε τέτοιο τμήμα μπορεί αυτόματα να μετατραπεί σε FA που την αναγνωρίζει. Ένας σαρωτής ή αλλιώς tokenizer είναι ένα πρόγραμμα που σαρώνει την είσοδο και καθορίζει και προσδιορίζει το επόμενο τμήμα. Ένας tokenizer παρέχεται σα βασικό τμήμα στη Java και το Unix.

Οι κανονικές εκφράσεις χρησιμοποιούνται επίσης για αναζήτηση σε κάποιο αρχείο. Για παράδειγμα, υποθέστε ότι έχουμε ένα πολύ μεγάλο αρχείο με δεκαδικούς αριθμούς που στο τέλος του καθενός υπάρχει το  $h$  για να χωρίζονται μεταξύ τους, π.χ.,  $273h$  ή  $22h$ , και μάς ζητάνε να τροποποιήσουμε το αρχείο ώστε κάθε αριθμός να έχει μπροστά ένα  $\#$ , π.χ.,  $\#273$  ή  $\#22$ . Αν διαθέτουμε κάποιον τρόπο για να εκτελέσουμε αναζήτηση και αντικατάσταση, θα μπορούσαμε να κάνουμε το εξής:

αναζήτηση:  $([0123456789]+)h$

αντικατάσταση με:  $\#\backslash 1$

Στη γραμμή για την αναζήτηση, το σύμβολο  $+$  έχει τη σημασία του  $*$  δηλαδή σημαίνει ένα ή περισσότερα αντίγραφα. Οι αγκύλες καθορίζουν το σύνολο των συμβόλων ενώ οι παρενθέσεις καθορίζουν μια έκφραση. Στη γραμμή για την αντικατάσταση, το  $\backslash 1$  σημαίνει την πρώτη έκφραση στη γραμμή της αναζήτησης.

**Για εξάσκηση...** Δώστε κανονική έκφραση για τις ακόλουθες τρεις γλώσσες:

1. Όλες οι δυαδικές συμβολοσειρές που περιέχουν τουλάχιστον ένα 0
2. Όλες οι δυαδικές συμβολοσειρές που περιέχουν το πολύ ένα 0
3. Όλες οι δυαδικές συμβολοσειρές που ξεκινούν και καταλήγουν με 0

### Ασκήσεις

1. Για κάθε κανονική έκφραση, πείτε ποιες από τις ακόλουθες συμβολοσειρές ανήκουν στη γλώσσα της κανονικής έκφρασης:  $\epsilon, abba, bababbandbaaaaa$ .

$$(\alpha') (a+b)^*ab(a+b)^*$$

$$(\beta') b^*ab^*ab^*$$

$$(\gamma') a + (a^*b)^*$$

2. Για κάθε κανονική έκφραση, δώστε δύο συμβολοσειρές που να ανήκουν στην αντίστοιχη γλώσσα και δύο που να μην ανήκουν:

$$(\alpha') a(a+b)^*b$$

$$(\beta') a^*a + " + b^*$$

$$(\gamma') (ab+ba)^*$$

3. Δώστε κανονικές εκφράσεις για τις παρακάτω γλώσσες:
  - (α') Όλες οι δυαδικές συμβολοσειρές που περιέχουν ακριβώς δύο 1
  - (β') Όλες οι δυαδικές συμβολοσειρές που περιέχουν διπλό σύμβολο (δηλ. περιέχουν είτε την 00 είτε την 11)
  - (γ') Όλες οι δυαδικές συμβολοσειρές που περιέχουν και την 00 και την 11
  - (δ') Όλες οι δυαδικές συμβολοσειρές που δεν περιέχουν καθόλου διπλά σύμβολα
4. Κατασκευάστε ένα FA που να αναγνωρίζει δεκαδικούς αριθμούς με βάση την περιγραφή που παρουσιάστηκε στην αρχή της ενότητας.
5. Δώστε κανονική έκφραση για τη γλώσσα που περιέχει όλες τις δυαδικές συμβολοσειρές με μήκος τουλάχιστον 2 που αρχίζουν και τελειώνουν με το ίδιο σύμβολο.
6. Δώστε κανονικές εκφράσεις και FA για τις παρακάτω γλώσσες που είναι ορισμένες στο αλφάβητο  $\{a, b\}$ :
  - (α') Όλες οι συμβολοσειρές που περιέχουν την υποσυμβολοσειρά  $aaa$
  - (β') Όλες οι συμβολοσειρές που δεν περιέχουν την υποσυμβολοσειρά  $aaa$
  - (γ') Όλες οι συμβολοσειρές που δεν καταλήγουν σε  $aaa$
  - (δ') Όλες οι συμβολοσειρές που περιέχουν ακριβώς 3  $a$ .
  - (ε') Όλες οι συμβολοσειρές που περιέχουν αριθμό  $a$  που διαιρείται ακριβώς με 3.
7. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.12.
8. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.13.
9. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.14.
10. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.15.
11. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.16.
12. Δώστε κανονική έκφραση για τη γλώσσα της άσκησης 1.19.
13. Δώστε κανονική έκφραση για το συμπλήρωμα της κανονικής έκφρασης 111 στο αλφάβητο  $\{0, 1\}$ . Δηλαδή, η κανονική έκφραση θα πρέπει να περιγράφει  $\text{κα}\bullet\text{θε}$  δυαδική συμβολοσειρά  $\text{ακτός από την 111}$ .
14. Κάθε κανονική γλώσσα αντιστοιχεί σε μία μόνο κανονική έκφραση; Κάθε κανονική έκφραση αντιστοιχεί σε μία μόνο κανονική γλώσσα; Εξηγήστε.



15. Δείξτε ότι η γλώσσα της κανονικής έκφρασης  $(0^*1^*)^*$  περιέχει όλες τις δυαδικές συμβολοσειρές.
16. Απλοποιήστε της ακόλουθες κανονικές εκφράσεις (δηλ., βρείτε μια απλούστερη κανονική έκφραση που να περιγράφει την ίδια γλώσσα):
  - (α')  $(r + \epsilon)^*$
  - (β')  $ss^* + \epsilon$
  - (γ')  $(\epsilon + r)(r + s)^*(\epsilon + r)$
  - (δ')  $(r + s + rs + sr)^*$
17. Σε μια συμβολοσειρά, ένα block είναι μια μέγιστου μήκους υποσυμβολοσειρά (που δεν μπορεί να μεγεθυνθεί) της οποίας όλα τα σύμβολα είναι τα ίδια. Για παράδειγμα, η συμβολοσειρά 0001100 έχει τρία block. Έστω  $L$  η γλώσσα που περιέχει όλες τις δυαδικές συμβολοσειρές που κάθε block τους έχει μήκος 2 ή 3. Δώστε FA και κανονική έκφραση για τη  $L$ .
18. Συμβολίζουμε με  $C_n$  το σύνολο όλων των δυαδικών αριθμών που είναι πολλαπλάσια του  $n$ . Δείξτε ότι η γλώσσα  $C_7$  είναι κανονική. (Ιδέα: κατασκευάστε ένα FA.)
19. Δώστε ορισμό για κανονικές εκφράσεις της μορφής του ορισμού για τα πεπερασμένα αυτόματα.

## 2.3 Μη ντετερμινισμός

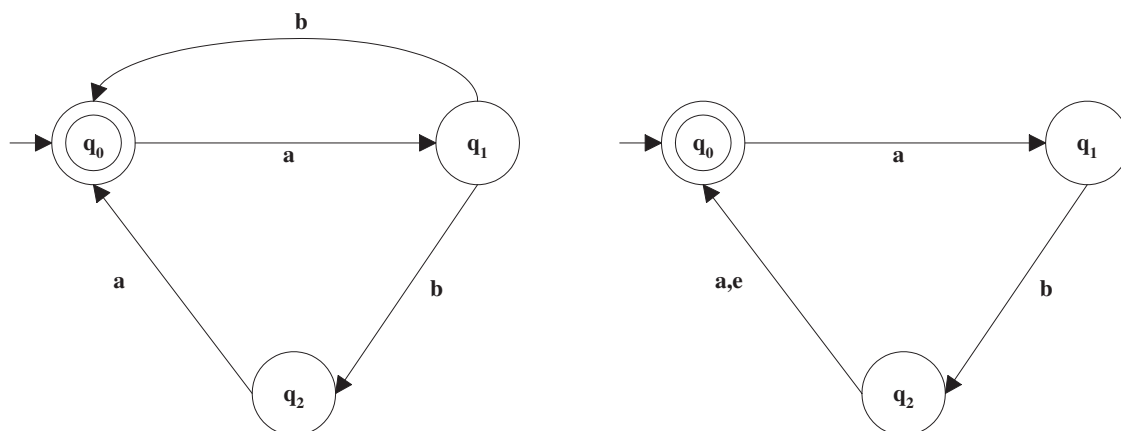
Τα FA που ορίσαμε και οι πραγματικοί υπολογιστές είναι ντετερμινιστικές μηχανές: το πρόγραμμα, δηλαδή, καθορίζει πλήρως μια μοναδική ενέργεια σε κάθε βήμα. Τι γίνεται όμως αν επιτρέψουμε στις μηχανές παραπάνω ευελιξία ή μη ντετερμινισμό; Αυτό φυσικά δεν έχει μόνο θεωρητικό ενδιαφέρον: η μετατροπή κανονικών εκφράσεων σε πεπερασμένα αυτόματα δεν μπορεί να γίνει χωρίς τη χρήση μη ντετερμινισμού...

Τα NFAs αποτελούν γενίκευση των DFAs χωρίς να αποτελούν ένα ισχυρότερο υπολογιστικό μοντέλο. Εν τούτοις σε αρκετές περιπτώσεις τα NFA παρέχουν έναν καλύτερο τρόπο αναπαράστασης των πεπερασμένων αυτομάτων. Τα ακόλουθα πεπερασμένα αυτόματα αναγνωρίζουν την ίδια γλώσσα αλλά είναι προφανώς απλούστερο να ακολουθήσουμε την πρώτη προσέγγιση [1]:

Για κάθε κατάσταση  $p \in K$  και κάθε είσοδο  $a \in \Sigma \cup \{\epsilon\}$ , το σύνολο των καταστάσεων  $\delta(p, a)$  ορίζεται με μοναδικό τρόπο. Γράφουμε  $q \in \delta(p, a)$ .

### 2.3.1 Μη ντετερμινιστικά πεπερασμένα αυτόματα

Ένα μη ντετερμινιστικό πεπερασμένο αυτόματο ορίζεται ως εξής: για κάθε κατάσταση μπορεί να υπάρχουν καμία, μία ή περισσότερες μεταβάσεις με κατανάλωση του ίδιου συμβόλου του αλφαβήτου.

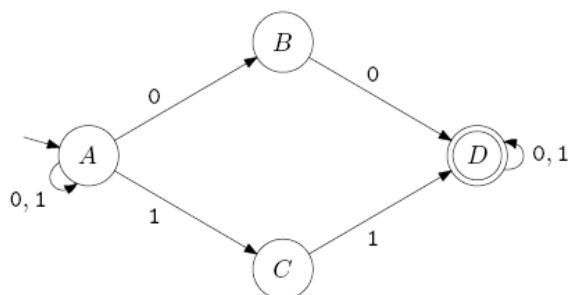


Αν το αυτόματο φτάσει σε κατάσταση από την οποία υπάρχουν πολλαπλές δυνατές μεταβάσεις με βάση το σύμβολο εισόδου, λέμε ότι το αυτόματο διακλαδίζεται. Αν το αυτόματο φτάσει σε κατάσταση από την οποία δεν υπάρχει μετάβαση, τότε ο αντίστοιχος κλάδος σταματά ή απορρίπτει την είσοδο. Επομένως είναι δυνατόν να υπάρχουν κλάδοι που αποδέχονται και άλλοι που απορρίπτουν την είσοδο. Πώς μπορούμε να γνωρίζουμε αν μια συμβολοσειρά ανήκει στη γλώσσα του αυτομάτου;

**Ορισμός 2.3.1.** Ένα μη ντετερμινιστικό πεπερασμένο αυτόματο (NFA) αποδέχεται τη συμβολοσειρά εισόδου αν υπάρχει κάποια ακολουθία μεταβάσεων που οδηγεί σε τελική κατάσταση.

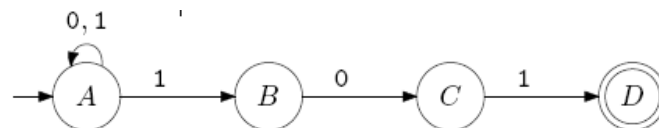
Παρατηρήστε ότι ο ορισμός δεν είναι συμμετρικός: ένας κλάδος που οδηγεί σε κατάσταση αποδοχής είναι αρκετός για να κάνει το αυτόματα συνολικά να αποδέχεται την είσοδο, αλλά κάθε κλάδος πρέπει να απορρίπτει μια συμβολοσειρά για να μπορούμε να πούμε ότι το αυτόματος συνολικά την απορρίπτει. Τέλος, οι μη ντετερμινιστικές μηχανές είναι μόνο μοντέλα υπολογισμού αφού δεν υπάρχουν στην πραγματικότητα (αν και σε αυτό θα μπορούσε κανείς και να διαφωνήσει).

**Παράδειγμα 2.3.1.** Τι αναγνωρίζει το παρακάτω NFA;



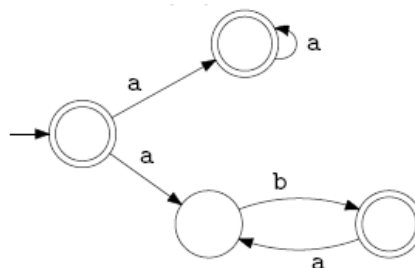
Για παράδειγμα, η συμβολοσειρά 0110 είναι αποδεκτή, γιατί το αυτόματο μπορεί να περάσει από τις καταστάσεις AACDD. Αλλά η 101010 δεν είναι αποδεκτή: κάθε κλάδος που φτάνει στις καταστάσεις B ή C διακόπτεται. Πράγματι, αυτό το NFA αποδέχεται κάθε δυαδική συμβολοσειρά που περιέχει τη 00 ή την 11.

**Παράδειγμα 2.3.2.** Ένα εύκολο μη ντετερμινιστικό πεπερασμένο αυτόματο που μπορούμε να φτιάξουμε είναι για τη γλώσσα που περιλαμβάνει όλες τις συμβολοσειρές με συγκεκριμένο τέλος. Παρακάτω φαίνεται ένα NFA που αποδέχεται όλες τις δυαδικές συμβολοσειρές που καταλήγουν σε 101. Για να αναγνωρίσει συμβολοσειρές της γλώσσας, το NFA παραμένει στην κατάσταση  $A$  μέχρι να μαντέψει μη ντετερμινιστικά και να μεταβεί στην κατάσταση  $B$  διαβάζοντας το σωστό 1.



Στο επόμενο παράδειγμα, χρησιμοποιούμε μη ντετερμινισμό για να αναζητήσουμε δύο πρότυπα ‘ταυτόχρονα’:

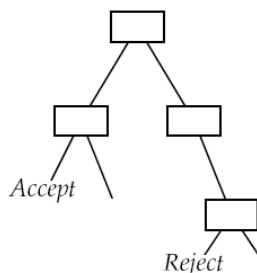
**Παράδειγμα 2.3.3.** Ένα NFA για τη γλώσσα  $a^* + (ab)^*$ :



### 2.3.2 Η έννοια του μη ντετερμινισμού

Υπάρχουν διάφοροι τρόποι να κατανοήσουμε την έννοια του μη ντετερμινισμού. Μπορούμε να θεωρήσουμε ότι το αυτόματο μαντεύει την επόμενη κίνηση: αυτή είναι η ιδέα ‘μάντεψε και επαλήθευσε’. Σύμφωνα με αυτή την ιδέα, υποθέτουμε ότι το αυτόματο είναι έξυπνο και πάντα μαντεύει σωστά σε ποια επόμενη κατάσταση πρέπει να μεταβεί. Πάντως, το αυτόματο μπορεί να κάνει όσες μαντεψιές θέλει, αλλά πρέπει να τις ‘επαληθεύσει’.

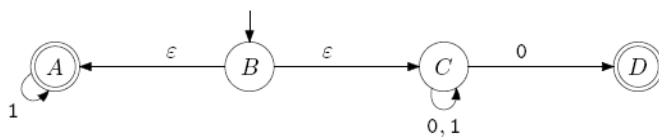
Στο παράδειγμα 2.3.1, το αυτόματο μαντεύει αν η αρχή του διπλού συμβόλου έχει διαβαστεί ή όχι και ελέγχει τις μαντεψιές διαβάζοντας τα επόμενα δύο σύμβολα. Αν η συμβολοσειρά δεν ανήκει στη γλώσσα τότε καμία ακολουθία από μαντεψιές δεν μπορεί να οδηγήσει σε αποδοχή. Διαφορετικά, μπορεί να δει κανείς το μη ντετερμινισμό σαν ένα υπολογιστικό δένδρο που μεγαλώνει προς τα κάτω. Τα παιδιά κάθε κόμβου είναι όλες οι πιθανές επόμενες καταστάσεις και δημιουργούνται κλάδοι στο υπολογιστικό δένδρο. Μια συμβολοσειρά είναι αποδεκτή αν ένας από τους κλάδους καταλήγει σε κατάσταση αποδοχής, δηλ. σε τελική κατάσταση.



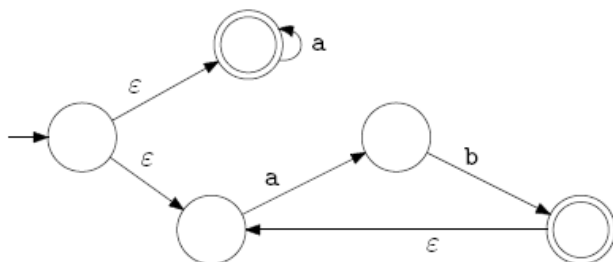
### 2.3.3 $\epsilon$ - μεταβάσεις

Επιπλέον, στα NFA επιτρέπονται  $\epsilon$ -μεταβάσεις: βέλη που επιγράφονται με την κενή συμβολοσειρά. Αυτές οι μεταβάσεις επιτρέπουν στο αυτόματο να αλλάζει κατάσταση χωρίς την κατανάλωση συμβόλων της εισόδου. Για παράδειγμα, αν η συνολική γλώσσα προκύπτει ως ένωση δύο άλλων γλωσσών, μια τέτοια μετάβαση επιτρέπει στο αυτόματο να μαντέψει σε ποια από τις δύο γλώσσες ανήκει η συμβολοσειρά εισόδου.

**Παράδειγμα 2.3.4.** Ένα NFA που αναγνωρίζει όλες τις δυαδικές συμβολοσειρές στις οποίες το τελευταίο σύμβολο είναι 0 ή περιέχουν ένα μόνο 1:



**Παράδειγμα 2.3.5.** Ένα NFA για τη γλώσσα  $a^* + (ab)^*$ :



Ο τυπικός ορισμός για το NFA είναι ο εξής: πρόκειται για μια 5-άδα  $(Q, \Sigma, \delta, q_0, F)$  με:

- $Q$  πεπερασμένο σύνολο καταστάσεων
- $\Sigma$  ένα αλφάβητο εισόδου
- $\delta$  η σχέση μετάβασης

- $q_0$  η αρχική κατάσταση
- $F$  σύνολο τελικών καταστάσεων, υποσύνολο του  $Q$

Η διαφορά είναι ότι τώρα η σχέση μετάβασης καθορίζει ένα σύνολο επόμενων καταστάσεων αντί για μία: απεικονίζει το  $Q \times \Sigma$  στο  $\{\text{υποσύνολα του } Q\}$ . Για παράδειγμα, στο NFA του παραδείγματος 2.3.4, έχουμε  $\delta(A, 1) = \{A\}$ ,  $\delta(B, \epsilon) = \{A, C\}$ , και  $\delta(D, 0) = \emptyset$ .

**Για εξάσκηση...** Δώστε NFA που να ανγνωρίζει το σύνολο των δυαδικών συμβολοσειρών που είτε περιέχουν περιττό αριθμό 0, ή περιέχουν αριθμό 1 που είναι πολλαπλάσιο του 3, είτε και τα δύο.

### 2.3.4 Το θεώρημα του Kleene

Ο μη ντετερμινισμός δεν προσδίδει περισσότερη υπολογιστική ισχύ σε ένα πεπερασμένο αυτόματο:

**Θεώρημα 2.3.1** (Θεώρημα του Kleene). . Τα παρακάτω είναι ισοδύναμα για μια γλώσσα  $L$ :

1. Υπάρχει ένα DFA για την  $L$ .
2. Υπάρχει ένα NFA για την  $L$ .
3. Υπάρχει μία κανονική έκφραση για την  $L$ .

Το θεώρημα αποδεικνύεται σε τρία βήματα. Ουσιαστικά, κατασκευάζουμε έναν αλγόριθμο που εκτελεί τη μετατροπή:

$$(3) \Rightarrow (2) \Rightarrow (1) \Rightarrow (3)$$

Επομένως, αν γνωρίζουμε ότι μια γλώσσα είναι κανονική, γνωρίζουμε επίσης ότι γίνεται δεκτή από κάποιο DFA, κάποιο NFA, και περιγράφεται από μία κανονική έκφραση.

### 2.3.5 Μετατροπή κανονικής έκφρασης σε NFA

Στη συνέχεια μελετάμε το πώς γίνεται η μετατροπή μιας κανονικής έκφρασης σε NFA. Αυτό που κάνουμε είναι αναδρομική κατασκευή.

Αφού μια κανονική έκφραση μπορεί να παραχθεί αναδρομικά, πρέπει (α) να περιγράψουμε ένα NFA για κάθε σύμβολο και για την  $\epsilon$  και (β) να δείξουμε ότι αν υπάρχουν NFA για τις κανονικές εκφράσεις  $A$  και  $B$ , τότε υπάρχει NFA και για τις  $A + B$ ,  $AB$  και  $A^*$ .

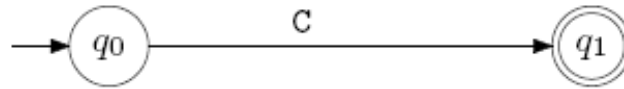
#### Μετατροπή κανονικής έκφρασης σε NFA (περιληπτικά)

1. Αν η κανονική έκφραση είναι η κενή συμβολοσειρά, κατασκεύασε απλά ένα NFA.
2. Αν η κανονική έκφραση είναι ένα απλό σύμβολο, κατασκεύασε απλά ένα NFA.

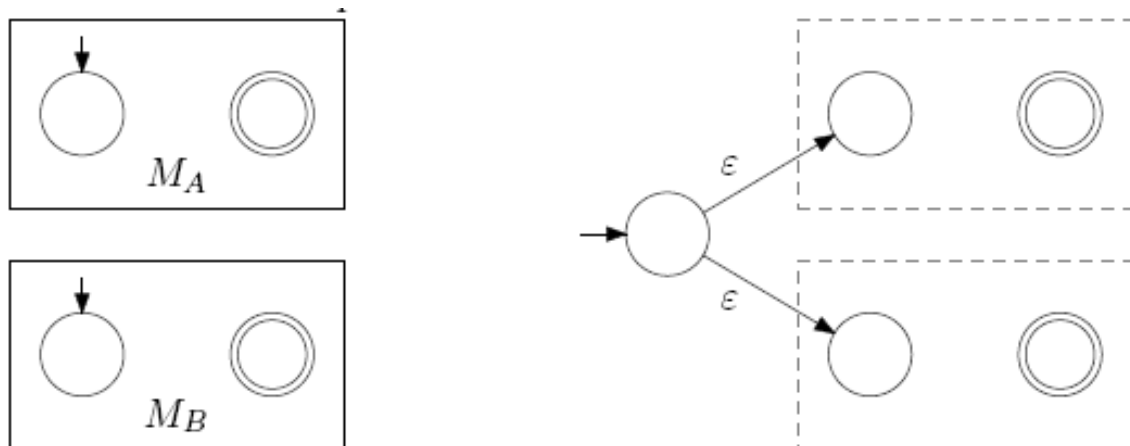
3. Αλλιώς αν η κανονική έκφραση είναι της μορφής  $A + B$ , συνδύασε το NFA για την  $A$  με αυτό για τη  $B$ .
4. Αλλιώς αν η κανονική έκφραση είναι της μορφής  $AB$ , συνδύασε το NFA για την  $A$  με αυτό για τη  $B$ .
5. Αλλιώς αν η κανονική έκφραση είναι της μορφής  $A^*$ , επέκτεινε το NFA για την  $A$ .

Αυτό που μένει είναι να δείξουμε πώς συνδυάζουμε ή επεκτείνουμε τα υπάρχοντα NFA.

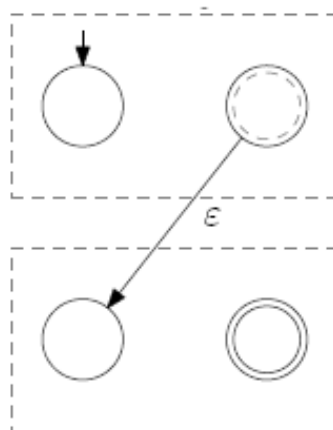
(1) Ένα NFA για ένα απλό σύμβολο  $C$  αποτελείται από δύο καταστάσεις  $q_0$  και  $q_1$ . Η πρώτη είναι η αρχική κατάσταση και η δεύτερη η τελική. Υπάρχει μία μετάβαση από την  $q_0$  στην  $q_1$  που επιγράφεται με το συγκεκριμένο σύμβολο:



(2) Έχοντας έτοιμο ένα NFA  $MA$  για την  $A$  και ένα  $MB$  για τη  $B$ , παρακάτω φαίνεται αυτό για την  $A+B$ . Προσθέτουμε μια καινούρια αρχική κατάσταση με  $\epsilon$ -μεταβάσεις στις αρχικές καταστάσεις των  $MA$  και  $MB$ . Το αυτόματο μαντεύει σε ποια από τις  $A$  ή  $B$  ανήκει η συμβολοσειρά εισόδου.



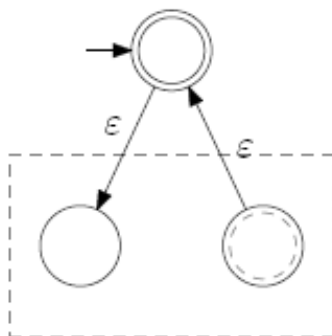
(3) Παρακάτω είναι ένα αυτόματο για την παράθεση  $AB$ . Ξεκινάμε με τα NFA  $MA$  και  $MB$ . Το πρώτο βήμα είναι να προσθέσουμε  $\epsilon$ -μεταβάσεις από τις τελικές καταστάσεις του  $MA$  στην αρχική κατάσταση του  $MB$ . Και φυσικά, οι τελικές καταστάσεις του  $MA$  δεν είναι πλέον τελικές.



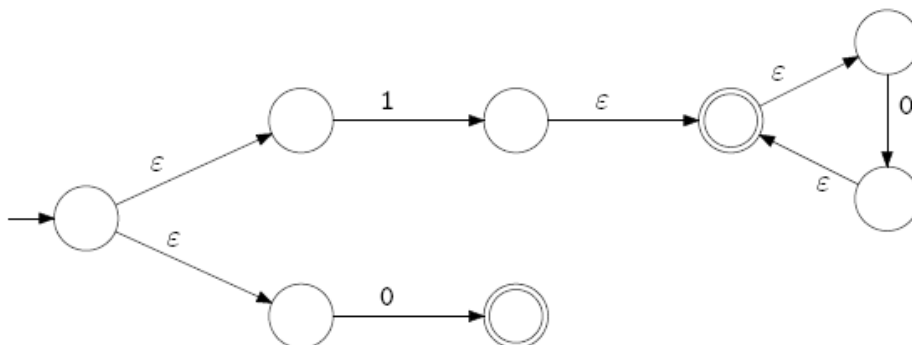
(4) Παρακάτω είναι ένα αυτόματο για την  $A^*$ . Η ιδέα είναι να επιτρέψουμε στη μηχανή να κάνει κύκλους από τις τελικές καταστάσεις πίσω στην αρχική κατάσταση. Μια επόμενη ιδέα θα ήταν να κάνουμε την αρχική κατάσταση και τελική και να προσθέσουμε  $\epsilon$ -μεταβάσεις από τις τελικές καταστάσεις προς την αρχική κατάσταση. Αλλά αυτό δεν δουλεύει πάντα. Γιατί; Μπορεί να προσθέσει συμβολοσειρές στη γλώσσα (που προηγουμένως κατέληγαν στην αρχική κατάσταση που όμως δεν ήταν τελική).

Αυτό που μπορούμε να κάνουμε είναι να προσθέσουμε μια νέα αρχική κατάσταση, που να είναι και τελική και στη συνέχεια να προσθέσουμε μια  $\epsilon$ -μετάβαση από αυτή προς την προηγούμενη αρχική

κατάσταση, και επίσης να προσθέσουμε  $\epsilon$ -μεταβάσεις από τις υπόλοιπες τελικές καταστάσεις προς την προηγούμενη αρχική κατάσταση.



**Παράδειγμα 2.3.6.** Ας εφαρμόσουμε τον παραπάνω αλγόριθμο για να κατασκευάσουμε ένα NFA για τη γλώσσα  $0 + 10^*$ . Ξεκινάμε φτιάχνοντας ένα NFA για το 0, ένα για το 1 και ένα για το  $0^*$ , μετά συνδυάζουμε τα δύο τελευταία και τελικά ενώνουμε τα δύο επιμέρους αυτόματα.



Συχνά, το NFA που προκύπτει μπορεί να απλοποιηθεί όπως και σε αυτή την περίπτωση.

### 2.3.6 Μετατροπή NFA σε DFA

Δείχνουμε τώρα πώς μετατρέπουμε ένα NFA σε DFA. Ο αλγόριθμος βασίζεται στην κατασκευή υποσυνόλων. Γι' αυτό, πρέπει να επανεξετάσουμε τον τρόπο λειτουργίας ενός NFA.

Ας δούμε πώς θα μπορούσαμε να πούμε αν ένα NFA αποδέχεται μια συγκεκριμένη συμβολοσειρά. Η απλούστερη ιδέα είναι να δοκιμάσουμε συστηματικά όλες τις δυνατές μεταβάσεις. Αυτό σίγουρα δουλεύει, αλλά δεν είναι αποδοτικό γιατί απαιτεί εκθετικό χρόνο ως προς το μέγεθος της εισόδου.

Μια καλύτερη προσέγγιση είναι να γνωρίζουμε σε κάθε βήμα το σύνολο των καταστάσεων στις οποίες θα μπορούσε να βρίσκεται το NFA. Το κλειδί είναι ότι μπορούμε να καθορίσουμε το σύνολο σε κάθε βήμα γνωρίζοντας μόνο το αντίστοιχο σύνολο του προηγούμενου βήματος.



**Παράδειγμα 2.3.7.** Ας εξετάσουμε τι κάνει το NFA του παραδείγματος 2.3.1 με είσοδο τη συμβολοσειρά 10100. Αρχικά είναι στην κατάσταση  $A$ . Αφού διαβάσει 1, μεταβαίνει είτε στην κατάσταση  $A$  είτε στη  $C$ . Τι κάνει όταν διαβάσει 0; Αν το αυτόματο είναι στην κατάσταση  $A$ , μπορεί να επιλέξει να μεταβεί είτε στην κατάσταση  $A$  είτε στην  $B$ . Αν το αυτόματο είναι στην κατάσταση  $C$  σταματάει τη λειτουργία του. Οπότε μετά το δεύτερο σύμβολο, το σύνολο των καταστάσεων στις οποίες μπορεί να βρίσκεται το αυτόματο είναι το  $\{A, B\}$ . Μετά διαβάζει το τρίτο σύμβολο, και αν είναι στην κατάσταση  $A$  μεταβαίνει στις  $\{A, C\}$ , ενώ αν είναι στη  $B$  τερματίζει τη λειτουργία του. Αν επομένως εξομοιώνουμε το αυτόματο, είναι αρκετό να ξέρουμε το σύνολο των καταστάσεων στις οποίες θα μπορούσε να βρίσκεται το αυτόματο. Το αυτόματο θα προχωρούσε ως εξής:

$$\{A\} \xrightarrow{1} \{A, C\} \xrightarrow{0} \{A, B\} \xrightarrow{1} \{A, C\} \xrightarrow{0} \{A, B\} \xrightarrow{0} \{A, B, D\}$$

Δέχεται το αυτόματο τη συμβολοσειρά 10100; Ναι: μπορεί να φτάσει σε τελική κατάσταση ( $D$ ) αφού διαβάσει το τελευταίο σύμβολο.

Υπάρχει όμως πεπερασμένο πλήθος συνόλων καταστάσεων. Οπότε, για τη μετατροπή ξεκινάμε από την αρχική κατάσταση, και μετά χρησιμοποιώντας συστηματικό τρόπο ανακλαύπτουμε το υπόλοιπο διάγραμμα. Παρατηρήστε ότι υπάρχουν υποσύνολα που μπορεί ποτέ να μη χρησιμοποιηθούν.

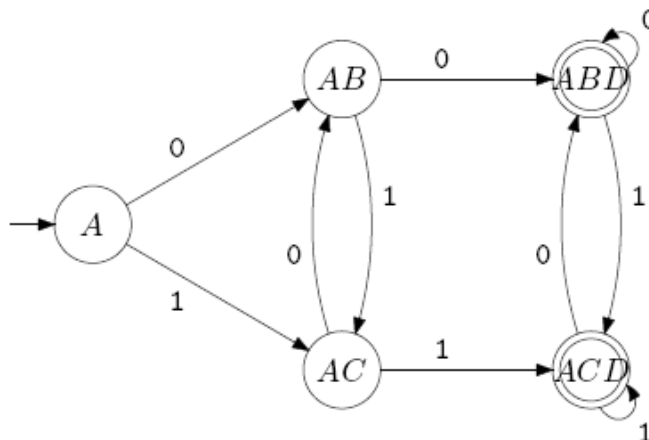
### Μετατροπή ενός NFA (χωρίς $\epsilon$ -μεταβάσεις) σε DFA

1. Κάθε κατάσταση είναι ένα σύνολο καταστάσεων του αρχικού αυτομάτου.
2. Η αρχική κατάσταση  $\{q_0\}$  όπου  $q_0$  είναι η αρχική κατάσταση του αρχικού αυτομάτου.
3. Όσο λείπει μετάβαση από κάποια κατάσταση του DFA κάνε τα εξής: κατασκεύασε τη μετάβαση συνδυάζοντας τις πιθανές επόμενες καταστάσεις για κάθε σύμβολο του συνόλου.
4. Κάνε τελική κατάσταση οποιοδήποτε σύνολο περιέχει τουλάχιστον μία τελική κατάσταση του αρχικού αυτομάτου.

**Παράδειγμα 2.3.8.** Το παρακάτω DFA είναι το αποτέλεσμα εφαρμογής του προηγούμενου αλγορίθμου στο NFA του παραδείγματος 2.3.1. Για παράδειγμα, θεωρήστε την κατάσταση  $\{A, B, D\}$ . Με 1, το NFA αν είναι στην κατάσταση  $A$  μπορεί να μεταβεί στις καταστάσεις  $A$  ή  $C$ , αν είναι στη  $B$  τερματίζει τη λειτουργία του, αν είναι στη  $D$  παραμένει σε αυτή. Επομένως με 1 το DFA μεταβαίνει από την  $\{A, B, D\}$  στην  $\{A, C, D\}$ . Και οι δύο είναι τελικές καταστάσεις αφού περιέχουν την  $D$ .

Η κατασκευή υποσυνόλων δεν δίνει απαραίτητα το μικρότερο δυνατό DFA.

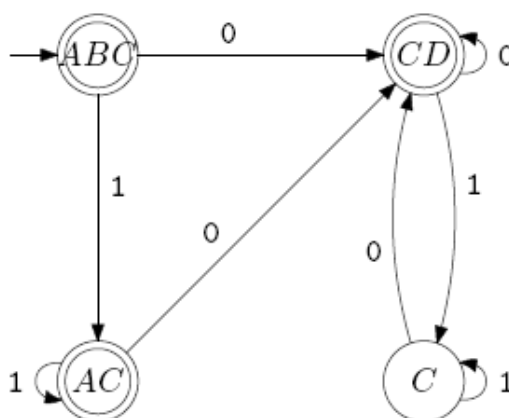
Αν υπάρχουν  $\epsilon$ -μεταβάσεις, πρέπει να τροποποιήσουμε λίγο τη διαδικασία κάνοντας τα παρακάτω.



**Μετατροπή ενός NFA (με  $\epsilon$ -μεταβάσεις) σε DFA** Όπως και προηγουμένως, εκτός του ότι:

1. Η αρχική κατάσταση περιλαμβάνει την προηγούμενη αρχική κατάσταση και όλες στις οποίες μπορούμε να φτάσουμε από αυτή με  $\epsilon$ -μεταβάσεις.
2. Όταν υπολογίζουμε τις καταστάσεις στις οποίες μπορούμε να μεταβούμε από κάποια άλλη, συμπεριλαμβάνουμε κι όλες τις καταστάσεις στις οποίες μπορούμε να μεταβούμε με  $\epsilon$ -μεταβάσεις μετά την κατάσταση προορισμού.

**Παράδειγμα 2.3.9.** Το παρακάτω είναι το αυτόματο που προκύπτει αν εκτελέσουμε τον αλγόριθμο για το NFA του παραδείγματος 2.3.4. Η αρχική κατάσταση περιλαμβάνει την  $A$  και τις δύο καταστάσεις στις οποίες μπορούμε να μεταβούμε με  $\epsilon$ -μεταβάσεις και επομένως είναι η  $\{A, B, C\}$ .



**ε-Κλειστότητα**

Η παρουσία ε-μεταβάσεων (δηλαδή όταν  $q \in \delta(p, \epsilon)$ ) προκαλεί πρακτικά προβλήματα και επομένως εισάγουμε την έννοια της ε-Κλειστότητας (ε-Closure).

**Ορισμός 2.3.2.** Δεδομένου ενός NFA  $N = (K, \Sigma, \delta, q_0, F)$  (με ε-μεταβάσεις) για κάθε κατάσταση  $p \in K$ , η ε-κλειστότητα της  $p$  είναι το σύνολο  $\epsilon\text{-closure}(p)$  που αποτελείται από όλες τις καταστάσεις  $q$  τέτοιες ώστε να υπάρχει μονοπάτι που να καλείται ε από την  $p$  στην  $q$ . Αυτό σημαίνει ότι είτε  $q = p$  είτε ότι όλες οι ακμές του μονοπατιού από την  $p$  στην  $q$  έχουν την ετικέτα ε.

Υπολογίζουμε την  $\epsilon\text{-closure}(p)$  χρησιμοποιώντας μια ακολουθία προσεγγίσεων ως εξής. Ορίζουμε την ακολουθία συνόλων καταστάσεων  $(\epsilon\text{-clo}_i(p))_{i \geq 0}$  όπως φαίνεται πιο κάτω:

$$\begin{aligned}\epsilon\text{-clo}_0(p) &= \{p\}, \\ \epsilon\text{-clo}_{i+1}(p) &= \epsilon\text{-clo}_i(p) \cup \{q \in K \mid \exists s \in \epsilon\text{-clo}_i(p), q \in \delta(s, \epsilon)\}.\end{aligned}$$

Αφού είναι  $\epsilon\text{-clo}_0(p) \subseteq \epsilon\text{-clo}_{i+1}(p)$ , και  $\epsilon\text{-clo}_0(p) \subseteq K$  για όλα τα  $i \geq 0$ , και το σύνολο  $K$  είναι πεπερασμένο, υπάρχει ένα μικρότερο  $i$ , έστω  $i_0$ , τέτοιο ώστε

$$\epsilon\text{-clo}_{i_0}(p) = \epsilon\text{-clo}_{i_0+1}(p),$$

και μπορεί άμεσα να επαληθευθεί ότι

$$\epsilon\text{-closure}(p) = \epsilon\text{-clo}_{i_0}(p).$$

Σημειώνεται ότι υπάρχουν πιο αποδοτικοί τρόποι υπολογισμού του συνόλου  $\epsilon\text{-closure}(p)$ , όπως για παράδειγμα, η χρησιμοποίηση μιας στοίβας (βασικά, ένας τρόπος depth-first αναζήτησης). Όταν το αυτόματο  $N$  δεν έχει ε-μεταβάσεις, δηλαδή όταν  $\delta(p, \epsilon) = \emptyset$  για κάθε  $p \in K$  (γεγονός που σημαίνει ότι η  $\delta$  μπορεί να θεωρηθεί σα συνάρτηση  $\delta : K \times \Sigma \rightarrow 2^Q$ ), τότε είναι

$$\epsilon\text{-closure}(p) = \{p\}.$$

Δεδομένου ενός υποσυνόλου  $S$  του  $K$ , ορίζουμε την  $\epsilon\text{-Closure}(S)$  ως εξής:

$$\epsilon\text{-closure}(S) = \bigcup_{p \in S} \epsilon\text{-closure}(p).$$

Όταν το  $N$  δεν έχει ε-μεταβάσεις, είναι:

$$\epsilon\text{-closure}(S) = S.$$

Στη συνέχεια ορίζουμε την επέκταση  $\delta^* : K \times \Sigma^* \rightarrow 2^Q$  της συνάρτησης μετάβασης  $\delta^* : K \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$ .

Η διαίσθηση σχετικά με τον ορισμό της επέκτασης της συνάρτησης μετάβασης είναι ότι  $\delta^*(p, w)$  είναι το σύνολο όλων των καταστάσεων στις οποίες μπορεί να μεταβεί το αυτόματο από την κατάσταση  $p$  ακολουθώντας ένα μονοπάτι που επιγράφεται  $w$ .

**Ορισμός 2.3.3.** Δεδομένου ενός NFA  $N = (K, \Sigma, \delta, q_0, F)$  (με  $\epsilon$ -μεταβάσεις), η επέκταση της συνάρτησης  $\delta^* : K \times \Sigma^* \rightarrow 2^Q$  ορίζεται ως εξής: για κάθε  $p \in Q$ , κάθε  $u \in \Sigma^*$ , και κάθε  $\alpha \in \Sigma$ ,

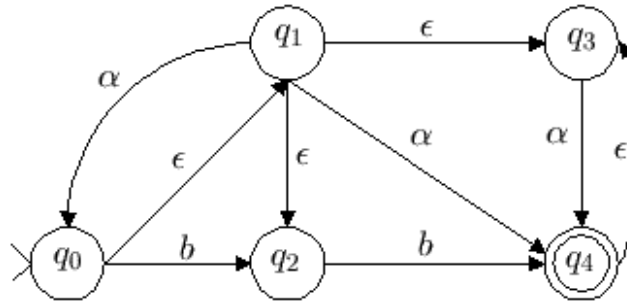
$$\delta^*(p, \epsilon) = \epsilon\text{-closure}(p),$$

$$\delta^*(p, u\alpha) = \epsilon\text{-closure}\left(\bigcup_{s \in \delta^*(p, u)} \delta(s, \alpha)\right).$$

Η γλώσσα  $L(N)$  που γίνεται αποδεκτή από ένα NFA  $N$  είναι το σύνολο

$$L(N) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}.$$

**Παράδειγμα 2.3.10.** Δώστε το ισοδύναμο ντετερμινιστικό αυτόματο  $M'$  για το μη ντετερμινιστικό αυτόματο  $M$  του σχήματος.



Εφόσον το  $M$  έχει πέντε καταστάσεις το  $M'$  θα έχει  $2^5 = 32$  καταστάσεις. Λήγες όμως από αυτές θα σχετίζονται με τη λειτουργία του  $M'$ , συγκεκριμένα θα είναι αυτές στις οποίες θα μπορεί να οδηγηθεί το αυτόματο από την κατάσταση  $s'$  διαβάζοντας κάποια συμβολοσειρά εισόδου. Θα κατασκευάσουμε το  $M'$  ξεκινώντας από την  $s'$  και θα εισάγουμε νέα κατάσταση μόνο όταν χρειάζεται ως τιμή της  $\delta'(q, \sigma)$  για κάποια κατάσταση  $q$  που έχει ήδη εισαχθεί και κάποιο σύμβολο  $\sigma \in \Sigma$ . Αφού  $s' = E(q_0) = \{q_0, q_1, q_2, q_3\}$ ,

$$(q_1, \alpha, q_0)$$

$$(q_1, \alpha, q_4)$$

είναι όλες οι μεταβάσεις  $(q, \alpha, p)$  για κάποια κατάσταση  $q \in s'$ . Προκύπτει ότι

$$\delta'(s', \alpha) = E(q_0) \cup E(q_4) = \{q_0, q_1, q_2, q_3, q_4\}.$$

Ομοίως,

$$(q_0, b, q_2)$$

$$(q_2, b, q_4)$$

είναι όλες οι μεταβάσεις  $(q, b, p)$  για κάποια κατάσταση  $q \in s'$  οπότε

$$\delta'(s', b) = E(q_2) \cup E(q_4) = \{q_2, q_3, q_4\}.$$

Επαναλαμβάνοντας τον υπολογισμό αυτόν για τις νεοεισαχθείσες καταστάσεις, έχουμε τα παρακάτω:

$$\delta'(\{q_0, q_1, q_2, q_3, q_4\}, \alpha) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_2, q_3, q_4\}$$

$$\delta'(\{q_2, q_3, q_4\}, \alpha) = E(q_4) = \{q_3, q_4\}$$

$$\delta'(\{q_2, q_3, q_4\}, b) = E(q_4) = \{q_3, q_4\}$$

Τελικά,

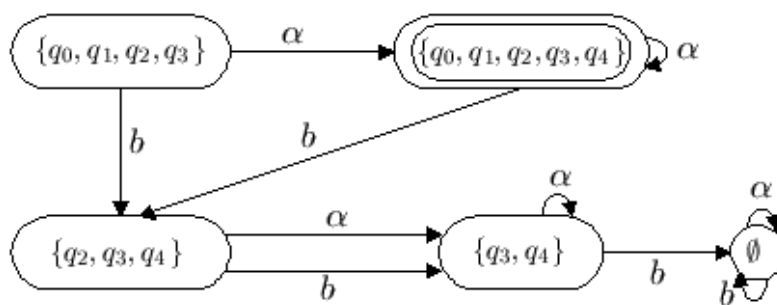
$$\delta'(\{q_3, q_4\}, \alpha) = E(q_4) = \{q_3, q_4\}$$

$$\delta'(\{q_3, q_4\}, b) = \emptyset$$

και

$$\delta'(\emptyset, \alpha) = \delta'(\emptyset, b) = \emptyset.$$

Το  $F'$  είναι το σύνολο των τελικών καταστάσεων περιέχει κάθε τελική κατάσταση του συνόλου τελικών καταστάσεων του  $M$ , δηλαδή των  $q_4$ . Έτσι οι τελικές καταστάσεις του  $M'$  είναι αυτές που περιέχουν την  $q_4$  δηλαδή οι  $\{q_0, q_1, q_2, q_3, q_4\}$ ,  $q_2, q_3, q_4$ , και  $q_3, q_4$ . Το αυτόματο  $M$  φαίνεται στο ακόλουθο σχήμα.



### Ισοδυναμία NFA και DFAs

Κάθε μη ντετερμινιστικό πεπερασμένο αυτόματο είναι ισοδύναμο με ένα ντετερμινιστικό πεπερασμένο αυτόματο. Εν τούτοις τα μη ντετερμινιστικά πεπερασμένα αυτόματα έχουν μερικά ιδιαίτερα χαρακτηριστικά που τους προσδίδουν μεγαλύτερη ευελιξία από τα ντετερμινιστικά: Μη ντετερμινισμός είναι ουσιαστικά η ικανότητα αλλαγής θέσης με τρόπο που αποφασίζεται μόνο εν μέρει από

την παρούσα κατάσταση και το σύμβολο εισόδου. Έτσι, υπάρχουν ενδεχομένως πολλές επόμενες καταστάσεις για δεδομένο συνδυασμό παρούσας κατάστασης και συμβόλου εισόδου. Για κάποιους συνδυασμούς καταστάσεων και συμβόλων εισόδου μπορεί να μην υπάρχει καμία επόμενη κατάσταση. Τα διαγράμματα καταστάσεων μη ντετερμινιστικών πεπερασμένων αυτομάτων επιτρέπεται να έχουν τόξα με ονόματα είτε σύμβολα του αλφαβήτου είτε ακόμα την κενή συμβολοσειρά  $\epsilon$ .

*Τα DFA και τα NFA αναγνωρίζουν ακριβώς το ίδιο σύνολο γλωσσών.*

Δεδομένου ότι κάθε DFA μπορεί να θεωρηθεί σαν ειδική περίπτωση ενός NFA (δηλ. όλα τα DFA είναι ουσιαστικά NFA), κάθε γλώσσα που αναγνωρίζεται από ένα DFA αναγνωρίζεται επίσης και από ένα αντίστοιχο NFA, δηλαδή το ίδιο.

Για να δείξουμε ότι μια γλώσσα που αναγνωρίζεται από ένα NFA αναγνωρίζεται και από ένα DFA, θεωρούμε ένα NFA  $M$  με  $M = (Q, \Sigma, \delta, S, F)$  τέτοιο ώστε το  $S$  να είναι ένα σύνολο αρχικών καταστάσεων. Θεωρούμε επίσης ένα DFA  $M'$  με  $M' = (\wp(Q), \Sigma, \delta', S, \{\text{σύνολα στοιχείων του } Q \text{ τέτοια ώστε } Q \cap F \neq \emptyset\})$ . Έστω τώρα ότι το  $M'$  είναι σε μια κατάσταση  $A$ , η οποία είναι ουσιαστικά ένα σύνολο καταστάσεων του  $M$ . Όταν το  $M'$  διαβάζει ένα σύμβολο  $\sigma$ , μεταβαίνει σε μια νέα κατάσταση η οποία αντιστοιχεί και πάλι σε ένα άλλο σύνολο καταστάσεων. Επομένως η συνάρτηση μετάβασης  $\delta'$  του  $M'$  είναι:  $\delta'(A, \sigma) = \bigcup_{q \in A} \delta(q, \sigma)$ . Δηλαδή συμπεριλαμβάνει όλα τα στοιχεία για μια μετάβαση από κάθε κατάσταση του  $A$  για κάθε σύμβολο  $\sigma$ .

Αν μια συμβολοσειρά αναγνωρίζεται από το αυτόματο  $M$ , θα πρέπει επίσης να αναγνωρίζεται και από το αυτόματο  $M'$ . Αν ο υπολογισμός για μια συμβολοσειρά  $w$  σε ένα αυτόματο  $M$  διέρχεται από τις καταστάσεις  $q_0, \dots, q_n$ , τότε ο αντίστοιχος υπολογισμός σε ένα αυτόματο  $M'$  θα πρέπει να διέλθει από την ίδια ακολουθία καταστάσεων  $s_0, \dots, s_n$ , όπου  $q_i \in s_i$  για κάθε  $i$ . Αφού η  $q_n$  είναι τελική κατάσταση, η  $s_n$  θα πρέπει να ανήκει στο σύνολο των τελικών καταστάσεων του αυτομάτου  $M'$ .

Αντίστροφα, αν μια συμβολοσειρά αναγνωρίζεται από το αυτόματο  $M'$ , αυτό σημαίνει ότι το αυτόματο διέρχεται από κάποιες καταστάσεις  $s_0, \dots, s_n$  όπου  $s_n \cap F \neq \emptyset$ . Τότε θα πρέπει να υπάρχει μια ακολουθία καταστάσεων  $q_0, \dots, q_n$  τέτοια ώστε  $q_n \in F$  και να προκύπτει ένας σωστός υπολογισμός του αυτομάτου  $M$ . Αυτό μπορούμε να το διαπιστώσουμε επαγωγικά:

Αν  $n = 0$  τότε η  $s_0$  περιλαμβάνει μόνο αρχικές καταστάσεις. Στην περίπτωση αυτή, για κάθε  $q \in s_0$ , έχουμε έναν σωστό υπολογισμό του αυτομάτου  $M$  για τη συμβολοσειρά  $w$  αν ο υπολογισμός τερματίζει στην  $q$ . Αν υποθέσουμε ότι για κάθε ακολουθία σύνθετων καταστάσεων  $s_0, \dots, s_k$  για  $k \leq n$ , και κάθε κατάσταση  $q_k$  που ανήκει στην  $s_k$  υπάρχει ένας σωστός υπολογισμός  $q_1, \dots, q_k$  που τερματίζει στην κατάσταση  $q_k$ , τότε για κάθε  $q_{k+1} \in s_{k+1}$ , η κατάσταση  $q_{k+1}$  ανήκει στην ένωση  $\bigcup q \in s_k \delta(q, \sigma)$ . Επομένως υπάρχει κάποια κατάσταση  $q$  τέτοια ώστε  $q_{k+1} \in \delta(q, \sigma)$  με  $q \in s_k$ . Επίσης, θα πρέπει να υπάρχει κάποιος σωστός υπολογισμός για τα πρώτα  $k$  σύμβολα της συμβολοσειράς  $w$ , με  $q_0, \dots, q_k = q$  από την επαγωγική υπόθεση, και επομένως, υπάρχει σωστός υπολογισμός  $q_0, \dots, q_{k+1}$  για την  $w$  που απλά παραθέτει την  $q_{k+1}$  με την ακολουθία καταστάσεων του υπολογισμού που έχει ήδη σχηματιστεί.

Επαγωγικά, με είσοδο τη συμβολοσειρά  $w$  για κάθε σωστό υπολογισμό του αυτομάτου  $M'$ ,  $s_0, \dots, s_n$ , και για κάθε κατάσταση  $q \in s_n$ , υπάρχει ένας σωστός υπολογισμός του αυτομάτου  $M$  με είσοδο την ίδια συμβολοσειρά  $w$  που τερματίζει στην κατάσταση  $q$ . Άρα, αν είναι  $s_n \cap F \neq \emptyset$  υπάρχει κατάσταση  $q \in F \cap s_n$ , και κατά συνέπεια υπάρχει σωστός υπολογισμός του αυτομάτου  $M$  που τερματίζει στην κατάσταση  $q$ , και επομένως  $w \in L(M)$ .

Δύο πεπερασμένα αυτόματα  $M_1$  και  $M_2$  είναι ισοδύναμα αν και μόνον αν  $L(M_1) = L(M_2)$ . Απλούστερα, δύο αυτόματα θεωρούνται ισοδύναμα αν δέχονται την ίδια γλώσσα ανεξάρτητα από τον τρόπο που υιοθετεί καθένα για να το πετύχει. Ένα DFA αποτελεί ειδική περίπτωση ενός NFA και δεν περιέχει μεταβάσεις χωρίς κατανάλωση εισόδου ούτε πολλαπλές μεταβάσεις για το ίδιο σύμβολο εισόδου.

**Θεώρημα 2.3.2.** *Για κάθε μη ντετερμινιστικό πεπερασμένο αυτόματο υπάρχει ένα ισοδύναμο ντετερμινιστικό πεπερασμένο αυτόματο.*

*Απόδειξη.* Έστω  $M = (K, \Sigma, \Delta, s, F)$  ένα μη ντετερμινιστικό πεπερασμένο αυτόματο.

Για να μετατραπεί σε ισοδύναμο ντετερμινιστικό πεπερασμένο αυτόματο πρέπει να αφαιρεθούν:

- μεταβάσεις του τύπου  $(q, u, q') \in \Delta$  έτσι ώστε  $|u| > 1$  ή  $u = e$ ,
- μεταβάσεις που λείπουν και
- πολλαπλές μεταβάσεις που μπορούν να εφαρμοστούν σε μια συνολική κατάσταση.

Προκειμένου να απαλειφθούν μεταβάσεις του πρώτου τύπου, εισάγουμε νέες καταστάσεις για να αντικαταστήσουμε ένα τόξο στο διάγραμμα καταστάσεων, όπως φαίνεται στο σχήμα που ακολουθεί:

Τυπικά, αν  $(q, \sigma_1, \dots, \sigma_k, q')$  είναι μια μετάβαση του  $M$  και  $\sigma_1, \dots, \sigma_k \in \Sigma$ ,  $k \geq 2$ , τότε προσθέτουμε νέες μη τελικές καταστάσεις  $p_1, \dots, p_k$  στο  $K$  και νέες μεταβάσεις  $(q, \sigma_1, p_1)$ ,  $(p_1, \sigma_2, p_2)$ ,  $\dots$ ,  $(p_{k-1}, \sigma_k, q')$  στο  $\Delta$ . Έστω  $M' = (K', \Sigma, \Delta', s', F')$  το μη ντετερμινιστικό πεπερασμένο αυτόματο που προκύπτει από το  $M$  όταν εφαρμοστεί ο μετασχηματισμός αυτός σε κάθε μετάβαση  $(q, u, q') \in \Delta$  έτσι ώστε  $|u| > 1$ . Είναι φανερό ότι τα  $M$  και  $M'$  είναι ισοδύναμα και ότι  $|u| \leq 1$  για κάθε  $(q, u, q')$  του  $M$ .

Θα κατασκευάσουμε τώρα ένα ντετερμινιστικό πεπερασμένο αυτόματο  $M'' = (K'', \Sigma, \delta'', s'', F'')$  ισοδύναμο του  $M'$ . Η βασική ιδέα είναι να θεωρήσουμε ότι ένα μη ντετερμινιστικό πεπερασμένο αυτόματο βρίσκεται όχι σε μία κατάσταση αλλά σε ένα σύνολο καταστάσεων, δηλαδή σε όλες τις δυνατές καταστάσεις που μπορεί να οδηγηθεί από την αρχική κατάσταση καταναλώνοντας την είσοδο που έχει διαβάσει ως τώρα. Με βάση την ιδέα αυτή κατασκευάζουμε το ζητούμενο αυτόματο:

- Το σύνολο καταστάσεων του  $M''$  θα είναι το  $2K'$ , το δυναμοσύνολο δηλαδή των καταστάσεων του  $M'$ .
- Το σύνολο των τελικών καταστάσεων του  $M''$  θα αποτελείται από όλα εκείνα τα υποσύνολα του  $K'$  που περιέχουν μια τουλάχιστο τελική κατάσταση του  $M'$ .

- Προκειμένου να ορίσουμε τη συνάρτηση μετάβασης του  $M''$  ακολουθούμε την εξής ιδέα: μια κίνηση του  $M''$ , όταν αυτό διαβάζει ένα σύμβολο εισόδου  $\sigma \in \Sigma$ , μιμείται την κίνηση του  $M'$  με σύμβολο εισόδου  $\sigma$  ακολουθούμενη από κάποιο αριθμό κινήσεων του  $M'$ , κατά τις οποίες δεν καταναλώνεται είσοδος.

**Πιο τυπικά:** Για κάθε κατάσταση  $q \in K'$ , έστω  $E(q)$  το σύνολο όλων των καταστάσεων του  $M'$ , στις οποίες είναι δυνατό να μεταβεί το  $M'$  από την  $q$  χωρίς να διαβάσει κανένα σύμβολο εισόδου, δηλαδή:  $E(q) = \{p \in K' : (q, \epsilon) \stackrel{*}{\vdash}_{M'} (p, \epsilon)\}$ . Αν το  $M'$  κινείται χωρίς να καταναλώνει καθόλου είσοδο, η λειτουργία του είναι ανεξάρτητη από την είσοδο. Οπότε το  $E(q)$  θα μπορούσε εναλλακτικά να οριστεί μέσω μιας τυχαίας συμβολοσειράς  $w \in \Sigma^*$  ως εξής:  $E(q) = \{p \in K' : (q, w) \stackrel{*}{\vdash}_{M'} (p, w)\}$ .

Απομένει να δείξουμε ότι το  $M''$  είναι ντετερμινιστικό και ισοδύναμο του  $M'$ . Το ότι το  $M''$  είναι ντετερμινιστικό είναι προφανές, μιας και η  $\delta''$  είναι μονοσήμαντα και καλά ορισμένη από κατασκευής. Ισχυριζόμαστε ότι για κάθε συμβολοσειρά  $w \in \Sigma^*$  και οποιεσδήποτε καταστάσεις  $q, p \in K'$  είναι:  $(q, w) \stackrel{*}{\vdash}_{M'} (p, \epsilon)$  αν και μόνον αν  $(E(q), w) \stackrel{*}{\vdash}_{M''} (P, \epsilon)$  για κάποιο  $R$  που περιέχει την κατάσταση  $p$ .

Για να δείξουμε ότι τα  $M'$  και  $M''$  είναι ισοδύναμα θεωρούμε μια συμβολοσειρά  $w \in \Sigma^*$ . Τότε  $w \in L(M')$  αν και μόνον αν  $(s', w) \stackrel{*}{\vdash}_{M'} (f', \epsilon)$  για κάποια  $f' \in F'$  και αν μόνον αν  $(E(s'), w) \stackrel{*}{\vdash}_{M'} (Q, \epsilon)$  για κάποιο  $Q$  που περιέχει την  $f'$ , δηλαδή με αν και μόνον αν  $(s'', w) \stackrel{*}{\vdash}_{M'} (Q, \epsilon)$ . Η τελευταία συνθήκη είναι ο ορισμός της πρότασης  $w \in L(M'')$ .

Θα αποδείξουμε τον ισχυρισμό με επαγωγή στο  $|w|$ .

- **Βασικό Βήμα:** για  $|w| = 0$  – δηλαδή για  $w = \epsilon$  – πρέπει να δείξουμε ότι  $(q, \epsilon) \stackrel{*}{\vdash}_{M'} (p, \epsilon)$  αν και μόνο αν  $(E(q), \epsilon) \stackrel{*}{\vdash}_{M''} (P, \epsilon)$  για κάποιο σύνολο  $R$  που περιέχει την κατάσταση  $p$ . Η πρώτη πρόταση είναι ισοδύναμη με την  $p \in E(q)$ . Εφόσον το  $M''$  είναι ντετερμινιστικό, δηλαδή διαβάζει ένα σύμβολο σε κάθε κίνηση, η δεύτερη πρόταση είναι ισοδύναμη με την  $P = E(q)$  και το  $R$  περιέχει την  $p$ , δηλαδή  $p \in E(q)$ .
- **Επαγωγική υπόθεση:** Υποθέτουμε ότι ο ισχυρισμός ισχύει για όλες τις συμβολοσειρές  $w$  μήκους  $k$  ή λιγότερο για κάποιο  $k \geq 0$ .
- **Επαγωγικό Βήμα:** Θα αποδείξουμε τον ισχυρισμό για κάθε συμβολοσειρά  $w$  μήκους  $k + 1$ . Έστω  $w = va$ , όπου  $a \in \Sigma$  και  $v \in \Sigma^*$ . Υποθέτουμε ότι  $(q, w) \stackrel{*}{\vdash}_{M'} (p, \epsilon)$ . Τότε υπάρχουν καταστάσεις  $r_1$  και  $r_2$  τέτοιες ώστε:

$$(q, va) \stackrel{*}{\vdash}_{M'} (r_1, a) \stackrel{*}{\vdash}_{M'} (r_2, \epsilon) \stackrel{*}{\vdash}_{M'} (p, \epsilon),$$

δηλαδή το  $M'$  οδηγείται σε κατάσταση  $p$  από κατάσταση  $q$  με τις ακόλουθες κινήσεις: έναν αριθμό κινήσεων κατά τις οποίες διαβάζεται η συμβολοσειρά εισόδου  $v$ , μια κίνηση κατά την



οποία διαβάζεται η είσοδος  $a$  και έναν αριθμό κινήσεων κατά τις οποίες δεν διαβάζεται είσοδος. Εφόσον  $(q, va) \vdash_{M'}^* (r_1, a)$  τότε  $(q, v) \vdash_{M'}^* (r_1, \epsilon)$  και εφόσον  $|v| = k$ , από την επαγωγική υπόθεση  $(E(q), v) \vdash_{M''}^* (R_1, \epsilon)$  για κάποιο σύνολο  $R_1$  που περιέχει τη  $r_1$  και  $(r_1, a) \vdash_{M'}^* (r_2, \epsilon)$  και  $(r_1, a, r_2) \in \Delta'$ .

Τότε από την κατασκευή του  $M''$ ,  $E(r_2) \in \delta'(R_1, a)$ . Αφού όμως  $(r_2, \epsilon) \vdash_{M'}^* (p, \epsilon), p \in E(r_2)$  και γι' αυτό  $p \in \delta''(R_1, a)$ . Έτσι,  $(R_1, a) \vdash_{M''}^* (P, \epsilon)$  για κάποιο  $R$  που περιέχει την κατάσταση  $p$  και  $(E(q), va) \vdash_{M''}^* (R_1, a) \vdash_{M''}^* (P, \epsilon)$ .

Υποθέτουμε ότι  $(E(q), va) \vdash_{M''}^* (R_1, a) \vdash_{M''}^* (P, \epsilon)$  για κάποιο  $R$  που περιέχει την  $p$  και για κάποιο  $R_1$  τέτοιο ώστε  $\delta''(R_1, a) = R$ . Από τον ορισμό της  $\delta'$ ,  $\delta'(R_1, a)$  είναι η ένωση όλων των συνόλων  $E(r_2)$  όπου για κάποια κατάσταση  $r_1 \in R_1$ ,  $(r_1, a, r_2)$  είναι μια μετάβαση του  $M'$ . Αφού  $p \in P = \delta''(R_1, a)$ , υπάρχει κάποιο συγκεκριμένο  $r_2$  τέτοιο ώστε  $p \in E(r_2)$  και για κάποιο  $r_1 \in R_1$ ,  $(r_1, a, r_2)$  είναι μια μετάβαση του  $M'$ . Τότε

$$(r_2, \epsilon) \vdash_{M'}^* (p, \epsilon)$$

κατά τον ορισμό του  $E(r_2)$ .

Επίσης από την επαγωγική υπόθεση  $(q, v) \vdash_{M'}^* (r_1, \epsilon)$  και γι' αυτό

$$(q, va) \vdash_{M'}^* (r_1, a) \vdash_{M'}^* (r_2, \epsilon) \vdash_{M'}^* (p, \epsilon).$$

□

### 2.3.7 Μετατροπή FA σε κανονική έκφραση

Δείχνουμε τώρα πώς να μετατρέζουμε ένα FA σε κανονική έκφραση.

Ένας τρόπος να γίνει αυτό είναι να γενικεύσουμε ακόμα περισσότερο την ιδέα του NFA. Σε μια γενίκευσή του που λέγεται γενικευμένο μη ντετερμινιστικό πεπερασμένο αυτόματο (GNFA), κάθε μετάβαση γίνεται με κάποια κανονική έκφραση. Δίνουμε ιδιαίτερη προσοχή στην παρακάτω περίπτωση:

υπάρχει μία μόνο τελική κατάσταση, δεν υπάρχει εξερχόμενη μετάβαση από την τελική κατάσταση, και δεν υπάρχει εισερχόμενη μετάβαση προς την αρχική κατάσταση (ούτε καν ανακύκλωση). Χρησιμοποιώντας  $\epsilon$ -μεταβάσεις, μπορούμε εύκολα να μετατρέψουμε οποιοδήποτε NFA στην παραπάνω μορφή.

Προχωράμε φτιάχνοντας μια σειρά από GNFA εξαλείφοντας σε κάθε βήμα μία κατάσταση (όχι την αρχική ή την τελική) του δοσμένου αυτομάτου η οποία αντικαθίσταται από μεταβάσεις που έχουν το ίδιο αποτέλεσμα. Συγκεκριμένα, αν υπάρχει μετάβαση  $a$  από μια κατάσταση 1 σε μία κατάσταση 2,

μετάβαση  $b$  από τη 2 στη 2 και μετάβαση  $c$  από τη 2 στην 3, μπορούμε να έχουμε το ίδιο αποτέλεσμα με μια μετάβαση  $ab^*c$  από την κατάσταση 1 στην 3.



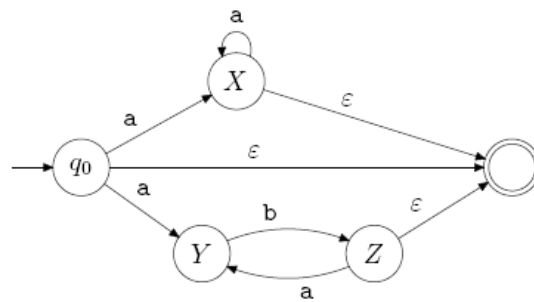
Η γλώσσα που αναγνωρίζεται από το αυτόματο δεν αλλάζει, ενώ δυο μεταβάσεις που συνδέουν το ίδιο ζεύγος καταστάσεων μπορούν να συγχωνευθούν με χρήση της πράξης ένωση για κανονικές εκφράσεις.

Επαναλαμβάνουμε τη διαδικασία αυτή μέχρι να μείνουν μόνο δύο καταστάσεις: η αρχική και η τελική. Η επιγραφή στη μοναδική μετάβαση που συνδέει αυτές τις δύο καταστάσεις είναι η κανονική έκφραση που αντιστοιχεί στη γλώσσα που αβανγνρίζει το αρχικό NFA. Η διαδικασία συνοψίζεται ως εξής:

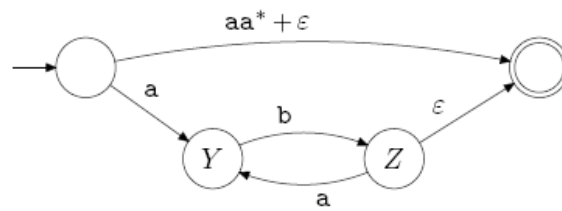
### Μετατροπή NFA σε κανονική έκφραση

1. Μετατρέπουμε το NFA στη σωστή μορφή (κανένα εισερχόμενο βέλος στην αρχική κατάσταση, κανένα εξερχόμενο βέλος από τη μοναδική τελική κατάσταση).
2. Όσο υπάρχουν παραπάνω από δύο καταστάσεις απομακρύνουμε μία κατάσταση αντικαθιστώντας τη με κατάλληλες μεταβάσεις.
3. Βρίσκουμε την κανονική έκφραση σαν επιγραφή στη μετάβαση που μένει τελικά.

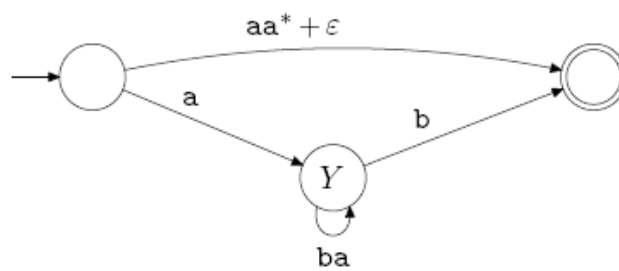
**Παράδειγμα 2.3.11.** Παρακάτω είναι το NFA του παραδείγματος 2.3.3 τροποποιημένο ώστε να έχει μία μόνο τελική κατάσταση.



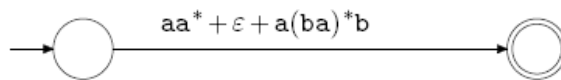
Απομακρύνοντας την κατάσταση  $X$  έχουμε



Απομακρύνοντας την κατάσταση  $Z$  έχουμε



Απομακρύνοντας την κατάσταση  $Y$  έχουμε



### Ο αλγόριθμος εξάλειψης κόμβων (node elimination algorithm)

Η ιδέα είναι να χρησιμοποιήσουμε πιο γενικές επιγραφές στις ακμές μεταξύ των κόμβων ενός NFA, δηλαδή να επιγράφουμε τις μεταβάσεις με κανονικές εκφράσεις. Αυτή η διαδικασία γίνεται αυστηρή αν μελετήσουμε τη γενική μορφή ενός NFA με βάση γράφους με επιγραφές (labeled graphs). Ένα γενικευμένο μη ντετερμινιστικό αυτόματο GNFA (generalized NFA), ορισμένο σε ένα αλφάβητο  $\Sigma$  πρόκειται για έναν κατευθυνόμενο γράφο με επιγραφές από το σύνολο επιγραφών  $\mathcal{R}(\Sigma)$ , με αρχική κατάσταση  $v_0$  και ένα σύνολο τελικών καταστάσεων,  $F$ . Δηλαδή, ένα GNFA είναι μια ακολουθία της μορφής:  $(V, E, R(\Sigma), s, t, \lambda, e_0, F)$ . Ένα GNFA αποδέχεται μια συμβολοσειρά  $w$  αν υπάρχει μονοπάτι  $\pi = (u, e_0 \dots e_n, v)$ , όπου  $e_n$  είναι τελική κατάσταση και  $w \in L(\lambda(e_1) \dots \lambda(e_n))$ . Για κάθε NFA ορισμένο με βάση μια γλώσσα, υπάρχει ένα πραγματικό GNFA ορισμένο στην ίδια γλώσσα, που αναγνωρίζει το ίδιο σύνολο συμβολοσειρών με το NFA.

**Παράδειγμα 2.3.12.** Το απλούστερο παράδειγμα είναι ένα GNFA με κορυφές  $v_0$  και  $v_1$ , με τελική την  $v_1$  και μία ακμή που επιγράφεται με μια κανονική έκφραση  $R$ . Η γλώσσα που γίνεται δεκτή από αυτό το GNFA είναι ακριβώς η  $L(R)$ .

$$v_0 \xrightarrow{R} v_1$$

Η απόδειξη προχωράει ως εξής: Ξεκινάμε με ένα NFA το οποίο θεωρούμε GNFA. Στη συνέχεια ελαττώνουμε το GNFA, διατηρώντας τη γλώσσα που αναγνωρίζει σε κάθε βήμα της μετατροπής, έως ότου καταλήξουμε σε ένα GNFA με μία μόνο ακμή. Η έκφραση που θα υπάρχει τότε σαν επιγραφή της ακμής θα είναι η γλώσσα που δέχεται το αρχικό NFA. Η παραπάνω διαδικασία εξελίσσεται στις ακόλουθες φάσεις:

**Φάση Προεπεξεργασίας 1:** Θεωρούμε το δοσμένο NFA. Αν υπάρχουν εισερχόμενες ακμές στην αρχική του κατάσταση, πρόσθεσε μια καινούρια αρχική κατάσταση με μια  $\epsilon$ -μετάβαση προς την προηγούμενη. Αν είναι απαραίτητο, προσθέτουμε και μια καινούρια (μοναδική) τελική κατάσταση με  $\epsilon$ -μεταβάσεις από κάθε προηγούμενη τελική κατάσταση προς τη νέα (στην περίπτωση που υπάρχουν εξερχόμενες ακμές από τις προηγούμενες τελικές καταστάσεις). Στο τέλος της παρούσας φάσης δεν υπάρχουν εισερχόμενες ακμές στην αρχική κατάσταση  $s$  κι επίσης δεν υπάρχουν εξερχόμενες ακμές από την τελική κατάσταση  $t$ .

**Φάση Προεπεξεργασίας 2:** Για κάθε ζεύγος καταστάσεων  $(p, q)$  (μπορεί να είναι  $p = q$ ), αν υπάρχουν  $k$  ακμές από την  $p$  στην  $q$  που έχουν επιγραφές  $u_1, \dots, u_k$  τότε δημιουργήσε μια

απλή ακμή που να επιγράφεται με την κανονική έκφραση

$$u_1 + \dots + u_k.$$

Για κάθε ζεύγος καταστάσεων  $(p, q)$  (μπορεί να είναι  $p = q$ ), με ταξύ των οποίων δεν υπάρχει ακμή, δημιουργήσε μια ακμή με επιγραφή  $\emptyset$ . Στο τέλος της φάσης αυτής, το GNFA που προέκυψε έχει την εξής ιδιότητα:

**Ιδιότητα 2.3.1.** Για κάθε ζεύγος καταστάσεων  $(p, q)$  (μπορεί να είναι  $p = q$ ), υπάρχει μοναδική ακμή μεταξύ τους που έχει σαν επιγραφή κάποια κανονική έκφραση που συμβολίζεται  $R_{p,q}$ . Όταν  $R_{p,q} = 0$ , τότε δεν υπάρχει ακμή μεταξύ των  $p$  και  $q$  στο αρχικό NFA. Αν υλοποιήσουμε κάθε  $R_{p,q}$  σε μια κλήση συνάρτησης προς ένα NFA  $N_{p,q}$  (που κατασκευάστηκε με τον προηγούμενο αλγόριθμο), που να δέχεται τη γλώσσα  $L[R_{p,q}]$ , μπορούμε να επαληθεύσουμε ότι η αρχική γλώσσα  $L(N)$  γίνεται δεκτή από το το γενικευμένο αυτόματο GNFA.

**Εξάλειψη Κόμβων:** Ο αλγόριθμος εφαρμόζεται μόνο αν το γενικευμένο NFA έχει τουλάχιστο έναν ακόμα κόμβο διαφορετικό από τους  $s$  και  $t$ . Η λειτουργία του είναι η εξής: Διάλεξε έναν κόμβο  $r$ , διαφορετικό από τους  $s$  και  $t$ . Για κάθε ζεύγος  $(p, q)$  με  $p \neq r$  και  $q \neq r$ , αντικατάστησε την επιγραφή της ακμής από τον  $p$  στον  $q$  όπως φαίνεται παρακάτω.

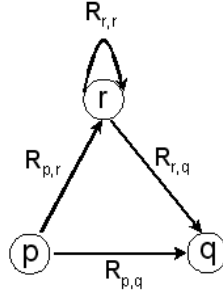
Πριν την εξάλειψη του κόμβου  $r$ .

Στο τέλος αυτού του βήματος, διάγραψε τον κόμβο  $r$  και όλες τις προσκείμενες σε αυτόν ακμές. Παρατηρούμε ότι στην περίπτωση που  $p = q$ , το τρίγωνο είναι επίπεδο. Επίσης, ενδέχεται να είναι  $p = s$  ή  $q = t$ . Και αυτό το βήμα εκτελείται για όλα τα ζεύγη  $(p, q)$ , γεγονός που σημαίνει ότι λαμβάνονται υπ' όψη και οι μεταβάσεις  $(p, q)$  και οι  $(q, p)$  (όταν  $p \neq q$ ). Το βήμα αυτό έχει επίδραση μόνον όταν στο αρχικό NFA υπάρχουν ακμές από τον κόμβο  $p$  στον  $r$  και από τον  $r$  στον  $q$ . Διαφορετικά, τόσο ο  $r$  όσο και οι προσκείμενες ακμές του μπορούν απλά να διαγραφούν. Επίσης, μπορούν να γίνουν κι άλλες απλοποιήσεις, όπως για παράδειγμα, όταν  $R_{r,r} = \emptyset$  μπορούν να αντικατασταθεί η  $R_{p,r}R_{r,r}^*R_{r,q}$  με  $R_{p,r}R_{r,q}$ . Ακόμα, όταν  $R_{p,q} = \emptyset$ , είναι  $R_{p,r}R_{r,r}^*R_{r,q}$ . Η σειρά εξάλειψης των κόμβων δεν έχει σημασία, αν και επηρεάζει το μέγεθος της τελικής έκφρασης. Ο αλγόριθμος τερματίζει όταν οι εναπομείναντες κόμβοι είναι μόνο οι  $s$  και  $t$ . Τότε η επιγραφή της ακμής μεταξύ των  $s$  και  $t$  είναι η κανονική έκφραση που συμβολίζει τη γλώσσα  $L(N)$ .

$$p \xrightarrow{R_{p,q} \cup R_{p,r}R_{r,r}^*R_{r,q}} q$$

**Για εξάσκηση...**

1. Μετατρέψτε το NFA του παραδείγματος 2.3.2 σε DFA χρησιμοποιώντας τον αλγόριθμο κατασλυσής υποσυνόλων.



2. Μετατρέψτε το DFA του παραδείγματος 2.3.9 σε κανονική έκφραση χρησιμοποιώντας την παραπάνω μέθοδο.

## 2.4 Ιδιότητες των κανονικών γλωσσών

### 2.4.1 Πράξεις που ορίζονται με γλώσσες

Αναφέρθηκε ήδη ότι οι γλώσσες (languages) είναι σύνολα οπότε οι πράξεις που γίνονται με σύνολα μπορούν να πραγματοποιηθούν και με γλώσσες. Έτσι, αν θεωρήσουμε τις γλώσσες  $L$ ,  $L_1$  και  $L_2$  ισχύουν τα ακόλουθα:

**Ορισμός 2.4.1.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , για δύο οποιεσδήποτε γλώσσες  $L_1, L_2$  ορισμένες στο  $\Sigma$ :

- Η ένωση  $L_1 \cup L_2$  των γλωσσών  $L_1$  και  $L_2$  είναι η γλώσσα  $L_1 \cup L_2 = \{w \in \Sigma^* | w \in L_1 \text{ ή } w \in L_2\}$ .
- Η τομή  $L_1 \cap L_2$  των γλωσσών  $L_1$  και  $L_2$  είναι η γλώσσα  $L_1 \cap L_2 = \{w \in \Sigma^* | w \in L_1 \text{ και } w \in L_2\}$ .
- Η διαφορά (ή σχετικό συμπλήρωμα (relative complement) )  $L_1 - L_2$  των γλωσσών  $L_1$  και  $L_2$  είναι η γλώσσα  $L_1 - L_2 = \{w \in \Sigma^* | w \in L_1 \text{ και } w \notin L_2\}$ .

Μια ειδική περίπτωση για τη διαφορά προκύπτει όταν  $L_1 = \Sigma^*$ , για την οποία το συμπλήρωμα  $\bar{L}$  της γλώσσας  $L$  ορίζεται σαν  $\bar{L} = \{w \in \Sigma^* | w \notin L\}$ .

**Ορισμός 2.4.2.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , για οποιεσδήποτε γλώσσες  $L_1, L_2$  ορισμένες στο  $\Sigma$ , η παράθεση  $L_1 L_2$  των γλωσσών  $L_1$  και  $L_2$  είναι η γλώσσα  $L_1 L_2 = \{w \in \Sigma^* | \exists u \in L_1, \exists v \in L_2, w = uv\}$ . Για κάθε γλώσσα  $L$ , ορίζουμε τη  $L^n$  ως εξής:

$$L^0 = \{\epsilon\},$$

$$L^{n+1} = L^n L.$$

Οι ακόλουθες ιδιότητες μπορούν εύκολα να επαληθευθούν:

1.  $L\emptyset = \emptyset$ ,
2.  $\emptyset L = \emptyset$ ,
3.  $L\{\epsilon\} = L$ ,
4.  $\{\epsilon\}L = L$ ,
5.  $(L_1 \cup \{\epsilon\})L_2 = L_1L_2 \cup L_2$ ,
6.  $(L_1(L_2 \cup \{\epsilon\})) = L_1L_2 \cup L_1$ ,
7.  $L^n L = LL^n$ .

**Παράδειγμα 2.4.1.**  $\{0, 11, 001\}$ ,  $\{\epsilon, 10\}$ , και  $\{0, 1\}^*$  είναι υποσύνολα του  $\{0, 1\}^*$ , και επομένως αποτελούν γλώσσες με βάση το αλφάβητο  $\{0, 1\}$ . Το κενό σύνολο  $\emptyset$  και το σύνολο  $\{\epsilon\}$  θεωρούνται γλώσσες βασισμένες σε κάθε αλφάβητο. Το  $\emptyset$  είναι μια γλώσσα που δεν περιέχει καμία συμβολοσειρά. Το  $\{\epsilon\}$  είναι μια γλώσσα που περιέχει μόνο την κενή συμβολοσειρά.

**Παράδειγμα 2.4.2.** Θεωρούμε τις γλώσσες  $L_1 = \{\epsilon, 0, 1\}$  και  $L_2 = \{\epsilon, 01, 11\}$ . Η ένωση των γλωσσών αυτών είναι η  $L_1 \cup L_2 = \{\epsilon, 0, 1, 01, 11\}$ , η τομή τους είναι  $L_1 \cap L_2 = \{\epsilon\}$ , και το συμπλήρωμα της  $L_1$  είναι  $L_1' = 00, 01, 10, 11, 000, 001, \dots$ . Για κάθε γλώσσα  $L$  ισχύουν:

$$\emptyset \cup L = L,$$

$$\emptyset \cap L = \emptyset.$$

**Παράδειγμα 2.4.3.** Έστω  $L_1 = \{\epsilon, 1, 01, 11\}$  και  $L_2 = \{1, 01, 101\}$  τότε  $L_1 - L_2 = \{\epsilon, 11\}$  ενώ  $L_2 - L_1 = \{101\}$ . Αν  $L_1 = \{\epsilon, 0, 1\}$  και  $L_2 = \{01, 11\}$ , τότε το καρτεσιανό γινόμενο των γλωσσών αυτών είναι  $L_1 \times L_2 = \{(\epsilon, 01), (\epsilon, 11), (0, 01), (0, 11), (1, 01), (1, 11)\}$ , ενώ η σύνθεσή τους είναι  $L_1 L_2 = \{01, 11, 001, 011, 101, 111\}$ .

Για κάθε γλώσσα  $L$  είναι  $L - \emptyset = L$ ,  $\emptyset - L = \emptyset$ ,  $\emptyset L = \emptyset$ , και  $\{\epsilon\}L = L$ .

Γενικά,  $L_1 L_2 \neq L_2 L_1$ . Οι πράξεις στις οποίες έχουμε αναφερθεί μέχρι τώρα διατηρούν την πεπερασμένη φύση των γλωσσών. Αυτό δεν ισχύει για τις ακόλουθες δύο πράξεις.

**Ορισμός 2.4.3.** Δεδομένου ενός αλφαβήτου  $\Sigma$ , για οποιαδήποτε γλώσσα  $L$  ορισμένη στο  $\Sigma$ , ορίζεται ως *Kleene \** Κλειστότητα (*Kleene \* closure*)  $L^*$  της γλώσσας  $L$  η γλώσσα

$$L^* = \bigcup_{n \geq 0} L^n.$$

Η *Kleene<sup>+</sup>* Κλειστότητα (*Kleene<sup>+</sup> closure*)  $L^+$  της γλώσσας  $L$  είναι η γλώσσα

$$L^+ = \bigcup_{n \geq 1} L^n.$$

Επομένως,  $L^*$  είναι η άπειρη ένωση

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots \cup L^n \cup \dots,$$

και  $L^+$  είναι η άπειρη ένωση

$$L^+ = L^1 \cup L^2 \cup \dots \cup L^n \cup \dots$$

Αφού  $L^1 = L$ , και η  $L^*$  και  $L^+$  περιέχουν τη γλώσσα  $L$ . Στην πραγματικότητα,

$$L^+ = \{w \in \Sigma^*, \exists n \geq 1, \exists u_1 \in L \dots \exists u_n \in L, w = u_1 \dots u_n\},$$

και αφού  $L^0 = \{\epsilon\}$ ,

$$L^* = \{\epsilon\} \cup \{w \in \Sigma^*, \exists n \geq 1, \exists u_1 \in L \dots \exists u_n \in L, w = u_1 \dots u_n\},$$

Επομένως, η γλώσσα  $L^*$  πάντα περιέχει την  $\epsilon$ , και είναι

$$L^* = L^+ \cup \{\epsilon\}.$$

Όμως, αν  $\{\epsilon\} \neq L$ , τότε  $\{\epsilon\} \neq L^+$ . Εύκολα αποδεικνύονται τα ακόλουθα:

$$\begin{aligned} \emptyset^* &= \{\epsilon\} \\ L^+ &= L^*L \\ (L^*)^* &= L^* \\ L^*L^* &= L^*. \end{aligned}$$

Η κλειστότητα ή Kleene Star μιας γλώσσας  $L$ , είναι το σύνολο

$$L^* = \emptyset \cup L \cup L \cup LL \cup LLL \cup LLLL \dots = L^0 \cup L^1 \cup \dots = L^k =$$

$$\{s = s_1s_2 \dots s_k | k = 0 \text{ και για κάθε } i, 0 < i \leq k, s_i \text{ ανήκει στη } L\},$$

δηλαδή το σύνολο των συμβολοσειρών που προκύπτουν από παράθεση μηδέν ή περισσότερων συμβολοσειρών της  $L$ , και είναι επίσης γλώσσα. Με  $L^i$  συμβολίζεται το σύνολο που αποτελείται από τις συμβολοσειρές εκείνες που μπορούν να προκύψουν από συνένωση (concatenation)  $i$  συμβολοσειρών της γλώσσας  $L$ , πρόκειται δηλαδή για τη σύνθεση  $i$  αντιγράφων της γλώσσας  $L$ , όπου ως  $L^0$  ορίζεται το  $\{\epsilon\}$ . Το σύνολο  $L^0 \cup L^1 \cup L^2 \cup L^3 \dots$ , ονομάζεται Kleene closure ή απλά κλειστότητα της γλώσσας  $L$ , συμβολίζεται με  $L^*$  και περιλαμβάνει τις συμβολοσειρές που μπορούν να προκύψουν από συνένωση τυχαιού αριθμού συμβολοσειρών της γλώσσας  $L$ . Η Kleene closure μιας γλώσσας περιέχει πάντα την κενή συμβολοσειρά,  $\epsilon$ , αφού η συμβολοσειρά  $\epsilon$  αντιστοιχεί στην κενή συνάρτηση, που είναι η μόνη συνάρτηση στη  $L^0$ , αν θεωρηθεί σαν το σύνολο όλων των συναρτήσεων από το κενό σύνολο  $\{\}$  στη  $L$ . Αυτός ο ορισμός είναι ισοδύναμος με την έννοια του δυναμοσυνόλου κάποιου



συνόλου που περιέχει πάντα το κενό σύνολο. Συγκεκριμένα, είναι  $0^* = \{\epsilon\}$ .

Η θετική κλειστότητα (positive closure) μιας γλώσσας  $L$ , είναι το σύνολο

$$L^+ = L \cup L \cup LL \cup LLL \cup LLLL \dots = L^1 \cup L^2 \cup L^3,$$

δηλαδή το σύνολο των συμβολοσειρών που προκύπτουν από παράθεση τουλάχιστο μιας ή περισσότερων συμβολοσειρών της  $L$  και είναι επίσης γλώσσα.

**Παράδειγμα 2.4.4.** Θεωρούμε τις γλώσσες  $L_1 = \{\epsilon, 0, 1\}$  και  $L_2 = \{01, 11\}$ . Για τις γλώσσες αυτές είναι

$$L_1^2 = \{\epsilon, 0, 1, 00, 01, 10, 11\},$$

και

$$L_2^3 = \{010101, 010111, 011101, 011111, 110101, 110111, 111101, 111111\}.$$

Επιπλέον,  $\epsilon$  ανήκει στη  $L_1^*$ , στη  $L_1^+$ , και στη  $L_2^*$  αλλά όχι στη  $L_2^+$ .

Κάθε γλώσσα που μπορεί να οριστεί από ένα τυπικό σύστημα (formal system), ένα σύστημα, δηλαδή, που έχει πεπερασμένο αριθμό αξιωμάτων και πεπερασμένο αριθμό κανόνων ονομάζεται τυπική γλώσσα (formal language).

Στη συνέχεια, παρουσιάζουμε ιδιότητες κλειστότητας στην κλάση των κανονικών γλωσσών.

#### 2.4.2 Ιδιότητες κλειστότητας

Ένα σύνολο είναι κλειστό ως προς μία πράξη αν εφαρμόζοντας την πράξη αυτή μεταξύ στοιχείων του συνόλου προκύπτει πάλι στοιχείο που ανήκει στο σύνολο. Για παράδειγμα, το σύνολο των ακεραίων είναι κλειστό ως προς τον πολλαπλασιασμό αλλά όχι ως προς τη διαίρεση.

**Πρόταση 2.4.1.** Στην κλάση των κανονικών γλωσσών ισχύει η κλειστότητα ως προς τις πράξεις Kleene (ένωση, παράθεση και star).

Ας υποθέσουμε ότι οι γλώσσες  $L_1$  και  $L_2$  είναι κανονικές. Τότε σύμφωνα με το παραπάνω κάθε μία από τις γλώσσες  $L_1 \cup L_2$ ,  $L_1 L_2$  και  $L_1^*$  είναι επίσης κανονικές.

*Απόδειξη.* Ο ευκολότερος τρόπος να προχωρήσουμε είναι να δείξουμε ότι οι κανονικές εκφράσεις για τις γλώσσες  $L_1$  και  $L_2$  μπορούν να συνδυασθούν ή να τροποποιηθούν ώστε να δώσουν την κανονική έκφραση για το συνδυασμό των γλωσσών. Για παράδειγμα, η κανονική έκφραση για τη γλώσσα  $L_1 L_2$  δίνεται αν παραθέσουμε τις κανονικές εκφράσεις για τις γλώσσες  $L_1$  και  $L_2$ .  $\square$

**Πρόταση 2.4.2.** Στην κλάση των κανονικών γλωσσών ισχύει η κλειστότητα ως προς τις πράξεις (α) συμπλήρωση, και (β) τομή.

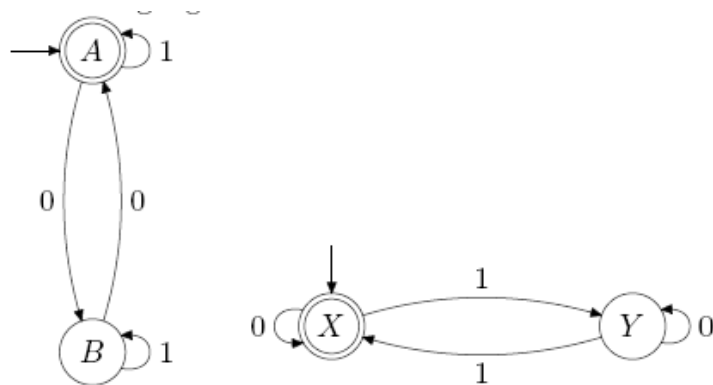
Απόδειξη. (α) Αν η γλώσσα  $L$  είναι κανονική, τότε και το συμπλήρωμά της είναι κανονική γλώσσα. Το συμπλήρωμα μιας γλώσσας συμβολίζεται  $\bar{L}$  είναι το σύνολο που περιέχει όλες τις συμβολοσειρές που δεν ανήκουν στη γλώσσα  $L$  αλλά προκύπτουν με βάση το ίδιο αλφάβητο. Ας θεωρήσουμε το ντετερμινιστικό πεπερασμένο αυτόματο για τη  $L$ . Για να κατασκευάσουμε ένα FA για τη γλώσσα  $\bar{L}$ , απλά εναλλάσσουμε τελικές και μη τελικές καταστάσεις. (β) Αν οι γλώσσες  $L_1$  και  $L_2$  είναι κανονικές, τότε και η τομή τους  $L_1 \cap L_2$  είναι κανονική γλώσσα. Για να δούμε γιατί ισχύει αυτό, μπορούμε να χρησιμοποιήσουμε το προηγούμενο μέρος αφού με βάση του νόμους de Morgan:  $L_1 \cap L_2 = \overline{(\bar{L}_1 \cup \bar{L}_2)}$  Έτσι έχουμε τον τρόπο να κατασκευάσουμε ένα FA για τη  $L_1 \cap L_2$ .  $\square$

Η απόδειξη του μέρους (β) παρέχει ουσιαστικά έναν αλγόριθμο για την κατασκευή ενός DFA για τη γλώσσα  $L_1 \cap L_2$  δεδομένων των DFA για τις  $L_1$  και  $L_2$ . Δηλαδή, χρησιμοποιούμε τα DFA για τις  $L_1$  και  $L_2$ . Εναλλάσσουμε τελικές και μη τελικές καταστάσεις για να πάρουμε τα DFA για τις  $\bar{L}_1$  και  $\bar{L}_2$  και μετά υλοποιούμε την ένωσή τους (προσθέτουμε μία καινούρια αρχική κατάσταση και τις  $\epsilon$ -μεταβάσεις). Χρησιμοποιώντας τον αλγόριθμο κατασκευής υποσυνόλων μπορούμε να πάρουμε το ντετερμινιστικό αυτόματο

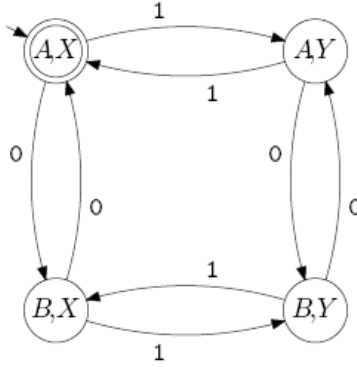
Υπάρχει και άλλος αλγόριθμος για την κατασκευή του αυτομάτου για την τομή που ονομάζεται κατασκευή γινομένου. Η ιδέα προέρχεται από το ότι αυτό που θα θέλαμε να κάνουμε είναι να δώσουμε τη συμβολοσειρά σαν είσοδο ταυτόχρονα και στα δύο επιμέρους αυτόματα. Η λύση είναι να παρακολουθούμε τα ζεύγη καταστάσεων και των δύο αυτομάτων.

Συγκεκριμένα, το αυτόματο για το γινόμενο έχει μία κατάσταση για κάθε ζεύγος καταστάσεων των αρχικών αυτομάτων. Αρχική κατάσταση είναι το ζεύγος των αρχικών καταστάσεων των αρχικών αυτομάτων ενώ τελικές καταστάσεις είναι αυτές που περιέχουν τελικές καταστάσεις και των δύο αρχικών αυτομάτων.

**Παράδειγμα 2.4.5.** Για παράδειγμα, έστω  $L_1$  το σύνολο των συμβολοσειρών με άρτιο αριθμό 0, και  $L_2$  το σύνολο των συμβολοσειρών με άρτιο αριθμό 1. Τότε τα FA για αυτές τις γλώσσες έχουν και τα δύο από δύο καταστάσεις:



Άρα το FA για την τομή  $L_1 \cap L_2$  έχει τέσσερις καταστάσεις:



Ο τυπικός ορισμός ενός FA δείχνει ότι η παραπάνω κατασκευή είναι περιεκτική. Καρτεσιανό γινόμενο δύο συνόλων  $A$  και  $B$ , που συμβολίζεται  $A \times B$ , είναι το σύνολο  $\{(a, b) : a \in A, b \in B\}$ .

Έστω η  $L_1$  αναγνωρίζεται από το αυτόματο  $M_1$  που είναι η 5-άδα  $(Q_1, \Sigma, \delta_1, q_1, F_1)$  και η  $L_2$  αναγνωρίζεται από το αυτόματο  $M_2$  που είναι η 5-άδα  $(Q_2, \Sigma, \delta_2, q_2, F_2)$ . Τότε η  $L_1 \cap L_2$  αναγνωρίζεται από το αυτόματο  $(Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$  με  $\delta((r, s), x) = (\delta_1(r, x), \delta_2(s, x))$ .

### 2.4.3 Κατασκευαστικές αποδείξεις

Μια γλώσσα  $L$  είναι *κανονική* (*Regular*) αν γίνεται δεκτή από κάποιο DFA. Μια κανονική γλώσσα μπορεί να είναι αποδεκτή από πολλά, διαφορετικά DFAs.

Η κλάση των κανονικών γλωσσών είναι κλειστή ως προς τις πράξεις Ένωση, Παράθεση, Kleene Star, Τομή και Συμπλήρωση.

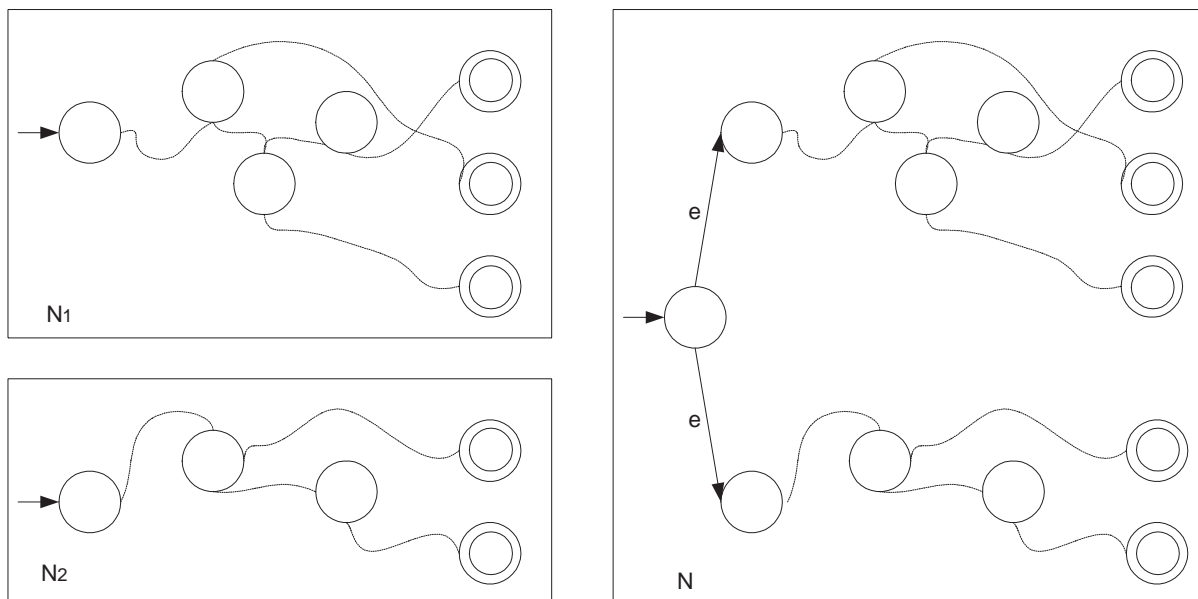
#### Ένωση

Μπορούμε τώρα να τροποποιήσουμε το  $D$  για να δέχεται τη γλώσσα  $L(D_1) \cup L(D_2)$ . Αλλάζουμε τα σύνολο των τελικών καταστάσεων ώστε να γίνει  $(F_1 \times Q_2) \cup (Q_1 \times F_2)$ . Πράγματι,

$$\begin{aligned}
 w \in L(D_1) \cup L(D_2) &\Leftrightarrow w_1 \in L(D_1) \text{ ή } w_2 \in L(D_2), \\
 &\Leftrightarrow \delta_1^*(q_{0,1}, w) \in F_1 \text{ ή } \delta_2^*(q_{0,2}, w) \in F_2, \\
 &\Leftrightarrow \delta^*((q_{0,1}, q_{0,2}), w) \in (F_1 \times Q_2) \cup (Q_1 \times F_2), \\
 &\Leftrightarrow w \in L(D).
 \end{aligned}$$

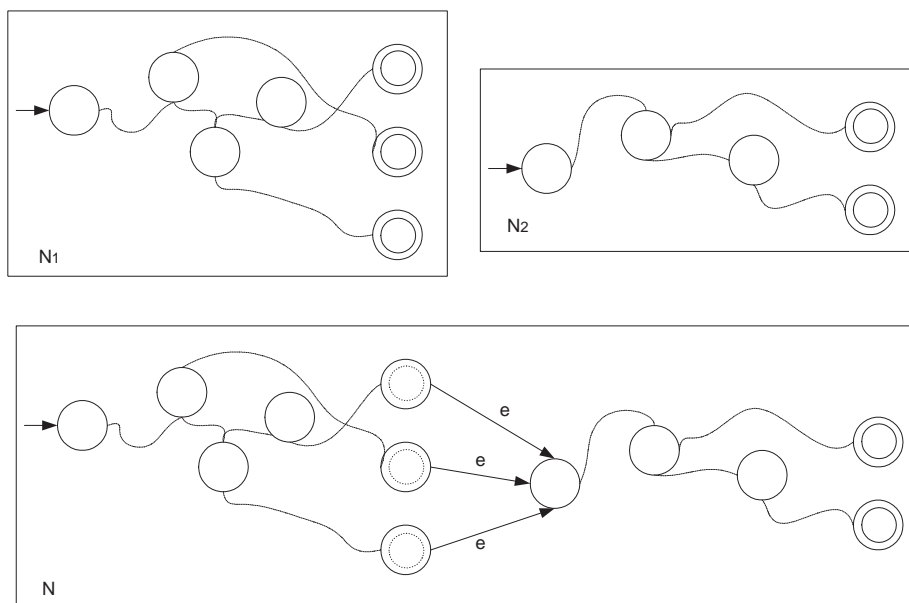
Επομένως,  $L(D) = L(D_1) \cup L(D_2)$ .

Έστω δύο NFAs,  $N_1$  και  $N_2$ . Μπορούμε να κατασκευάσουμε ένα άλλο NFA,  $N$ , τέτοιο ώστε  $L(N) = L(N_1) \cup L(N_2)$ , δημιουργώντας μια νέα κατάσταση και εισάγοντας μεταβάσεις με την κενή συμβολοσειρά από την καινούρια κατάσταση προς τις αρχικές καταστάσεις κάθε αυτομάτου. Η νέα κατάσταση αποτελεί την αρχική κατάσταση του αυτομάτου  $N$ .



### Παράθεση

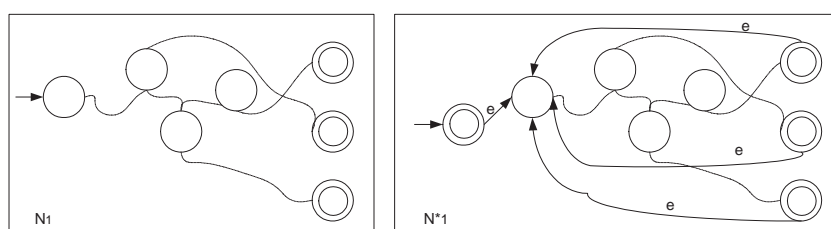
Έστω δύο NFAs,  $N_1$  και  $N_2$ . Μπορούμε να κατασκευάσουμε ένα άλλο NFA,  $N$ , τέτοιο ώστε  $L(N) = L(N_1) \circ L(N_2)$ . Αυτό γίνεται με τη μετατροπή όλων των τελικών καταστάσεων του  $N_1$  σε μη τελικές και την εισαγωγή μεταβάσεων με την κενή συμβολοσειρά από κάθε μια από αυτές στην αρχική κατάσταση του  $N_2$ . Η αρχική κατάσταση του νέου αυτομάτου  $N$  είναι η ίδια με την αρχική κατάσταση του  $N_1$ .



### Kleene Star

Έστω ένα NFA,  $N_1$ , μπορούμε να κατασκευάσουμε ένα NFA,  $N$ , τέτοιο ώστε  $L(N) = L(N_1)^*$ . Αυτό γίνεται με τη δημιουργία μιας καινούριας κατάστασης, την εισαγωγή μεταβάσεων με την κενή συμβολοσειρά από αυτή τη νέα κατάσταση και από όλες τις τελικές καταστάσεις προς την αρχική κατάσταση και τη μετατροπή της νέας κατάστασης σε αρχική για το νέο αυτόματο.

Εφαρμόζοντας την πράξη Kleene Star στο παραπάνω αυτόματο έχουμε:



### Τομή

Αν η γλώσσα  $L_1$  είναι κανονική (αναγνωρίζεται δηλαδή από κάποιο DFA  $M_1$ ), και η γλώσσα  $L_2$  είναι κανονική (αναγνωρίζεται δηλαδή από κάποιο DFA  $M_2$ ), και οι δύο γλώσσες  $L_1$  και  $L_2$  ορίζονται στο ίδιο αλφάβητο, τότε και η γλώσσα  $L_1 \cap L_2$  είναι επίσης κανονική (αναγνωρίζεται δηλαδή από κάποιο DFA  $M'$ ). Μπορούμε εναλλακτικά να πούμε ότι στην κλάση των κανονικών γλωσσών ισχύει η κλειστότητα ως προς την τομή.

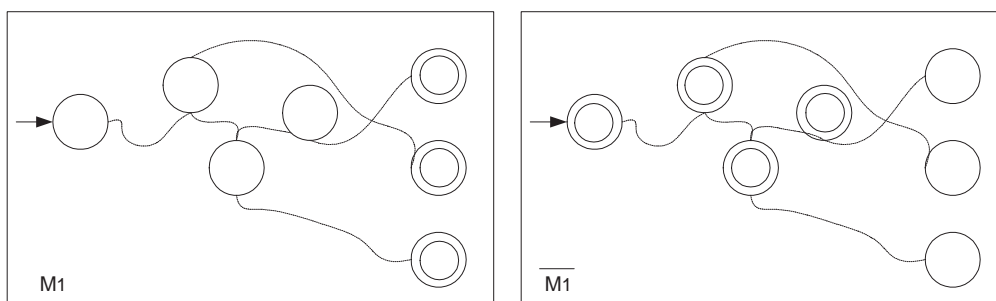
**Απόδειξη:** Θα κάνουμε την απόδειξη κατασκευαστικά. Προκειμένου να ισχύει το θεώρημα, θα πρέπει το αυτόματο  $M'$  να αναγνωρίζει κάποια συμβολοσειρά εισόδου όταν και το  $M_1$  και το  $M_2$  την αναγνωρίζουν. Για να δούμε πότε κάτι τέτοιο ισχύει θα πρέπει να παρακολουθήσουμε τουα υπολογισμούς των δύο αυτομάτων ταυτόχρονα καθώς επεξεργάζονται την είσοδο. Το νέο αυτόματο  $M'$  θα πρέπει να έχει σα σύνολο καταστάσεων όλους τους πιθανούς συνδυασμούς των καταστάσεων των  $M_1$  και  $M_2$ , ή αλλιώς το σύνολο  $Q_1 \times Q_2$ . Δεν διαθέτει μία μόνο αρχική κατάσταση, αλλά δύο  $(q_{0,1}, q_{0,2})$ . Το σύνολο των τελικών του καταστάσεων δεν είναι μόνον το  $F$ , αλλά το σύνολο όλων των δυνατών συνδυασμών των τελικών καταστάσεων του  $M_1$  και του  $M_2$ , ή διαφορετικά το σύνολο  $F_1 \times F_2$ . Επιπλέον, θα διαθέτει μια νέα συνάρτηση μεταβάσεων, που ορίζεται ως  $\delta' = ((q, q'), \sigma) \rightarrow (\delta_1(q, \sigma), \delta_2(q', \sigma))$ . Συνολικά, το αυτόματο  $M'$  θα οριζόταν ως  $(Q_1 \times Q_2, \Sigma, \delta', (q_{0,1}, q_{0,2}), F_1 \times F_2)$ . Θεωρούμε τώρα μια συμβολοσειρά  $w$ , τέτοια ώστε  $w \in L_1 \cap L_2$ . Σε κάθε διαδοχική μετάβαση για κάθε σύμβολο της συμβολοσειράς  $w$ ,  $w_1, \dots, w_n$ , ο υπολογισμός θα ξεκινούσε από την κατάσταση  $(q_{0,1}, q_{0,2})$ , θα προχωρούσε με το σύμβολο  $w_1$  στην κατάσταση  $(q_{1,1}, q_{1,2})$ , κτλ., μέχρι το αυτόματο να καταλήξει στην τελική κατάσταση  $(q_{n,1}, q_{n,2})$ . Αφού η  $q_{n,1}$  είναι τελική κατάσταση του  $M_1$  και η  $q_{n,2}$  είναι τελική κατάσταση του  $M_2$ , η συμβολοσειρά αναγνωρίζεται από το αυτόματο  $M'$ , και ισχύει  $L_1 \cap L_2 \subseteq \ell(M')$ .

Προκειμένου να εξετάσουμε και το αντίστροφο για να ολοκληρώσουμε την απόδειξη, υποθέτουμε

ότι  $w \in \ell(M')$ , και πρέπει να δείξουμε ότι ισχύει  $w \in L_1 \cap L_2$ . Υποθέτουμε λοιπόν ότι ισχύει  $w \in L_1 \cap L_2$  και χωρίς απώλεια της γενικότητας ότι  $w \notin L_1$ . Εξετάζοντας τώρα τον υπολογισμό του αυτομάτου  $M'$ , παρατηρούμε ότι η κατάσταση  $q_{n,1}$  δεν είναι τελική και κατά συνέπεια  $w \notin \ell(M')$  και  $\ell(M')$ .

### Συμπλήρωμα

Έστω ένα DFA,  $M_1$ , η συμπληρωματική γλώσσα ορίζεται σαν  $\tilde{L}(M_1) = S^* - L(M_1)$ . Δηλαδή πρόκειται για ακριβώς την αντίθετη της γλώσσας που γίνεται αποδεκτή από το  $M_1$  και μπορεί να επιτευχθεί με μετατροπή όλων των τελικών καταστάσεων του  $M_1$  σε μη τελικές και αντίστροφα.



## 2.5 Μη κανονικές γλώσσες

Υπάρχουν αρκετοί περιορισμοί στο είδος των προβλημάτων αναγνώρισης που μπορούμε να χειριστούμε με τα πεπερασμένα αυτόματα. Για να αποδείξουμε ότι μια γλώσσα δεν είναι κανονική απαιτούνται επιπλέον ιδέες που περιλαμβάνονται στην έννοια των διακρίσιμων συμβολοσειρών και το Pumping Lemma.

### 2.5.1 Διακρίσιμες συμβολοσειρές

Φυσικά, τα ζητήματα σχετικά με κλειστότητα δεν θα είχαν νόημα αν δεν υπήρχαν και γλώσσες που δεν είναι κανονικές.

**Παράδειγμα 2.5.1.** Θεωρούμε τη συμβολοσειρά  $0^n 1^n$ : ο συμβολισμός σημαίνει ότι  $n$  0 ακολουθούνται από  $n$  1. Ορίζουμε το σύνολο  $B$  ως εξής:

$$B = \{0^n 1^n : n \geq 0\} = \{\epsilon, 01, 0011, 000111, \dots\}$$

Δεν υπάρχει FA για τη γλώσσα αυτή. Ένα επιχείρημα είναι το ακόλουθο. Υποθέτουμε ότι η συμβολοσειρά εισόδου είναι σωστή με την έννοια ότι είναι μια σειρά 0 που ακολουθείται από μια σειρά 1. Η μηχανή πρέπει να μετρήσει τον αριθμό των 0: δηλαδή, όταν τελειώσει η ακολουθία με τα 0 πρέπει να είναι σε μια κατάσταση μοναδική σχετικά με το πλήθος των 0 που διαβάστηκαν.

Αλλά η μηχανή έχει σταθερή πεπερασμένη μνήμη, ενώ η συμβολοσειρά εισόδου μπορεί να είναι αυθαίρετα μεγάλη, και επίσης ο αριθμός των 0 μπορεί να είναι αυθαίρετα μεγάλος. Άρα, δεν μπορεί να κατασκευαστεί FA για τη γλώσσα  $B$ , οπότε η  $B$  δεν είναι κανονική.

Η ιδέα είναι ότι ένα FA διαθέτει σταθερό ποσό μνήμης: την πληροφορία που περιέχεται στην κάθε κατάστασή του. Ένα FA αντιστοιχεί σε έναν υπολογιστή χωρίς εξωτερική μνήμη, που έχει μόνο ένα προκαθορισμένο σύνολο bits που μπορεί να διαχειριστεί. Παρατηρήστε ότι μπορούμε να χρησιμοποιήσουμε όσο μεγάλη τέτοια μνήμη επιθυμούμε, αλλά άπαξ και κατασλεύασουμε το FA, αυτή η μνήμη δεν μπορεί να τροποποιηθεί.

Για να γίνουμε πιο σαφείς χρησιμοποιούμε την ιδέα των διακρίσιμων (distinguishable) συμβολοσειρών. Δύο συμβολοσειρές  $x$  και  $y$  είναι μη διακρίσιμες ως προς μια γλώσσα  $L$  αν για κάθε συμβολοσειρά  $z$ , ισχύει ότι  $xz \in L$  αν και μόνον αν  $yz \in L$ . Διαφορετικά, είναι διακρίσιμες. Δηλαδή, οι  $x$  και  $y$  είναι διακρίσιμες ως προς τη γλώσσα  $L$  αν υπάρχει κάποια κατάληξη  $z$  που μπορεί να προστεθεί έτσι ώστε  $xz \in L$  και  $yz \notin L$  και αντίστροφα.

**Θεώρημα 2.5.1.** Έστω  $M$  ένα FA που αναγνωρίζει τη γλώσσα  $L$ , και έστω  $x$  και  $y$  διακρίσιμες συμβολοσειρές ως προς τη γλώσσα  $L$ . Τότε το  $M$  πρέπει να είναι σε διαφορετική κατάσταση αφού διαβάσει το  $x$  από αυτή που θα είναι αφού διαβάσει το  $y$ .

*Απόδειξη.* Υποθέτουμε ότι οι συμβολοσειρές  $x$  και  $y$  οδηγούν το αυτόματο  $M$  στην ίδια κατάσταση. Τότε για κάθε συμβολοσειρά  $z$  οι  $xz$  και  $yz$  οδηγούν το  $M$  στην ίδια κατάσταση. Επομένως αν μία από τις  $xz$  και  $yz$  γίνεται αποδεκτή ενώ η άλλη απορρίπτεται υπάρχει πρόβλημα. Η μόνη λύση είναι ότι οι  $x$  και  $y$  πρέπει να οδηγούν το αυτόματο  $M$  σε διαφορετικές καταστάσεις: η μηχανή πρέπει να θυμάται ποια από τις  $x$  ή  $y$  έχει διαβάσει.  $\square$

Ένα σύνολο συμβολοσειρών είναι διακρίσιμο ανά ζεύγη αν κάθε δύο συμβολοσειρές του συνόλου είναι διακρίσιμες. Με βάση τη λογική που περιγράψαμε παραπάνω, θα πρέπει να υπάρχει μια διαφορετική κατάσταση για κάθε συμβολοσειρά του συνόλου:

**Πόρισμα 2.5.1.** Αν  $D_L$  είναι ένα διακρίσιμο ως προς μια γλώσσα  $L$  ανά ζεύγη σύνολο συμβολοσειρών, τότε κάθε FA για τη  $L$  έχει τουλάχιστον  $|D_L|$  καταστάσεις. Ειδικότερα, αν το  $D_L$  είναι άπειρο σύνολο τότε η γλώσσα  $L$  δεν είναι κανονική.

**Παράδειγμα 2.5.2.** Συνέχεια παραδείγματος 2.5.1. Επιστρέφουμε στη γλώσσα  $B = \{0^n 1^n : n \geq 0\}$ . Ισχυριζόμαστε ότι το σύνολο  $D_B = \{0^j : j \geq 0\}$  είναι διακρίσιμο ανά ζεύγη. Ας θεωρήσουμε δύο οποιεσδήποτε συμβολοσειρές του  $D_B$ : έστω  $0^j$  και  $0^{j'}$  με  $j \neq j'$ . Τότε, προσθέτοντας  $1^j$  στο τέλος της πρώτης παίρνουμε συμβολοσειρά που ανήκει στη  $B$ , ενώ προσθέτοντας  $1^j$  στο τέλος της δεύτερης παίρνουμε συμβολοσειρά που δεν ανήκει στη  $B$ : δηλαδή,  $0^j$  και  $0^{j'}$  είναι διακρίσιμες. Επειδή το σύνολο  $D_B$  είναι άπειρο, συμπεραίνουμε ότι η γλώσσα  $B$  δεν είναι κανονική.

**Παράδειγμα 2.5.3.** Θεωρούμε τη γλώσσα του παραδείγματος ;;. Το σύνολο  $\{\epsilon, 0, 1, 01\}$  είναι διακρίσιμο ανά ζεύγη. Επομένως το FA που κατασκευάσαμε έχει τις λιγότερες δυνατές καταστάσεις.

Δείξαμε, λοιπόν, ότι αν υπάρχει άπειρο σύνολο διακρίσιμων συμβολοσειρών για μια γλώσσα, τότε η γλώσσα αυτή δεν είναι κανονική. Υπάρχει και το αντίστροφο: δηλαδή, αν μια γλώσσα δεν είναι κανονική, τότε σίγουρα υπάρχει ένα τέτοιο άπειρο σύνολο. Το αποτέλεσμα αυτό είναι μέρος το λεγόμενου θεωρήματος των Myhill - Nerode. Ουσιαστικά, υπάρχει σχέση μεταξύ του ελάχιστου αριθμού καταστάσεων ενός FA και του μέγιστου συνόλου διακρίσιμων ανά ζεύγη συμβολοσειρών.

## 2.5.2 Το Pumping Lemma

Αν προσπαθήσουμε να κατασκευάσουμε ένα πεπερασμένο αυτόματα που να αναγνωρίζει τη γλώσσα  $\{0^n 1^n, n \geq 0\}$ , δεν θα τα καταφέρουμε, γιατί:

- είτε θα μπορούμε ενδεχομένως να αναγνωρίσουμε τη γλώσσα έχοντας στη διάθεσή μας ένα αυτόματο με άπειρες καταστάσεις (δηλ. μη πεπερασμένο),
- είτε θα μπορούμε να κατασκευάσουμε ένα πεπερασμένο αυτόματο που να αναγνωρίζει τη γλώσσα  $\{0^n 1^n, n \geq 0\}$  αλλά και πολλές άλλες συμβολοσειρές.

Το Pumping Lemma για κανονικά σύνολα είναι ένας τρόπος για να 'πειστούμε' ότι ένα σύνολο ΔΕΝ είναι κανονικό, και μπορεί να παρομοιαστεί με μια 'συζήτηση' μεταξύ ενός παίκτη κι ενός αντιπάλου.

**Παίκτης:** Το σύνολο  $\{0^n 1^n, n \geq 0\}$  δεν είναι κανονικό.

**Αντίπαλος:** Έχω ένα πεπερασμένο αυτόματο που το αναγνωρίζει.

**Παίκτης:** Πόσες καταστάσεις έχει το πεπερασμένο αυτόματο;

**Αντίπαλος:** 238

**Παίκτης:** Δώσε στο πεπερασμένο αυτόματο σαν είσοδο τη συμβολοσειρά  $0^{238}1^{238}$ .

Στο σημείο αυτό ο **Παίκτης** κάνει την εξής παρατήρηση: κάπου ανάμεσα στα πρώτα 238 σύμβολα το αυτόματο του αντιπάλου θα πρέπει να περάσει από την ίδια κατάσταση για δεύτερη φορά. Αυτό θα συμβεί λόγω της Αρχής του Περιστεριώνα γιατί ξεκινώντας από την αρχική κατάσταση πρέπει να γίνουν 238 κινήσεις, ενώ υπάρχουν μόνο 237 διαφορετικές καταστάσεις στις οποίες μπορεί να γίνει μετάβαση. Άρα στη διαδρομή του πεπερασμένου αυτομάτου θα σχηματιστεί ένας 'κύκλος' μεταξύ της πρώτης και δεύτερης φοράς που θα πραγματοποιηθεί μετάβαση στην ίδια κατάσταση.

**Παίκτης:** Ποιο είναι το μήκος αυτού του 'κύκλου' (στον οποίο όλες οι μεταβάσεις γίνονται με το σύμβολο '0');



**Αντίπαλος:** Το μήκος αυτού του ‘κύκλου’ είναι 32.

**Παίκτης:** Τότε το πεπερασμένο αυτόματο που έχεις, θα αναγνωρίζει και τη συμβολοσειρά  $0^{238+32}1^{238}$  που δεν είναι της μορφής  $0^n1^n$ .

Υπάρχουν διάφορες τεχνικές προκειμένου να αποδειχθεί η κανονικότητα μιας γλώσσας. Επίσης υπάρχουν ανάλογες τεχνικές για την απόδειξη της μη κανονικότητας γλωσσών. Η τεχνική που θα περιγραφεί στη συνέχεια βασίζεται στην ακόλουθη ιδιότητα όλων των κανονικών γλωσσών:

**Ιδιότητα 2.5.1.** Κάθε συμβολοσειρά που ανήκει σε μια κανονική γλώσσα, με μέγεθος μεγαλύτερο από μια συγκεκριμένη τιμή, έχει ένα τμήμα που μπορεί να επαναληφθεί άπειρες φορές, δίνοντας μια νέα συμβολοσειρά που επίσης ανήκει στη γλώσσα.

Η δημιουργία νέων συμβολοσειρών με τον τρόπο που περιγράφηκε λέγεται άντληση (pumping). Οδηγούμαστε έτσι διαισθητικά στο συμπέρασμα ότι η αποδοχή μιας συμβολοσειράς, που παράγεται με άντληση, από κάποιο πεπερασμένο αυτόματο, DFA ή NFA, προϋποθέτει την ύπαρξη κύκλου στο μονοπάτι που σχηματίζουν οι μεταβάσεις του αυτομάτου. Επανάληψη του ίδιου κύκλου για άπειρο αριθμό φορών αποκλείεται να παράγει μια συμβολοσειρά που δεν ανήκει στη γλώσσα. Το Pumping Lemma δηλώνει ότι κάθε κανονική γλώσσα έχει την παραπάνω ιδιότητα. Επομένως, προκειμένου να αποδείξουμε ότι μια γλώσσα δεν είναι κανονική, αρκεί να αποδείξουμε ότι η γλώσσα δεν έχει την παραπάνω ιδιότητα. Ο παραπάνω συλλογισμός παρουσιάζεται στη συνέχεια με μορφή διαδοχικών βημάτων: Αν μια άπειρη γλώσσα είναι κανονική μπορεί να οριστεί από κάποιο DFA. Το DFA έχει έναν πεπερασμένο αριθμό καταστάσεων (έστω,  $n$ ). Αφού η γλώσσα είναι άπειρη, θα υπάρχουν συμβολοσειρές που ανήκουν στη γλώσσα με μήκος μεγαλύτερο του  $n$ . Για μια συμβολοσειρά με μήκος μεγαλύτερο του  $n$  που γίνεται αποδεκτή από το DFA, η διαδρομή που σχηματίζουν οι μεταβάσεις του DFA πρέπει να περιέχει κύκλο. Επανάληψη, για τυχαίο αριθμό φορών, του κύκλου αυτού θα οδηγεί στην παραγωγή νέας συμβολοσειράς που θα γίνεται επίσης αποδεκτή από το DFA.

Το Pumping Lemma δηλώνει ότι κάθε κανονική γλώσσα έχει μια συγκεκριμένη επαναληπτικότητα. Στη συνέχεια, θα χρησιμοποιήσουμε τον ακόλουθο συμβολισμό. Αν  $v$  είναι μια συμβολοσειρά, τότε συμβολίζουμε με  $|v|$  το μήκος της  $v$ . Επιλέον, συμβολίζουμε με  $v^i$   $i$  διαδοχικά αντίγραφα του  $v$ .

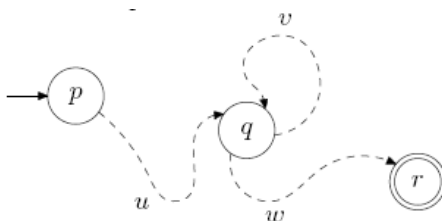
**Θεώρημα 2.5.2 (Pumping Lemma).** Έστω  $L$  μια κανονική γλώσσα που αναγνωρίζεται από κάποιο DFA με  $p$  καταστάσεις. Τότε, σε κάθε συμβολοσειρά  $w \in L$  με τουλάχιστον  $p$  σύμβολα, μπορεί να βρεθεί τμήμα που όταν επαναλαμβάνεται δίνει συμβολοσειρές που επίσης ανήκουν στη γλώσσα  $L$ . Δηλαδή, η  $w$  μπορεί να χωριστεί σε τρία τμήματα  $xyz$  έτσι ώστε:

- το τμήμα  $y$  δεν είναι η κενή συμβολοσειρά,
- $|xy| \leq p$ , και

- $xy^iz \in L \forall i \geq 0$ .

Απόδειξη. Παρατηρούμε την ακολουθία καταστάσεων από τις οποίες περνάει το DFA με είσοδο  $w$ . Έστω  $q$  η πρώτη κατάσταση που επαναλαμβάνεται. Αυτή η επανάληψη πρέπει να συμβεί όταν έχουν διαβαστεί  $p$  σύμβολα (αφού αυτό σημαίνει ότι το αυτόματο χρειάζεται  $p + 1$  καταστάσεις). Τότε χωρίζουμε τη συμβολοσειρά  $w$  ως εξής:

- το τμήμα  $x$  περιέχει τα σύμβολα που οδηγούν το αυτόματο για πρώτη φορά στην κατάσταση  $q$ ,
- το τμήμα  $y$  περιέχει τα σύμβολα που καταναλώνονται μεταξύ της πρώτης και της δεύτερης φοράς που το αυτόματο βρίσκεται στην κατάσταση  $q$ , και
- το τμήμα  $z$  περιέχει τα υπόλοιπα σύμβολα της εισόδου.



Το σημαντικό σημείο είναι ότι το DFA παραμένει στην ίδια κατάσταση  $q$  ανεξάρτητα από το αν έχει διαβάσει το τμήμα  $x$ , ή το  $xy$ , ή το  $xy^2$  κ.ο.κ. (είναι ντετερμινιστικό). Άρα το DFA παραμένει στην κατάσταση  $r$  ανεξάρτητα από το αν έχει διαβάσει το τμήμα  $xz$ , ή το  $xyz$  ή το  $xy^2z$ . Επομένως,  $xy^iz$  ανήκει στη γλώσσα  $L$  για κάθε  $i \geq 0$ .  $\square$

Το Pumping Lemma δεν μπορεί να χρησιμοποιηθεί για να δείξουμε ότι μια γλώσσα είναι κανονική. Αντίθετα, χρησιμοποιείται για να δείξουμε ότι μια γλώσσα δεν είναι κανονική, δείχνοντας ότι καταλήγουμε σε άτοπο αν ξεκινήσουμε με την υπόθεση ότι το Λήμμα ισχύει. Η επιτυχία της χρήσης του Pumping Lemma συνίσταται στην επιλογή κατάλληλης συμβολοσειράς  $w$  της οποίας η επανάληψη οδηγεί στο άτοπο.

Συγκεκριμένα χρησιμοποιούμε την ακόλουθη διατύπωση:

Αν για κάθε  $m$  υπάρχει ένα  $w \in L$  με  $|w| \geq m$ , τέτοιο ώστε για κάθε σχηματισμό  $w = xyz$  αν ικανοποιούνται οι συνθήκες  $|xy| \leq m$  και  $|y| \geq 1$ , τότε υπάρχει  $i \geq 0$  τέτοιο ώστε η συμβολοσειρά  $xy^iz \in L$ . Η παραπάνω πρόταση συνεπάγεται ότι η γλώσσα  $L$  δεν είναι κανονική. Απλούστερα, τα παραπάνω σημαίνουν τα εξής: το  $m$  είναι ένας πεπερασμένος αριθμός που έχει επιλεγεί έτσι ώστε οι συμβολοσειρές με μήκος  $m$  ή μεγαλύτερο να περιέχουν κύκλο. Κατά συνέπεια το  $m$  πρέπει να είναι τουλάχιστο ίσο με τον αριθμό καταστάσεων του DFA. Υπενθυμίζουμε ότι το DFA δεν είναι γνωστό, οπότε δεν μπορούμε στην πραγματικότητα να επιλέξουμε το  $m$ . Απλά γνωρίζουμε ότι ένα τέτοιο  $m$

υπάρχει. Αφού η συμβολοσειρά  $w$  έχει μήκος τουλάχιστο  $m$ , μπορούμε να την “σπάσουμε” σε δύο μέρη-συμβολοσειρές,  $xy$  και  $z$ , τέτοια ώστε το  $xy$  να περιέχει κύκλο. Το DFA δεν είναι γνωστό, άρα δεν μπορεί να επιλεγεί με ακρίβεια το μέγεθος καθενός από τα δύο μέρη. Γνωρίζουμε όμως ότι το  $|xy|$  μπορεί να είναι το πολύ  $m$ . Θεωρούμε ότι η συμβολοσειρά  $x$  είναι το τμήμα πριν τον κύκλο,  $y$  είναι ο κύκλος, και  $z$  το τμήμα μετά τον κύκλο. (Ενδέχεται οι  $x$  και  $z$  να περιέχουν κύκλους αλλά αυτό μας είναι αδιάφορο για τη διαδικασία). Και πάλι όμως δεν γνωρίζουμε σε ποια σημεία ακριβώς πρέπει να γίνει η τμηματοποίηση. Αφού η  $y$  είναι ο κύκλος που μας ενδιαφέρει, θα πρέπει  $|y| > 0$ , γιατί σε διαφορετική περίπτωση δεν σχηματίζεται κύκλος. Επαναλαμβάνοντας τη συμβολοσειρά  $y$  για έναν τυχαίο αριθμό φορών, σχηματίζοντας δηλαδή τη  $xy^*z$ , πρέπει να λαμβάνουμε άλλες συμβολοσειρές που να ανήκουν στη γλώσσα  $L$ . Αν, παρά τις παραπάνω αβεβαιότητες, μπορούμε να δείξουμε ότι το DFA πρέπει να αποδεχτεί κάποια συμβολοσειρά που ξέρουμε ότι δεν ανήκει στη γλώσσα, τότε καταλήγουμε στο συμπέρασμα ότι η γλώσσα δεν είναι κανονική.

Για να χρησιμοποιήσουμε το παραπάνω λήμμα, πρέπει να δείξουμε ότι:

- για κάθε επιλογή του  $m$ , για κάποια συμβολοσειρά  $w \in L$  που θα επιλέξουμε (και θα επιλέξουμε μια με μήκος τουλάχιστο  $m$ ),
- για κάθε τρόπο αποσύνθεσης της  $w$  στα τμήματα  $xyz$ , αρκεί το τμήμα  $|xy|$  να μην είναι μεγαλύτερο από  $m$  και το τμήμα  $y$  να μην είναι η κενή συμβολοσειρά  $\epsilon$ ,

μπορούμε να επιλέξουμε ένα  $i$  τέτοιο ώστε η  $xy^iz$  να μην ανήκει στη γλώσσα  $L$ . Η παραπάνω διαδικασία μπορεί να θεωρηθεί σαν ένα παιχνίδι στο οποίο ο αντίπαλός μας κάνει τις κινήσεις 1 και 3 (και επιλέγει το  $m$  και το  $xyz$ ) κι εμείς κάνουμε τις κινήσεις 2 και 4 (και επιλέγουμε τη συμβολοσειρά  $w$  και το  $i$ ).

**Αναλυτικότερα:** ο αντίπαλος διαλέγει το  $m$  κι εμείς πρέπει να είμαστε έτοιμοι να χειριστούμε οποιοδήποτε  $m$ . Στη συνέχεια επιλέγουμε μια συμβολοσειρά  $w \in L$  με  $|w| \geq m$ , με τέτοιον τρόπο ώστε η συμβολοσειρά  $w$  να εξαρτάται από το  $m$ . Ο αντίπαλος επιλέγει τη μορφή της διάσπασης της  $w$  σε  $xyz$ , τηρώντας τις συνθήκες  $|xy| \leq m$  και  $|y| \geq 1$ . Τέλος, εμείς επιλέγουμε το κατάλληλο  $i$  έτσι ώστε  $xy^iz \notin L$ , δείχνοντας κατά συνέπεια τη μη κανονικότητα της γλώσσας  $L$ . Στόχος είναι να δείξουμε ότι μπορούμε πάντα να κερδίζουμε τον αντίπαλο. Αν δείξουμε αυτό, έχουμε ουσιαστικά αποδείξει ότι η γλώσσα  $L$  δεν είναι κανονική.

**Παράδειγμα 2.5.4.** Θεωρούμε τη γλώσσα  $B = \{0^n 1^n : n \geq 0\}$ . Υποθέτουμε ότι η  $B$  είναι κανονική. Τότε η  $B$  θα αναγνωριζόταν από κάποιο DFA με  $p$  καταστάσεις. Επιλέγουμε μια συγκεκριμένη συμβολοσειρά  $w = 0^p 1^p$ , η οποία ανήκει στη γλώσσα  $B$ . Υπάρχει η διάσπαση  $w = xyz$  με βάση το Pumping Lemma. Τότε, επειδή  $|xy| \leq p$ , καταλαβαίνουμε ότι το τμήμα  $y$  περιέχει μόνο 0 (τουλάχιστον ένα 0). Τότε όμως η  $xy^0z = xz$  δεν ανήκει στη γλώσσα  $B$  (αφού περιέχει λιγότερα 0 από 1). Άρα εφαρμόζοντας το Pumping Lemma καταλήγουμε σε άτοπο έχοντας κάνει την υπόθεση ότι η γλώσσα είναι κανονική. Επομένως, δεν είναι κανονική.

Παρατηρήστε ότι αυτό που πρέπει να κάνουμε είναι να βρούμε μία συμβολοσειρά  $w$  που δεν δίνει συμβολοσειρές της γλώσσας, αλλά πρέπει να δείξουμε ότι υπάρχει πρόβλημα για οποιαδήποτε διάσπαση κι αν επιλεγεί.

**Παράδειγμα 2.5.5.** Παλίνδρομη είναι μια λέξη που είναι η ίδια είτε διαβαστεί από την αρχή στο τέλος είτε από το τέλος στην αρχή (π.χ., 'ΑΛΛΑ'. Έστω  $P$  το σύνολο όλων των παλίνδρομων λέξεων στο αλφάβητο  $\{a, b\}$ . Αυτή η γλώσσα δεν είναι κανονική. Ακολουθεί η απόδειξη με χρήση του Pumping Lemma. Υποθέτουμε ότι η γλώσσα  $P$  είναι κανονική. Τότε θα αναγνωρίζεται από κάποιο DFA με  $p$  καταστάσεις. Επιλέγουμε τη συμβολοσειρά  $w = a^p b a^p$ . Υπάρχει η διάσπαση  $w = xyz$  με βάση το Pumping Lemma. Τότε, επειδή  $|xy| \leq p$ , καταλαβαίνουμε ότι το τμήμα  $y$  περιέχει μόνο  $a$  (τουλάχιστον ένα  $a$ ). Τότε όμως η  $xy^0z = xz$ , η  $xy^2z = xygz$ , κτλ δεν ανήκει στη γλώσσα  $P$  (αφού το δεύτερο τμήμα των  $a$  παραμένει σταθερό περιέχοντας  $p$   $a$  ενώ το πρώτο τμήμα των  $a$  περιέχει λιγότερα ή περισσότερα  $a$  από  $p$ , αντίστοιχα). Άρα εφαρμόζοντας το Pumping Lemma καταλήγουμε σε άτοπο έχοντας κάνει την υπόθεση ότι η γλώσσα είναι κανονική. Επομένως, δεν είναι κανονική.

### Παραδείγματα εφαρμογής του Pumping Lemma

**Παράδειγμα 2.5.6.** Αποδείξτε ότι η γλώσσα  $L = \{a^n b^n : n \neq 0\}$  δεν είναι κανονική.

Απόδειξη. Δεν γνωρίζουμε το  $m$ , αλλά υποθέτουμε ότι υπάρχει κάποιο. Επιλέγουμε μια συμβολοσειρά  $w = a^n b^n$  με  $n > m$ , έτσι ώστε κάθε πρόθεμα με μήκος  $m$  να αποτελείται εξ ολοκλήρου από  $a$ . Δεν γνωρίζουμε πώς να επιλέξουμε τα τμήματα  $xyz$  της  $w$ , αλλά αφού  $|xy| \leq m$ , το τμήμα  $xy$  πρέπει να περιέχει μόνο  $a$ . Επιπλέον, το τμήμα  $y$  δεν μπορεί να είναι η κενή συμβολοσειρά. Επιλέγουμε  $i = 0$ . Αυτό έχει σαν αποτέλεσμα την απομάχρυνση  $|y|$  "α" από τη συμβολοσειρά χωρίς να επηρεάζεται ο αριθμός των "b". Η συμβολοσειρά που προκύπτει έχει λιγότερα "a" από "b", κι επομένως δεν ανήκει στη γλώσσα  $L$ . Άρα η  $L$  δεν είναι κανονική.  $\square$

**Παράδειγμα 2.5.7.** Αποδείξτε ότι η γλώσσα  $L = \{a^n b^k : n > k \text{ και } n \neq 0\}$  δεν είναι κανονική.

Απόδειξη. Δεν γνωρίζουμε το  $m$ , αλλά υποθέτουμε ότι υπάρχει κάποιο. Επιλέγουμε μια συμβολοσειρά  $w = a^n b^k$  με  $n > m$ , έτσι ώστε κάθε πρόθεμα με μήκος  $m$  να αποτελείται εξ ολοκλήρου από  $a$  και  $k = n - 1$ , έτσι ώστε να υπάρχει ένα περισσότερο  $a$  από ό,τι  $b$ . Δεν γνωρίζουμε πώς να επιλέξουμε τα τμήματα  $xyz$  της  $w$ , αλλά αφού  $|xy| \neq m$ , το τμήμα  $xy$  πρέπει να περιέχει μόνο  $a$ . Επιπλέον, το τμήμα  $y$  δεν μπορεί να είναι η κενή συμβολοσειρά. Επιλέγουμε  $i = 0$ . Αυτό έχει σαν αποτέλεσμα την απομάχρυνση  $|y|$  "a" από τη συμβολοσειρά χωρίς να επηρεάζεται ο αριθμός των "b". Η συμβολοσειρά που προκύπτει έχει λιγότερα  $a$  από την προηγούμενη οπότε είτε έχει λιγότερα  $a$  από ό,τι  $b$  ή ίδιο αριθμό από  $a$  και  $b$ , κι επομένως ούτως ή άλλως δεν ανήκει στη γλώσσα  $L$ . Άρα η  $L$  δεν είναι κανονική.  $\square$

**Παράδειγμα 2.5.8.** Αποδείξτε ότι η γλώσσα  $L = \{a^n : n \text{ πρώτος αριθμός}\}$  δεν είναι κανονική.

*Απόδειξη.* Δεν γνωρίζουμε το  $m$ , αλλά υποθέτουμε ότι υπάρχει κάποιο. Επιλέγουμε μια συμβολοσειρά  $w = a^n$  όπου  $n$  είναι πρώτος αριθμός και  $|xyz| = n > m + 1$ . Αυτό μπορεί πάντα να γίνει γιατί δεν υπάρχει κάποιος μέγιστος πρώτος αριθμός. Τότε κάθε πρόθεμα της συμβολοσειράς  $w$  με μήκος  $m$  να αποτελείται εξ ολοκλήρου από  $a$ . Δεν γνωρίζουμε πώς να επιλέξουμε τα τμήματα  $xyz$  της  $w$ , αλλά αφού  $|xy| \leq m$ , για το τμήμα  $z$  είναι:  $|z| > 1$ . Επιπλέον, το τμήμα  $y$  δεν μπορεί να είναι η κενή συμβολοσειρά. Αφού είναι  $|z| > 1$ , θα είναι και  $|xz| > 1$ . Επιλέγουμε  $i = |xz|$ . Τότε  $|xyiz| = |xz| + |y||xz| = (1 + |y|)|xz|$ . Αφού  $(1 + |y|)$  και  $|xz|$  είναι μεγαλύτεροι του 1, το γινόμενό τους θα είναι σύνθετος αριθμός, δηλαδή όχι πρώτος. Επομένως ο  $|xyiz|$  δεν είναι πρώτος αριθμός.  $\square$

**Παράδειγμα 2.5.9.** Αποδείξτε ότι η γλώσσα  $L = \{1^{n^2} : n \geq 0\}$  δεν είναι κανονική.

*Απόδειξη.* Υποθέτουμε ότι η γλώσσα  $L$  είναι κανονική και καλούμε  $p$  τη σταθερά που ορίζεται από το Pumping Lemma. Επιλέγουμε μια συμβολοσειρά  $s = 1^{p^2}$  που ανήκει στη γλώσσα. Αφού η  $s$  ανήκει στη γλώσσα και έχει μήκος τουλάχιστο  $p$  το Pumping Lemma εγγυάται ότι η  $s$  μπορεί να χωριστεί σε τρεις υποσυμβολοσειρές  $x, y, z$ , έτσι ώστε  $s = xyz$  και για κάθε  $i \geq 0$  η συμβολοσειρά  $xy^iz$  να ανήκει επίσης στη γλώσσα  $L$ . Θεωρούμε τώρα τις συμβολοσειρές  $xy^iz$  και  $xy^{i+1}z$ . Αυτές διαφέρουν μεταξύ τους κατά μια επανάληψη της υποσυμβολοσειράς  $y$  κι επομένως τα μήκη τους διαφέρουν κατά το μήκος της  $|y|$ . Αν επιλέξουμε ένα πολύ μεγάλο  $i$ , τα μήκη των  $xy^iz$  και  $xy^{i+1}z$  δεν μπορεί να είναι και τα δύο τέλεια τετράγωνα μιας και είναι πολύ κοντά το ένα στο άλλο. Επομένως δεν είναι δυνατόν και η  $xy^iz$  και η  $xy^{i+1}z$  να ανήκουν στη γλώσσα  $L$ . Στη συνέχεια υπολογίζουμε μια τιμή του  $i$  που οδηγεί σε άτοπο. Αν  $m = n^2$  είναι τέλειο τετράγωνο η διαφορά του με το επόμενο τέλειο τετράγωνο  $(n + 1)^2$  είναι:

$$\begin{aligned} (n + 1)^2 - n^2 &= n^2 + 2n + 1 - n^2 \\ &= 2n + 1 \\ &= 2\sqrt{m} + 1. \end{aligned}$$

Το Pumping Lemma δηλώνει ότι και το  $|xy^iz|$  και το  $|xy^{i+1}z|$  είναι τέλεια τετράγωνα για κάθε  $i$ . Αλλά όταν  $|xy^iz|$  είναι  $m$ , όπως παραπάνω, βλέπουμε ότι δεν μπορεί και τα δύο να είναι τέλεια τετράγωνα αν  $|y| < 2\sqrt{|xy^iz|} + 1$  γιατί τότε θα ήταν πολύ κοντά μεταξύ τους. Παρατηρούμε ότι  $|y| \leq |s| = p^2$ . Έστω  $i = p^4$ . Τότε:

$$\begin{aligned} |y| \leq p^2 &= \sqrt{p^4} \\ &< 2\sqrt{p^4} + 1 \\ &\leq 2\sqrt{|xy^iz|} + 1. \end{aligned}$$

$\square$

**Παράδειγμα 2.5.10.** Είναι το σύνολο  $\{0^{2n} | n \geq 1\}$  κανονικό; Αποδείξτε την απάντησή σας.

Απόδειξη. Είναι ένα κανονικό σύνολο αφού μπορεί να παραχθεί από την κανονική έκφραση  $00(00)^*$ .  $\square$

**Παράδειγμα 2.5.11.** Είναι το σύνολο  $\{0^m 1 n 0^{m+n} | m \geq 1 \text{ και } n \geq 1\}$  κανονικό; Αποδείξτε την απάντησή σας.

Απόδειξη. Δεν είναι κανονικό σύνολο. Για να το αποδείξουμε υποθέτουμε ότι η γλώσσα είναι κανονική και έστω  $N$  η σταθερά που χρησιμοποιείται από το Pumping Lemma γι' αυτή τη γλώσσα. Θεωρούμε τη συμβολοσειρά

$$z = 0^N 1^N 0^{2N}.$$

Προφανώς η συμβολοσειρά  $z$  ανήκει στην εν λόγω γλώσσα και ικανοποιεί τις απαιτήσεις (σχετικά με το μέγεθος) που ορίζονται από το Pumping Lemma. Αυτό σημαίνει ότι υπάρχουν υποσυμβολοσειρές  $u, v, w$  τέτοιες ώστε  $z = uvw$ , με  $v \neq \epsilon$ ,  $|uv| \leq N$  και  $uv^i w$  να ανήκει στη γλώσσα για κάθε  $i \geq 0$ . Αφού  $|uv| \leq N$  τα τμήματα  $u$  και  $v$  πρέπει να καλύπτουν τους  $N$  πρώτους χαρακτήρες της  $z$  γεγονός που σημαίνει ότι θα πρέπει να αποτελούνται εξ ολοκλήρου από "0". Αφού  $v \neq \epsilon$ , το  $v$  περιέχει τουλάχιστο ένα "0". Έστω ότι το  $v$  αποτελείται από  $k$  "0" (δηλαδή  $v = 0^k$ ). Τότε αν θεωρήσουμε  $uv^0 w = uw$  έχουμε αφαιρέσει  $k$  "0" από το αρχικό τμήμα της αρχικής συμβολοσειράς  $z$ , δηλαδή

$$uv^0 w = uw = 0^{N-k} 1^N 0^{2N}.$$

Με βάση το Pumping Lemma η συμβολοσειρά  $uw$  ανήκει στη γλώσσα που μελετάμε. Αλλά αυτό είναι άτοπο αφού εξ ορισμού οι συμβολοσειρές της γλώσσας πρέπει να έχουν τη μορφή  $0^m 1^n 0^{m+n}$ , δηλαδή το μήκος της πρώτης ακολουθίας "0" και "1" θα πρέπει να είναι ίσο με το μήκος της δεύτερης ακολουθίας "0". Αυτό δεν είναι αληθές για τη συμβολοσειρά  $0^{N-k} 1^N 0^{2N}$ .  $\square$

**Παράδειγμα 2.5.12.** Είναι το σύνολο όλων των συμβολοσειρών του αλφαβήτου  $\{0, 1\}$  που δεν περιέχουν τρία συνεχόμενα "0" κανονικό; Αποδείξτε την απάντησή σας.

Απόδειξη. Η γλώσσα είναι κανονική. Θεωρούμε τη γλώσσα  $L$  που αποτελείται από όλες τις συμβολοσειρές που ορίζονται στο αλφάβητο  $\{0, 1\}$  και περιέχουν τρία συνεχόμενα "0". Η  $L$  είναι κανονική αφού παράγεται από την κανονική έκφραση  $(0 \cup 1)^* 000 (0 \cup 1)^*$ . Το σύνολο των συμβολοσειρών του αλφαβήτου  $\{0, 1\}$  που δεν περιέχουν τρία συνεχόμενα "0" είναι το συμπλήρωμα της γλώσσας  $L$ . Αφού η  $L$  είναι κανονική και οι κανονικές γλώσσες είναι κλειστές ως προς τη συμπλήρωση, συμπεραίνουμε ότι και η δοσμένη γλώσσα είναι κανονική.  $\square$

**Παράδειγμα 2.5.13.** Είναι το σύνολο όλων των συμβολοσειρών του αλφαβήτου  $\{0, 1\}$  με ίδιο αριθμό "0" και "1" κανονικό; Αποδείξτε την απάντησή σας.

Απόδειξη. Η γλώσσα δεν είναι κανονική. Για να το αποδείξουμε καλούμε τη δοσμένη γλώσσα  $L$  και θεωρούμε τη γλώσσα

$$L' = L \cap 0^*1^*.$$

Έστω μια τυχαία συμβολοσειρά  $w \in L$ . Αφού η συμβολοσειρά  $w$  ανήκει στη  $L$ , θα πρέπει να περιέχει ίδιο αριθμό “0” και “1”. Επιπλέον, επειδή η  $w$  πρέπει να ανήκει στη  $0^*1^*$ , όλα τα “0” θα πρέπει να προηγούνται των “1”. Κατά συνέπεια η συμβολοσειρά  $w$  θα πρέπει να είναι της μορφής  $0^n1^n$  για κάποιο  $n \geq 0$ . Ή διαφορετικά

$$L' = \{0^n1^n | n \geq 0\}.$$

Όμως η γλώσσα  $L'$  δεν είναι κανονική. Η γλώσσα  $0^*1^*$  είναι κανονική εξ' ορισμού άρα η γλώσσα  $L$  δεν είναι κανονική.

Μια εναλλακτική μέθοδος απόδειξης θα ήταν η χρησιμοποίηση του Pumping Lemma. Υποθέτουμε ότι η γλώσσα  $L$  είναι κανονική και έστω  $N$  η σταθερά του Pumping Lemma. Επιλέγουμε  $z = 0^N1^N$ . Η συμβολοσειρά  $z$  ικανοποιεί τις συνθήκες του Pumping Lemma. Προχωράμε όπως στην απόδειξη της μη κανονικότητας της γλώσσας  $\{0^n1^n | n \geq 0\}$ .  $\square$

### Σημεία για ιδιαίτερη προσοχή

#### 1. Επιλογή μιας συμβολοσειράς $w$ που δεν ανήκει στη γλώσσα $L$ .

Έστω για παράδειγμα η γλώσσα  $L = \{ww : w \in (a+b)^*\}$ . Η επιλογή  $w = a^nb^n$  δεν είναι σωστή αφού  $a^nb^n \in L$ . Το άτοπο σε αυτή την περίπτωση προκύπτει αν επιλέξουμε  $i = 1$  οπότε  $w = xy^iz \in L$ .

#### 2. Παράλειψη συνδυασμών των $x, y, z$ για την αποσύνθεση της συμβολοσειράς $w$ .

Θεωρούμε ξανά σαν παράδειγμα τη γλώσσα  $L = \{ww : w \in (a+b)^*\}$ . Έστω ότι ο αντίπαλος επιλέγει κάποιο  $m$  κι εμείς επιλέγουμε  $w = a^{2m}$ . Στη συνέχεια, ο αντίπαλος θα επιλέξει κάποια  $x, y, z$  ώστε  $w = xyz$ . Ενδέχεται, αν δεν λάβουμε υπόψη μας όλα τα πιθανά σενάρια για τα  $x, y, z$ , να εξετάσουμε μόνο την περίπτωση που  $x = \epsilon, y = a, z = a^{2m-1}$ . Τότε όμως για  $i = 0$ , είναι  $xy^iz = a^{2m-1} \in L$ , δηλαδή φαίνεται ότι καταλήγουμε σε άτοπο. Αν όμως ο αντίπαλος επιλέξει  $x = \epsilon, y = aa, z = a^{2m-2}$ , τότε για κάθε  $i \geq 0$  είναι  $xy^iz \in L$ .

#### 3. Επιλογή μιας συμβολοσειράς $w$ που δεν είναι επαρκώς καθορισμένη

Ο αντίπαλος προσπαθεί να μας νικήσει επιλέγοντας μια κακή “διάσπαση”,  $xyz$ , της συμβολοσειράς  $w$  που εμείς έχουμε επιλέξει. Η  $w$  μπορεί να είναι κάθε συμβολοσειρά της γλώσσας  $L$  με μήκος το πολύ  $m$ , αλλά όσο πιο συγκεκριμένη είναι η επιλογή μας τόσο πιο πολύ δυσκολεύουμε τον αντίπαλο.

Θεωρούμε για παράδειγμα ξανά την παραπάνω γλώσσα  $L = \{ww : w \in (a+b)^*\}$ . Αν επιλέξουμε μια  $w = aa$ , με  $a$  οποιαδήποτε συμβολοσειρά μήκους το πολύ  $m$ , τότε ο αντίπαλος

επιλέγει  $w = xyz = aa$ . Για  $i = 0$ , είναι  $xz \neq aa$ , δηλαδή φαίνεται ότι έχουμε καταλήξει σε άτοπο, αλλά ουσιαστικά έχουμε κάνει λάθος. Δεν αρκεί να δείξουμε ότι είναι  $xy^iz \neq aa$  για κάποιο συγκεκριμένο  $a$  αλλά για κάθε ενδεχόμενη επιλογή του  $a$ .

Δηλαδή, η επιλογή της  $w$  δεν πρέπει να είναι γενική: αν διαλέγαμε  $w = a^n b^n$ , ο αντίπαλος θα επέλεγε  $x = \epsilon$ ,  $y = aa$ ,  $z = a^{2n-2}$ , οπότε για οποιοδήποτε  $i$  θα συμπεραίναμε ότι  $xy^iz \in L$  και δεν θα υπήρχε περίπτωση να νικήσουμε.

4. **Επιλογή συμβολοσειράς  $w$  που δεν εξαρτάται από το  $m$ .**

Θεωρούμε πάλι τη γλώσσα  $L = \{ww : w \in (a+b)^*\}$  και έστω ότι επιλέγουμε  $w = abab \neq f(m)$ . Επειδή δεν ξέρουμε την τιμή του  $m$ , αν το μήκος της συμβολοσειράς που επιλέξαμε δεν είναι συνάρτηση του  $m$ , θα έχουμε πρόβλημα μιας και θα πρέπει να είμαστε σε θέση να αντιμετωπίσουμε οποιαδήποτε επιλογή του  $m$  από τον αντίπαλο.

5. **Επιλογή αρνητικού ή κλασματικού  $i$ .**

Αυτό δεν επιτρέπεται με βάση τις υποθέσεις του Pumping Lemma.

6. **Εφαρμογή του Pumping Lemma για κανονική γλώσσα.** Έστω η γλώσσα  $L = 0^x 1^y : x + y \Rightarrow 0 \pmod{4}$  που είναι κανονική. Έστω ότι θέλουμε να δείξουμε ότι δεν είναι κανονική με χρήση του Pumping Lemma και για το σκοπό αυτό επιλέγουμε  $w = 0^{4n+3}1$ , ενώ ο αντίπαλος επιλέγει  $w = xyz$ , με  $x = 0^a$ ,  $y = 0^b$ ,  $z = 0^c 1$ , με  $a + b + c = 4n + 3$ . Στην περίπτωση αυτή θα μπορούσε κάποιος να πει ότι αν επιλέξουμε  $i$  τέτοιο ώστε  $xy^iz = 0^{4n+3+ib}1$ , τότε για κάθε  $b : x + y = 4n + 3 + ib + 1$  που δεν είναι πολλαπλάσιο του 4. Ο ισχυρισμός αυτός όμως είναι λάθος αφού για  $b = 4$ , η ποσότητα  $4n + 3 + ib + 1$  είναι πολλαπλάσιο του 4 για κάθε  $i$ .

7. **Υπόθεση ότι κάθε συμβολοσειρά μιας κανονικής γλώσσας  $L$  μπορεί να γραφεί σαν  $xy^iz$  για κάποιο  $i \geq 2$ .** Αυτός ο ισχυρισμός δεν είναι απαραίτητα σωστός. Ας θεωρήσουμε τη γλώσσα  $L = (0 + 1 + 2)^*$ . Τότε μπορούμε να πούμε ότι για  $i \geq 2$ , υπάρχουν  $x, y, z$  έτσι ώστε συμβολοσειρές της γλώσσας να μπορούν να γραφούν με τη μορφή  $xy^iz$ . Αυτό όμως είναι λάθος αφού υπάρχουν συμβολοσειρές της  $L$  που δεν περιέχουν υποσυμβολοσειρές του τύπου  $yy$  (αυτό αποδείχθηκε αρχικά από το Νορβηγό μαθηματικό Axel Thue στις αρχές του 1900).

8. **Προσπάθεια χρησιμοποίησης του Pumping Lemma για την απόδειξη της κανονικότητας μιας γλώσσας.** Το Pumping Lemma βασίζεται σε μια ιδιότητα των κανονικών γλωσσών και αυτό που λέει είναι “Αν η γλώσσα  $L$  είναι κανονική, τότε η  $L$  έχει την παραπάνω ιδιότητα”. Επομένως το Pumping Lemma δεν μπορεί να χρησιμοποιηθεί για την απόδειξη της κανονικότητας μιας γλώσσας, παρά μόνο με αντιθετοαντιστροφή, δηλ. “αν μια γλώσσα  $L$  δεν



έχει τη συγκεκριμένη ιδιότητα, τότε η  $L$  δεν είναι κανονική”. Εν τούτοις, υπάρχουν γλώσσες που αν και δεν είναι κανονικές ικανοποιούν τις προϋποθέσεις του Pumping Lemma.

Παραθέτουμε ένα σχετικό παράδειγμα:

**Παράδειγμα 2.5.14.** Έστω η γλώσσα  $L = \{a^i b^j c^k : i = 0 \text{ ή } j = k\}$ , κι έστω  $w \in L$  η συμβολοσειρά που επιλέγουμε για την εφαρμογή του Pumping Lemma. Διακρίνουμε 2 περιπτώσεις:

1.  $w = b^j c^k$  για κάποιους ακέραιους  $j, k$ . Διαλέγουμε  $m = 1$ , οπότε μπορούμε να υποθέσουμε ότι είτε  $j \geq 1$  είτε  $k \geq 1$ . Τότε είναι  $w = xyz$ , για  $x = \epsilon$ ,  $y = b$  (αν  $j \geq 1$ ) ή  $y = c$  (αν  $k \geq 1$ ) και  $z =$  με το υπόλοιπο τμήμα της συμβολοσειράς  $w$  και κατά συνέπεια  $xy^i z \in L$  για κάθε  $i \geq 0$ .
2.  $w = a^i b^j c^k$  για κάποιους ακέραιους  $j, k$ , με  $i \geq 1$ . Διαλέγουμε  $m = 1$ . Τότε είναι  $w = xyz$ , για  $x = \epsilon$ ,  $y = a$  και  $z =$  με το υπόλοιπο τμήμα της συμβολοσειράς  $w$  και κατά συνέπεια  $xy^i z \in L$  για κάθε  $i \geq 0$ .

Το συμπέρασμα από το παραπάνω παράδειγμα είναι ότι το (κανονικό) Pumping Lemma δεν είναι τόσο ισχυρό ώστε να μπορεί άμεσα να αποδείξει τη μη κανονικότητα συγκεκριμένων μη κανονικών γλωσσών.

**Για εξάσκηση...** Ορίζουμε σαν  $E$  τη γλώσσα που περιέχει όλες τις δυαδικές συμβολοσειρές που έχουν ίδιο αριθμό από 0 και 1.

1. Δείξτε ότι η  $E$  δεν είναι κανονική βρίσκοντας άπειρο σύνολο από pairwise distinguishable συμβολοσειρές.
2. Δείξτε ότι η  $E$  δεν είναι κανονική δείχνοντας ότι παραβιάζει το Pumping Lemma.

## 2.6 Εφαρμογές πεπερασμένων αυτομάτων

Είδαμε τη θεωρητική πλευρά των πεπερασμένων αυτομάτων. Βέβαια τόσο αυτά όσο και οι ιδέες τους είναι εκληπτικά κοινές στην επιστήμη των υπολογιστών. Στη συνέχεια, παρουσιάζουμε μερικές από τις πιο γνωστές εφαρμογές του.

### 2.6.1 Επεξεργασία συμβολοσειρών (String Processing)

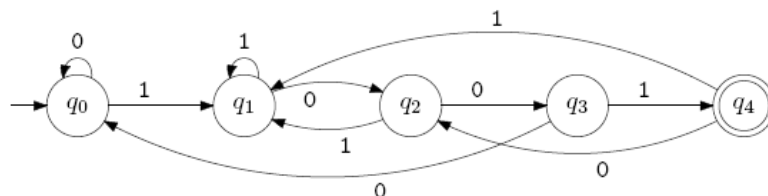
Σεφτείτε προβλήματα που περιλαμβάνουν συμβολοσειρές. Μια συνηθισμένη εργασία είναι η εύρεση λολων των εμφανίσεων μικρών συμβολοσειρών, που λέγονται πρότυπα, μέσα σε μεγαλύτερες, που λέγονται κείμενο. Για παράδειγμα, αν το πρότυπο είναι *ana* και το κείμενο είναι το *bananarama's ana*, υπάρχουν 3 εμφανίσεις του προτύπου. Πέρα από τη χρήση από επεξεργαστές κειμένου,

τέτοιου είδους ερωτήσεις απατώνται και στη βιολογία: η συμβολοσειρά κειμένου είναι η ακολουθία DNA του οργανισμού που έχει κωδικοποιηθεί σε συμβολοσειρά και το πρότυπο είναι κάποια συγκεκριμένη ακολουθία DNA που μάς ενδιαφέρει.

Ένας απλοϊκός αλγόριθμος θεωρεί κάθε πιθανό γράμμα του κειμένου σαν αρχικό του προτύπου, και απαιτεί χρόνο ανάλογο του γινομένου των δύο μηκών, με πλεονέκτημα φυσικά την απλότητα.

Μια απλή βελτίωση θα ήταν να χρησιμοποιήσουμε ένα FA για την επεξεργασία του κειμένου: συγκεκριμένα, το DFA που αναγνωρίζει όλες τις συμβολοσειρές που καταλήγουν στο συγκεκριμένο πρότυπο. Αυτή η επιλογή επιτρέπει να εξετάσουμε κάθε σύμβολο του κειμένου μία μόνο φορά γεγονός που αποτελεί τη βάση του αλγορίθμου των Knuth - Morris - Pratt οι οποίοι έδειξαν επιπλέον πώς να κατασκευάζεται γρήγορα το DFA. Το τελικό αποτέλεσμα είναι ένας αλγόριθμος που τρέχει σε χρόνο ανάλογο του μήκους του κειμένου.

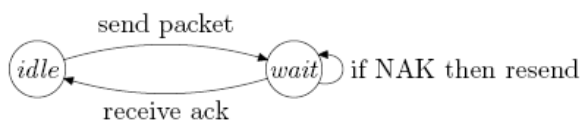
**Παράδειγμα 2.6.1.** Ας υποθέσουμε ότι αναζητούμε όλες τις εμφανίσεις του προτύπου 1001 σε ένα κείμενο. Θα πρέπει να κατασκευάσουμε το DFA που να αναγνωρίζει όλες τις συμβολοσειρές που καταλήγουν σε 1001 και στη συνέχεια να του δώσουμε το κείμενο σαν είσοδο σύμβολο - σύμβολο. Κάθε φορά που φτάνουμε στην τελική κατάσταση  $q_4$ , τυπώνεται η τρέχουσα θέση στο κείμενο.



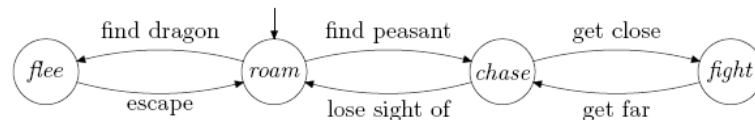
## 2.6.2 Μηχανές πεπερασμένων καταστάσεων (Finite - State Machines)

Μηχανές πεπερασμένων καταστάσεων χρησιμοποιούνται στα δίκτυα. Για παράδειγμα, μια συσκευή για επικοινωνία συχνά σχεδιάζεται ώστε να μπορεί να βρίσκεται σε έναν σταθερό αριθμό καταστάσεων και να ανταποκρίνεται σε καθορισμένο αριθμό γεγονότων. Για κάθε τέτοιο γεγονός, το πρόγραμμα της συσκευής τής υποδεικνύει ποια ενέργεια πρέπει να πραγματοποιήσει και σε ποια κατάσταση να μεταβεί. Επομένως, γεγονότα όπως 'receive message' ή 'timer expires' εμφανίζονται στη θέση των συμβόλων στις μεταβάσεις.

Το αποτέλεσμα λέγεται συνήθως μηχανή πεπερασμένων καταστάσεων: ένα FA μαζί με ενέργειες σαν επιγραφές των μεταβάσεων. Παράκάτω δείτε ένα απλό παράδειγμα για μια σύνδεση επικοινωνίας:



Μηχανές πεπερασμένων καταστάσεων χρησιμοποιούνται επίσης για τον καθορισμό της συμπεριφοράς χαρακτήρων ενός παιχνιδιού που παράγονται από υπολογιστές, και λέγονται bots. Παράγονται στιγμιότυπα όπως το παρακάτω:

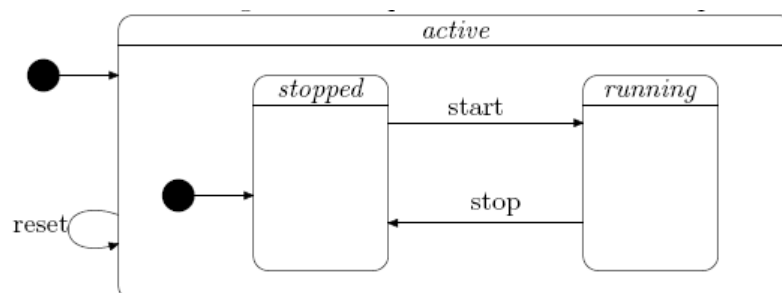


Αν και η προσέγγιση με πεπερασμένες καταστάσεις ακούγεται απλοϊκή, και σίγουρα περιορίζει το εύρος των συμπεριφορών ενός τέτοιου χαρακτήρα, το πλεονέκτημα είναι η ευκολία χρήσης, αφού μπορεί να χρησιμοποιηθεί μια συγκεκριμένη γλώσσα για τον καθορισμό του διαγράμματος και αυτοματοποιημένα εργαλεία για την παραγωγή του κώδικα.

### 2.6.3 Διαγράμματα καταστάσεων

Πολλές εργασίες μπορούν να αποδομηθούν σε σύνολα καταστάσεων και ενεργειών που πρέπει να πραγματοποιηθούν. Για παράδειγμα, η κλήση μέσω ενός σταθερού τηλεφώνου όταν ακούμε το χαρακτηριστικό ήχο οπότε και προχωράμε στο σχηματισμού του αριθμού. Τα διαγράμματα καταστάσεων παρέχουν έναν τρόπο για τον καθορισμό της λειτουργίας τέτοιων διεργασιών. Εισήχθησαν από τον Harel το 1987 και αποτελούν σήμερα μέρος της γλώσσας μοντελοποίησης UML. Τα διαγράμματα καταστάσεων επεκτείνουν τα διαγράμματα των πεπερασμένων αυτομάτων συμπεριλαμβάνοντας δυνατότητες όπως ταυτόχρονη εκτέλεση, απλοκλειστική διάζευξη, και ιεραρχικές δομές με επανεπεξεργασία και υπερκαταστάσεις.

**Παράδειγμα 2.6.2.** Παρακάτω δείτε ένα απλουστευμένο διάγραμμα καταστάσεων για την παρακολούθηση τερματισμών.



Τα διαγράμματα καταστάσεων εκτός από ότι διασαφηνίζουν το σχεδιασμό, είναι χρήσιμα για την παραγωγή κώδικα αφού υπάρχουν προγράμματα λογισμικού που μετατρέπουν αυτόματα τα διαγράμματα καταστάσεων σε λειτουργικά τμήματα κώδικα.

#### 2.6.4 Λεκτική ανάλυση (Lexical Analysis)

Το πρώτο βήμα για τη μεταγλώττιση ενός προγράμματος είναι η λεκτική ανάλυση. Κατά τη διάρκειά της απομονώνονται λέξεις κλειδιά, περοσδιοριστικά, τελεστές κτλ., και εκκαθαρίζονται σύμβολα όπως σχόλια που δεν είναι χρήσιμα για περαιτέρω επεξεργασία. Η είσοδος σε ενά λεκτικό αναλυτή είναι επομένως ο πηγαίος κώδικας με τη μορφή συμβολοσειράς και η έξοδος είναι μια ακολουθία μονάδων που λέγονται tokens.

Ένα token είναι μια κατηγορία, για παράδειγμα ‘προσδιοριστικό’ ή ‘τελεστής πράξης’. Οι λέξεις κλειδιά αποτελούν συνήθως διαφορετικά tokens.

**Παράδειγμα 2.6.3.** Το κομμάτι κώδικα μπορεί να δώσει την ακολουθία token

```
for i = 1 to max do
  x[i] = 0;
```

`for` `id` `=` `num` `to` `id` `do` `id` `[` `id` `]` `=` `num` `sep`

Μια γλώσσα προγραμματισμού δεν μπορεί φυσικά να περιγραφεί από μια κανονική έκφραση, αλλά υπάρχουν τμήματά της που μπορούν να περιγραφούν από κανονικές εκφράσεις (π.χ., αριθμητικές σταθερές), και αποτελούν tokens. Για παράδειγμα, η δεσμευμένη λέξη `then` και ονόματα μεταβλητών μπορούν να περιγραφούν από τις παρακάτω εκτεταμένες κανονικές εκφράσεις:

<i>token</i>	<i>RE</i>
<code>then</code>	<code>then</code>
variable name	<code>[a-zA-Z][a-zA-Z0-9]*</code>

όπου η κανονική έκφραση για το όνομα της μεταβλητής ορίζει ότι πρόκειται για οποιαδήποτε συμβολοσειρά αλφαριθμητικών χαρακτήρων που ξεκινάει με κάποιο γράμμα.

Ο λεκτικός αναλυτής αντικαθιστά κάθε token με μια απλή τιμή διαβάζοντας τη συμβολοσειρά του πηγαίου κώδικα σύμβολο - σύμβολο, αν και στην πράξη θα πρέπει να διαβάζει και επόμενους χαρακτήρες για να προσδιορίσει που τελειώνει το τρέχον token. Αν διαβάσει `then` δεν είναι προφανές μέχρι να διαβαστεί το επόμενο σύμβολο αν πρόκειται για τη δεσμευμένη λέξη ή για κάποιο προσδιοριστικό π.χ., `thence`.

Μετά τον προσδιορισμό κάθε token, υπάρχουν ενδεχομένως συγκεκριμένες ενέργειες που πρέπει να γίνουν. Για παράδειγμα, όταν προσδιορίζει έναν αριθμό, ο λεκτικός αναλυτής θα πρέπει να υπολογίσει και την τιμή του αριθμού. Ένας τρόπος για να γίνει αυτό είναι η τροποποίηση του FA ώστε να υπολογίζει παράλληλα και τις τιμές αυτές. Ο λεκτικός αναλυτής συγκρατεί επίσης τις δηλωμένες μεταβλητές μέσω του πίνακα συμβόλων. Για παράδειγμα, σε γλώσσες όπως η Java και η C, κάθε μεταβλητή πρέπει να δηλωθεί πριν χρησιμοποιηθεί.

Υπάρχει ένα βοηθητικό πρόγραμμα που διανέμεται με το UNIX που λέγεται Lex και παράγει κώδικα σε C για λεκτικό αναλυτή. Ο χρήστης δίνει τους ορισμούς των tokens και τις ενέργειες που προκαλεί το καθένα. Η προσέγγιση αυτή έχει πλεονεκτήματα όπως το ότι το αρχείο εισόδου στο Lex μπορεί εύκολα να τροποποιηθεί ενώ η ορθότητα της εισόδου στο Lex μπορεί να διαπιστωθεί σαφώς ευκολότερα από το να κατασκευάσει κανείς λεκτικό αναλυτή από την αρχή.

## Σύνοψη κεφαλαίου

Αλφάβητο είναι ένα σύνολο συμβόλων. Συμβολοσειρά είναι μια πεπερασμένη ακολουθία συμβόλων ενός αλφαβήτου. Γλώσσα είναι κάθε σύνολο συμβολοσειρών. Η κενή συμβολοσειρά δεν περιέχει κανένα σύμβολο και συμβολίζεται με  $\epsilon$ .

Ένα πεπερασμένο αυτόματο (FA) είναι ένας αναγνωριστής γλωσσών. Διαθέτει πεπερασμένη μνήμη και μια ταινία εισόδου: κάθε σύμβολο εισόδου που διαβάζεται οδηγεί το αυτόματο σε μια νέα κατάσταση με βάση την τρέχουσα κατάστασή του και το σύμβολο που καταναλώνεται. Το αυτόματο αποδέχεται την είσοδο αν βρίσκεται σε τελική κατάσταση όταν έχει καταναλωθεί όλη η είσοδος αλλιώς την απορρίπτει.

Μια κανονική έκφραση αποτελείται από ανεξάρτητα σύμβολα και χρήστη των τριών πράξεων Kleene: ένωση (+), παράθεση και star (\*). Το star μιας γλώσσας δίνεται από όλους τους πιθανούς τρόπους παράθεσης συμβολοσειρών της γλώσσας, επιτρέπονται οι επαναλήψεις ενώ η κενή συμβολοσειρά ανήκει πάντα στο star μιας γλώσσας.

Ένα μη ντετερμινιστικό πεπερασμένο αυτόματο (NFA) μπορεί να έχει καμία, μία ή πολλαπλές μεταβάσεις για ένα συγκεκριμένο σύμβολο εισόδου. Αποδέχεται την είσοδο αν υπάρχει κάποια ακολουθία μεταβάσεων που το οδηγεί σε τελική κατάσταση. Ο μη ντετερμινισμός μπορεί να θεωρηθεί σαν ένα δένδρο, ή σαν την έννοια 'μάντεψε και επιβεβαίωσε'. Επιτρέπονται  $\epsilon$ -μεταβάσεις, σύμφωνα με τις οποίες το NFA μπορεί να αλλάζει κατάσταση χωρίς την κατανάλωση συμβόλων της εισόδου.

Το θεώρημα του Kleene δηλώνει ότι τα ακόλουθα είναι ισοδύναμα για μια γλώσσα: υπάρχει DFA για τη γλώσσα, υπάρχει NFA για τη γλώσσα και υπάρχει κανονική έκφραση για τη γλώσσα. Η απόδειξη παρέχει έναν αλγόριθμο για να κάνουμε μετατροπές από τη μια μορφή στην άλλη. Ο τρόπος μετατροπής NFA σε DFA είναι η κατασκευή υποσυνόλων.

Μια γλώσσα είναι κανονική όταν μπορούμε να κατασκευάσουμε για αυτή FA ή κανονική έκφραση. Στην κλάση των κανονικών γλωσσών ισχύει η κλειστότητα ως προς τις πράξεις ένωση, παράθεση,

star, τομή και συμπλήρωση. Για να δείξουμε ότι μια γλώσσα δεν είναι κανονική, μπορούμε είτε να δείξουμε ότι υπάρχει ένα άπειρο σύνολο διακρίσιμων ανά ζεύγη συμβολοσειρών, είτε να χρησιμοποιήσουμε το Pumping Lemma αποδεικνύοντας ότι υπάρχει κάποια συμβολοσειρά από την οποία δεν μπορούν να ‘αντλούνται’ συμβολοσειρές που να ανήκουν στη γλώσσα.

Τα πεπερασμένα αυτόματα βρίσκουν εφαρμογή σε αλγόριθμους αναγνώρισης συμβολοσειρών, σε δικτυακά πρωτόκολλα και σε λεκτικούς αναλυτές.

## Κεφάλαιο 3

# Γραμματικές χωρίς συμφραζόμενα και αυτόματα στοίβας

### 3.1 Γραμματικές και γλώσσες χωρίς συμφραζόμενα

Οι γραμματικές ορίζονται με βάση: δύο σύνολα συμβόλων, τα τερματικά (terminals), που αντιστοιχούν στα σύμβολα του αλφαβήτου πάνω στο οποίο ορίζεται η γλώσσα και τα μη τερματικά (non-terminals) που λειτουργούν σα μεταβλητές, κάποιους κανόνες, τρόπους, δηλαδή, για εκκίνηση των λειτουργιών και για αντικατάσταση των μη τερματικών συμβόλων με άλλες συμβολοσειρές. Ακολουθιακή εφαρμογή των κανόνων ενδέχεται να οδηγήσει σε συμβολοσειρά τερματικών συμβόλων οπότε η είσοδος ανήκει στη γλώσσα που ορίζεται από τη συγκεκριμένη γραμματική. Η διαδικασία αυτή καλείται παραγωγή. Ο καθορισμός του αν κάποια συμβολοσειρά ανήκει σε γλώσσα που παράγεται από συγκεκριμένη γραμματική πραγματοποιείται με συντακτική ανάλυση (parsing) της συμβολοσειράς, δηλαδή με εύρεση μιας ακολουθίας κανόνων που να παράγουν τη συμβολοσειρά. Οι γλώσσες ορίζονται με τρεις τρόπους:

- άμεσα,
- με περιγραφή μιας μηχανής που τις αναγνωρίζει ή
- με ορισμό μιας γραμματικής που τις παράγει.

Οι τρόποι αυτοί σχετίζονται άμεσα. Στη συνέχεια θα μελετήσουμε τους τρόπους αυτής της συσχέτισης. Οι γλώσσες, οι μηχανές και οι γραμματικές οργανώνονται σε κλάσεις με βάση κάποιες ιδιότητες που ενδεχομένως έχουν και σε πολλές περιπτώσεις υπάρχει αντιστοιχία μεταξύ ανάλογων κλάσεων γλωσσών, μηχανών και γραμματικών. Επιπλέον, ορίζονται λειτουργίες πάνω σε γραμματικές. Αυτό σημαίνει ότι μια ή περισσότερες γραμματικές μπορούν να συνδυαστούν ή να τροποποιηθούν με συστηματικό τρόπο ώστε να παραχθούν νέες γραμματικές. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι ιδιότητες κλειστότητας κλάσεων γλωσσών ως προς συγκεκριμένες λειτουργίες - πράξεις, δηλαδή

το αν η γλώσσα, που θα προκύψει από εφαρμογή των πράξεων αυτών σε μια δοσμένη γλώσσα, θα ανήκει στην ίδια κλάση με την αρχική.

Προκειμένου να οριστούν γλώσσες είναι πολλές φορές απλούστερο να χρησιμοποιούνται γραμματικές. Το βασικό πλεονέκτημα αυτής της προσέγγισης έγκειται κυρίως στη χρήση μικρού αριθμού κανόνων για την περιγραφή μιας γλώσσας με μεγάλο αριθμό προτάσεων. Για παράδειγμα, το γεγονός ότι μια πρόταση στα ελληνικά μπορεί να αποτελείται από υποκείμενο και μια κατηγορηματική φράση εκφράζεται από τον ακόλουθο κανόνα (πρόταση)  $\rightarrow$  (υποκείμενο)(κατηγορηματική). Κατ' αναλογία μια έκφραση υποκειμένου της ελληνικής που αποτελείται από όνομα περιγράφεται από το γραμματικό κανόνα (υποκείμενο)  $\rightarrow$  (όνομα). Όμοια η πρόταση "Αυτή τραγουδά ένα τραγούδι" μπορεί να περιγραφεί.

$$\begin{aligned}
 \langle \text{πρόταση} \rangle &\rightarrow \langle \text{υποκείμενο} \rangle \langle \text{κατηγορηματική} \rangle \\
 \langle \text{υποκείμενο} \rangle &\rightarrow \langle \text{ουσιαστικό} \rangle \\
 \langle \text{κατηγορηματική} \rangle &\rightarrow \langle \text{ρήμα} \rangle \langle \text{ουσιαστικό} \rangle \\
 \langle \text{ουσιαστικό} \rangle &\rightarrow \langle \text{όνομα} \rangle \\
 \langle \text{ουσιαστικό} \rangle &\rightarrow \langle \text{συμβολοσειρά} \rangle \\
 \langle \text{όνομα} \rangle &\rightarrow \langle \text{κεφαλαίος-χαρακτήρας} \rangle \langle \text{συμβολοσειρά} \rangle \\
 \langle \text{συμβολοσειρά} \rangle &\rightarrow \langle \text{συμβολοσειρά} \rangle \langle \text{χαρακτήρας} \rangle \\
 \langle \text{συμβολοσειρά} \rangle &\rightarrow \langle \text{χαρακτήρας} \rangle \\
 \langle \text{χαρακτήρας} \rangle &\rightarrow \langle a \rangle \\
 &\vdots \\
 \langle \text{χαρακτήρας} \rangle &\rightarrow \langle z \rangle \\
 \langle \text{κεφαλαίος-χαρακτήρας} \rangle &\rightarrow \langle A \rangle \\
 &\vdots \\
 \langle \text{κεφαλαίος-χαρακτήρας} \rangle &\rightarrow \langle Z \rangle
 \end{aligned}$$

Οι παραπάνω γραμματικοί κανόνες επιτρέπουν την παραγωγή προτάσεων της μορφής "Αυτός τραγουδά ένα τραγούδι", δηλαδή μπορεί να μεταβάλλεται το υποκείμενο. Παρόλα αυτά πάντως το γραμματικό σύστημα που αμυδρά περιγράψαμε είναι ατελές και δεν είναι σε θέση να καθορίσει πλήρως την ελληνική γλώσσα. Οι γραμματικές που θα μας απασχολήσουν αποτελούνται από πεπερασμένο αριθμό κανόνων της παραπάνω μορφής και καλούνται Γραμματικές Τύπου 0 (**Type 0 grammars**) και οι τυπικές γλώσσες που παράγουν ονομάζονται Γλώσσες Τύπου 0 (**Type 0 languages**). Στη συνέχεια παραθέτουμε έναν αυστηρό ορισμό των γραμματικών τύπου 0: μια γραμματική τύπου 0



ορίζεται σαν ένα μαθηματικό σύστημα που αποτελείται από μια τετράδα  $\langle N, \Sigma, P, S \rangle$ , όπου:

- $N$ : ένα πεπερασμένο σύνολο συμβόλων ή μεταβλητών (variables), δηλαδή ένα αλφάβητο του οποίου τα στοιχεία ονομάζονται μη τερματικά σύμβολα (nonterminal symbols).
- $\Sigma$ : ένα αλφάβητο ξένο με το  $N$ , του οποίου τα στοιχεία ονομάζονται τερματικά σύμβολα (terminal symbols).
- $P$ : μια σχέση με πεπερασμένο πληθικό αριθμό (cardinality) ορισμένη στο  $(N \cup \Sigma)^*$ , της οποίας τα στοιχεία λέγονται κανόνες παραγωγής. (production rules). Κάθε κανόνας  $(a, b)$  στην  $P$ , συμβολίζεται με  $a \rightarrow b$ , και πρέπει να έχει τουλάχιστο ένα μη τερματικό σύμβολο στο μέρος  $a$ . Σε κάθε τέτοιο κανόνα, το τμήμα  $a$  καλείται αριστερότερο μέρος (left-hand side) της παραγωγής, ενώ το  $b$  καλείται δεξιότερο τμήμα (right-hand side) της παραγωγής.
- $S$ : είναι ένα διακεκριμένο στοιχείο του αλφαβήτου  $N$  που ονομάζεται αρχικό σύμβολο (start symbol).

**Παράδειγμα 3.1.1.** Η τετράδα  $\langle N, \Sigma, P, S \rangle$  ορίζει μια γραμματική τύπου 0 αν  $N = \{S\}$ ,  $\Sigma = \{a, b\}$ , και  $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$ . Από τον ορισμό, η γραμματική έχει ένα μη τερματικό σύμβολο  $S$ , δύο τερματικά σύμβολα  $a$  και  $b$ , και δύο κανόνες - παραγωγές  $S \rightarrow aSb$  και  $S \rightarrow \epsilon S$ . Και οι δύο κανόνες έχουν ένα αριστερότερο τμήμα που αποτελείται μόνο από το μη τερματικό σύμβολο  $S$ . Το δεξιότερο τμήμα του πρώτου κανόνα είναι το  $aSb$ , ενώ του δεύτερου είναι το  $\epsilon$ . Η τετράδα  $\langle N_1, \Sigma_1, P_1, S \rangle$  δεν ορίζει κάποια γραμματική αν το  $N_1$  είναι το σύνολο των φυσικών αριθμών, ή αν το  $\Sigma_1$  είναι το κενό, αφού τα  $N_1$  και  $\Sigma_1$  πρέπει να είναι αλφάβητα. Επίσης, αν  $N_2 = \{S\}$ ,  $\Sigma_2 = \{a, b\}$ , και  $P_2 = \{S \rightarrow aSb, S \rightarrow \epsilon, ab \rightarrow S\}$  τότε η τετράδα  $\langle N_2, \Sigma_2, P_2, S \rangle$  δεν ορίζει γραμματική, γιατί ο κανόνας  $ab \rightarrow S$  δεν ικανοποιεί την απαίτηση ότι κάθε κανόνας πρέπει να περιέχει τουλάχιστο ένα μη τερματικό σύμβολο στο αριστερότερο τμήμα του.

Παρακάτω δίνεται ένα παράδειγμα γραμματικής χωρίς συμφραζόμενα.

**Παράδειγμα 3.1.2.** Οι κανόνες της γραμματικής είναι:

$$S \rightarrow 0S1$$

$$S \rightarrow \epsilon$$

Μία γραμματική αποτελείται από:

- ένα σύνολο μεταβλητών (που καλούνται και μη τερματικά)
- ένα σύνολο τερματικών συμβόλων (ή αλλιώς το αλφάβητο) και
- μία ακολουθία παραγωγών (που καλούνται κανόνες)

Στο παραπάνω παράδειγμα, το  $S$  είναι η μόνη μεταβλητή. Τα τερματικά είναι το 0 και το 1. Υπάρχουν δύο κανόνες. Συνηθίζεται να χρησιμοποιούνται κεφαλαία γράμματα για τις μεταβλητές. Ας δούμε πώς λειτουργεί μία γραμματική. Ένας κανόνας επιτρέπει την αντικατάσταση μιας μεταβλητής μιας συμβολοσειράς με ό,τι εμφανίζεται στη πλευρά του κανόνα. Μια (πεπερασμένη) συμβολοσειρά  $w$ , που περιέχει τερματικά, παράγεται από τη γραμματική αν, ξεκινώντας από την αρχική μεταβλητή  $S$  της γραμματικής, μπορούμε μέσω παραγωγών να καταλήξουμε σε αυτή τη συμβολοσειρά. Η ακολουθία των συμβολοσειρών που προκύπτει αποτελεί μια παραγωγή για τη  $w$ . Φυσικά υπάρχουν και άλλοι τύποι γραμματικών, πάντως γραμματικές αυτού του τύπου καλούνται γραμματικές χωρίς συμφροζόμενα (CFG). Μια γλώσσα είναι χωρίς συμφροζόμενα (CFL) αν παράγεται από μια CFG.

**Παράδειγμα 3.1.3.** Συνέχεια παραδείγματος 3.1.2. Η συμβολοσειρά 0011 ανήκει στην παραπάνω γλώσσα. Μια παραγωγή είναι:

$$S \rightarrow 0S1 \rightarrow 00S11$$

Στα πρώτα δύο βήματα χρησιμοποιείται ο πρώτος κανόνας. Στο τελευταίο βήμα χρησιμοποιείται ο δεύτερος κανόνας. Τι συμβολοσειρές περιέχει η γλώσσα αυτή; Κάθε παραγόμενη συμβολοσειρά έχει ίδιο αριθμό από 0 και 1, επομένως, ουσιαστικά, πρόκειται για τη μη κανονική γλώσσα:  $\{0^n 1^n : n \geq 0\}$ .

Παρατηρούμε ότι κάθε CFG ενέχει μια αναδρομικότητα. Στο προηγούμενο παραδειγμα, η CFG παράγει συμβολοσειρές ξεκινώντας από την κενή συμβολοσειρά και προσθέτοντας στην αρχή και το τέλος, αναδρομικά, τα σύμβολα 0 και 1, αντίστοιχα. Χωρίς την αναδρομικότητα, η γραμματική θα μπορούσε να παράγει μόνο ένα πεπερασμένο πλήθος συμβολοσειρών.

Όσον αφορά στο συμβολισμό, η προηγούμενη CFG θα μπορούσε να γραφεί και ως:

$$S \rightarrow 0S1|ε,$$

όπου η το σύμβολο  $|$  σημαίνει 'ή'.

**Παράδειγμα 3.1.4.** Νωρίτερα, διαπιστώσαμε ότι η γλώσσα  $P$  που ορίζεται στο αλφάβητο  $\{a, b\}$  και περιέχει παλίνδρομες συμβολοσειρές δεν κανονική. Όμως, μπορούμε να ορίσουμε μια CFG για αυτή εκμεταλλευόμενοι την αναδρομικότητά της. Εάν αφαιρέσουμε το πρώτο και το τελευταίο σύμβολο από μια παλίνδρομη συμβολοσειρά, τότε αυτή που απομένει είναι επίσης παλίνδρομη. Επίσης, προσθέτοντας στην αρχή και το τέλος μίας παλίνδρομης συμβολοσειράς το ίδιο σύμβολο προκύπτει νέα παλίνδρομη συμβολοσειρά. Με βάση αυτή την παρατήρηση η αντίστοιχη γραμματική περιέχει τους εξής κανόνες:

$$P \rightarrow aPa|bPb|ε$$

Με τον τρόπο αυτό, μπορούμε να παράγουμε όλες τις παλίνδρομες συμβολοσειρές με άρτιο μήκος. Πώς θα μπορούσαμε να τροποποιήσουμε αυτή τη CFG ώστε να παράγει όλες τις παλίνδρομες συμβολοσειρές, δηλαδή τη γλώσσα  $\{ww^R : w \in \Sigma^*\}$ ;

Γενικά, τα μη τερματικά σύμβολα μιας γραμματικής τύπου 0 συμβολίζονται με το γράμμα  $S$  καθώς επίσης και με τα κεφαλαία γράμματα του αγγλικού αλφαβήτου  $A, B, C, D$ , και  $E$ . Το αρχικό σύμβολο συμβολίζεται με  $S$ . Τα τερματικά σύμβολα συμβολίζονται με δυαδικά ψηφία ή με μικρά γράμματα του αγγλικού αλφαβήτου  $a, b, c, d$ , και  $e$ . Για σύμβολα που δεν έχουν ιδιαίτερη σημασία χρησιμοποιούνται συνήθως γράμματα όπως τα  $X, Y$ , και  $Z$ . Συμβολοσειρές τερματικών συμβόλων συμβολίζονται με τα μικρά γράμματα  $u, v, w, x, y$ , και  $z$ . Συμβολοσειρές που αποτελούνται από τερματικά και μη τερματικά σύμβολα συμβολίζονται με μικρά γράμματα του ελληνικού αλφαβήτου  $\alpha, \beta$ , και  $\gamma$ . Επί πλέον, για λόγους ευκολίας, ακολουθίες κανόνων της μορφής:

$$\begin{aligned} a &\rightarrow b_1 \\ a &\rightarrow b_2 \\ a &\dots \\ a &\rightarrow b_n \end{aligned}$$

συμβολίζονται σαν

$$\begin{aligned} a &\rightarrow b_1 \\ &\rightarrow b_2 \\ &\dots \\ &\rightarrow b_n \end{aligned}$$

**Παράδειγμα 3.1.5.** Η τετράδα  $\langle N, \Sigma, P, S \rangle$  ορίζει μια γραμματική τύπου 0 αν  $N = \{S, B\}$ ,  $\Sigma = \{a, b, c\}$ , και το  $P$  αποτελείται από τους ακόλουθους κανόνες.

$$\begin{aligned} S &\rightarrow aBSc \\ &\rightarrow abc \\ &\rightarrow e \\ Ba &\rightarrow aB \\ Bb &\rightarrow bb \end{aligned}$$

Το μη τερματικό σύμβολο  $S$  αποτελεί το αριστερότερο τμήμα των τριών πρώτων κανόνων.  $Ba$  είναι το αριστερότερο τμήμα του τέταρτου κανόνα ενώ  $Bb$  είναι το αριστερότερο τμήμα του πέμπτου κανόνα. Το δεξιότερο τμήμα  $aBSc$  του πρώτου κανόνα περιέχει και τερματικά και μη τερματικά σύμβολα. Το δεξιότερο τμήμα  $abc$  του δεύτερου κανόνα αποτελείται μόνο από τερματικά σύμβολα.

Εκτός από την τετριμμένη περίπτωση του τρίτου κανόνα του οποίου το δεξιότερο τμήμα είναι  $\epsilon$  το δεξιότερο τμήμα των υπόλοιπων κανόνων περιέχει μόνο μη τερματικά σύμβολα, αν και θα επιτρεπόταν και η χρησιμοποίηση τερματικών συμβόλων.

Οι γραμματικές “γεννούν” γλώσσες με επαναλαμβανόμενη οροποποίηση δοσμένων συμβολοσειρών σύμφωνα με κάποιον από τους κανόνες της γραμματικής αυτής,  $G = \langle N, \Sigma, P, S \rangle$ . Έτσι δοθείσης μιας συμβολοσειράς  $g$  και ενός κανόνα της γραμματικής  $a \rightarrow b$ , μπορεί να παραχθεί μια νέα συμβολοσειρά με αντικατάσταση των εμφανίσεων της υποσυμβολοσειράς  $a$  στη  $g$  με  $b$ . Μια παραγωγή συμβολίζεται  $X \rightarrow Y$ , όπου  $X \in (V \cup T)^+$  που σημαίνει ότι η  $X$  ανήκει στο σύνολο των συμβολοσειρών που αποτελούνται από μεταβλητές και τερματικά σύμβολα αλλά η  $X$  δεν είναι η κενή συμβολοσειρά και  $Y \in (V \cup T)^*$  που σημαίνει ότι η  $Y$  ανήκει στο σύνολο των συμβολοσειρών που αποτελούνται από μεταβλητές και τερματικά σύμβολα και επιπλέον μπορεί να είναι η κενή συμβολοσειρά. Οι παραγωγές είναι κανόνες που χρησιμοποιούνται για να καθοριστεί αν μια συμβολοσειρά ανήκει ή όχι σε μια γλώσσα. Έστω μια γλώσσα  $L$  που ορίζεται από τη γραμματική  $G = (V, T, S, P)$ . Η πιστοποίηση του αν μια συμβολοσειρά ανήκει στη γλώσσα  $L$  γίνεται ως εξής:

1. Ξεκινάμε με μια συμβολοσειρά  $w$  που αποτελείται μόνο από το αρχικό σύμβολο  $S$ .
2. Βρίσκουμε μια υποσυμβολοσειρά της  $w$ , έστω  $x$  που ταιριάζει με το αριστερό τμήμα μιας παραγωγής  $p$  του  $P$ .
3. Αντικαθιστούμε την  $x$  στη  $w$  με το τμήμα που βρίσκεται στο δεξιό μέρος της παραγωγής  $p$ .
4. Επαναλαμβάνουμε τα βήματα 2 και 3 έως ότου η συμβολοσειρά περιλαμβάνει μόνο σύμβολα του  $T$  (δηλαδή δεν περιέχει μεταβλητές).
5. Η τελική συμβολοσειρά ανήκει στη γλώσσα  $L$ .

Έστω ότι η συμβολοσειρά  $w$  μπορεί να γραφεί σαν  $uxv$ , με  $u, v$  στοιχεία του  $(V \cup T)^*$   $x$  στοιχείο του  $(V \cup T)^+$  υπάρχει παραγωγή  $x \rightarrow y$  Τότε μπορούμε να γράψουμε  $uxv \Rightarrow uyv$  που σημαίνει ότι η  $uxv$  παράγει την  $uyv$ .

Χρήσιμοι είναι οι ακόλουθοι συμβολισμοί:

- $S \Rightarrow T$  : η  $S$  παράγει την  $T$  σε ακριβώς ένα βήμα.
- $S \xRightarrow{*} T$  : η  $S$  παράγει την  $T$  σε μηδέν ή περισσότερα βήματα.
- $S \xRightarrow{+} T$  : η  $S$  παράγει την  $T$  σε ένα ή περισσότερα βήματα.

Τυπικά, λέμε ότι μια συμβολοσειρά  $g$  παράγει άμεσα – δηλαδή σε ένα βήμα – με βάση μια γραμματική  $G$  μια συμβολοσειρά  $g'$  και συμβολίζουμε  $g \xRightarrow{*}_G g'$  αν: η συμβολοσειρά  $g'$  μπορεί να παραχθεί από τη  $g$  με αντικατάσταση ενός τμήματος  $a$  της  $g$  με την υποσυμβολοσειρά  $b$ , και  $a \rightarrow b$  είναι ένας κανόνας της γραμματικής  $G$ .

**Παράδειγμα 3.1.6.** Έστω η γραμματική  $G = \langle N, \Sigma, P, S \rangle$  με  $N = \{S\}$ ,  $\Sigma = \{a, b\}$ , και  $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$ , τότε τα  $\epsilon$  και  $aSb$  παράγονται σε ένα βήμα (άμεσα) από το σύμβολο  $S$ . Όμοια τα  $ab$  και  $a_2Sb_2$  παράγονται σε ένα βήμα από τη συμβολοσειρά  $aSb$ . Η κενή συμβολοσειρά  $\epsilon$  παράγεται άμεσα από το σύμβολο  $S$ , ενώ η  $ab$  παράγεται σε ένα βήμα από τη  $aSb$ , σύμφωνα με τον κανόνα  $S \rightarrow \epsilon$ . Η  $aSb$  παράγεται σε ένα βήμα από το  $S$ , και  $a_2Sb_2$  παράγεται άμεσα από τη συμβολοσειρά  $aSb$ , σύμφωνα με τον κανόνα  $S \rightarrow aSb$ . Αν θεωρήσουμε τη γραμματική  $G$  με  $N = \{S, B\}$ ,  $\Sigma = \{a, b, c\}$  και  $P = \{S \rightarrow aBSc, S \rightarrow abc, S \rightarrow \epsilon, Ba \rightarrow aB, Bb \rightarrow bb\}$ , τότε  $aBaBabccc \Rightarrow_G aaBBabccc$  και  $aBaBabccc \Rightarrow_G aBaaBbccc$  σύμφωνα με τον κανόνα  $Ba \rightarrow aB$ . Επιπλέον, καμιά άλλη συμβολοσειρά δεν μπορεί να παραχθεί σε ένα βήμα από την  $aBaBabccc$  στα πλαίσια της γραμματικής  $G$ . Λέμε ότι η  $g$  απλά παράγει τη  $g'$  με βάση τη γραμματική  $G$ , και συμβολίζουμε  $g \xrightarrow{*}_G g'$ , αν  $g_0 \Rightarrow_G \dots \Rightarrow_G g'_n$  για κάποια  $g_0, \dots, g_n$  τέτοια ώστε  $g_0 = g$  και  $g_n = g'$ . Στην περίπτωση αυτή, η ακολουθία  $g_0 \Rightarrow_G \dots \Rightarrow_G g'_n$  καλείται παραγωγή (derivation) από τη  $g$  μέχρι τη  $g'$  και έχει μήκος (length) ίσο με  $n$ . Αν  $g_0 = S$ , κάθε μια από τις  $g_0, \dots, g_n$  λέγεται προτασιακός τύπος (sentential forms). Ένας προτασιακός τύπος που δεν περιέχει τερματικά σύμβολα καλείται πρόταση (sentence).

**Παράδειγμα 3.1.7.** Έστω η γραμματική  $G \langle N, \Sigma, P, S \rangle$  με  $N = \{S\}$ ,  $\Sigma = \{a, b\}$ , και  $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$ , τότε υπάρχει παραγωγή της  $a_4Sb_4$  από το  $S$ . Η παραγωγή  $S \xrightarrow{*}_G a_4Sb_4$  έχει μήκος 4, και αναλυτικά είναι η εξέλιξη  $S \Rightarrow_G aSb \Rightarrow_G a_2Sb_2 \Rightarrow_G a_3Sb_3 \Rightarrow_G a_4Sb_4$ .

Μια συμβολοσειρά ανήκει σε μια γλώσσα που παράγεται από τη γραμματική  $G$  αν και μόνον αν είναι συμβολοσειρά τερματικών συμβόλων και μπορεί να παραχθεί από το αρχικό σύμβολο. Η γλώσσα που παράγεται από τη γραμματική  $G$ , συμβολίζεται  $L(G)$ , και είναι το σύνολο όλων των συμβολοσειρών που αποτελούνται μόνο από τερματικά σύμβολα και μπορούν να παραχθούν από το αρχικό σύμβολο, δηλαδή πρόκειται για το σύνολο  $\{w | w \text{ ανήκει στο } \Sigma^*, \text{ και } S \xrightarrow{*}_G w\}$ . Κάθε συμβολοσειρά της γλώσσας  $L(G)$  λέμε τότε ότι παράγεται από τη γραμματική  $G$ .

**Παράδειγμα 3.1.8.** Θεωρούμε τη γραμματική  $G = \langle N, \Sigma, P, S \rangle$  με  $N = \{S\}$ ,  $\Sigma = \{a, b\}$ , και  $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$ . Η κενή συμβολοσειρά  $\epsilon$  ανήκει στη γλώσσα που παράγει η γραμματική  $G$  γιατί υπάρχει η παραγωγή  $S \xrightarrow{*}_G \epsilon$ . Η συμβολοσειρά  $ab$  ανήκει στη  $L(G)$ , αφού υπάρχει η παραγωγή  $S \Rightarrow_G aSb \Rightarrow_G Gab$ . Τέλος, η συμβολοσειρά  $a_2b_2$  ανήκει στη γλώσσα  $L(G)$ , αφού μπορεί να οριστεί η παραγωγή  $S \Rightarrow_G aSb \Rightarrow_G a_2Sb_2 \Rightarrow_G a_2b_2$ . Με άλλα λόγια η γλώσσα  $L(G)$  που παράγει η γραμματική  $G$  αποτελείται από συμβολοσειρές της μορφής  $a \dots ab \dots b$  στις οποίες ο αριθμός των  $a$  ισούται με τον αριθμό των  $b$ , δηλαδή,  $L(G) = \{a_i b_i | i \text{ είναι φυσικός αριθμός}\}$ . Η συμβολοσειρά  $aSb$  δεν ανήκει στη γλώσσα  $L(G)$  αφού περιέχει ένα μη τερματικό σύμβολο, το  $S$ . Η συμβολοσειρά  $a_2b$  επίσης δεν ανήκει στη γλώσσα  $L(G)$  αφού δεν μπορεί να παραχθεί από το αρχικό σύμβολο  $S$  με βάση τη γραμματική  $G$ . Σε ό,τι ακολουθεί, οι συμβολισμοί  $g \Rightarrow g'$  και  $g \xrightarrow{*}_G g'$  θα χρησιμοποιούνται αντίστοιχα αντί για τους  $g \Rightarrow_G g'$  και  $g \xrightarrow{*}_G g'$ , στη περίπτωση που η γραμματική  $G$  υπονοείται.

Επιπλέον, γραμματικές τύπου 0 και γλώσσες τύπου 0 θα αναφέρονται απλά σε γραμματικές και γλώσσες, αντίστοιχα.

**Παράδειγμα 3.1.9.** Έστω η γραμματική  $G = \langle N, \Sigma, P, S \rangle$  με  $N = \{S, B\}$ ,  $\Sigma = \{a, b, c\}$  και  $P = \{S \rightarrow aBSc, S \rightarrow abc, S \rightarrow \epsilon, Ba \rightarrow aB, Bb \rightarrow bb\}$ . Στη συνέχεια παρουσιάζεται μια παραγωγή για τη συμβολοσειρά  $a^3b^3c^3$ . Τα υπογραμμισμένα τμήματα και τα τμήματα που φέρουν γραμμή στο πάνω μέρος τους είναι αντίστοιχα τα αριστερότερα και δεξιότερα μέρη των κανόνων που χρησιμοποιούνται στη παραγωγή.

$$\begin{aligned}
 \underline{S} &\Rightarrow \overline{aBSc} \\
 &\Rightarrow aB\overline{aBSc} \\
 &\Rightarrow aBa\overline{Babc} \\
 &\Rightarrow aBa\overline{aB}bcc \\
 &\Rightarrow \underline{aBa}b\overline{bb}ccc \\
 &\Rightarrow \overline{aa}B\overline{abb}ccc \\
 &\Rightarrow \overline{aaa}B\overline{bb}ccc \\
 &\Rightarrow \overline{aaab}b\overline{bb}ccc
 \end{aligned}$$

Η γλώσσα που παράγεται από τη γραμματική  $G$  αποτελείται από συμβολοσειρές της μορφής  $a \dots ab \dots bc \dots c$  στις οποίες υπάρχει ίδιος αριθμός από  $a$ ,  $b$ , και  $c$ , δηλαδή,

$$L(G) = \{a_i b_i c_i \mid i \text{ είναι φυσικός αριθμός}\}.$$

Οι δύο πρώτοι κανόνες της  $G$  χρησιμοποιούνται για την παραγωγή προτασιακών τύπων που ακολουθούν το σχήμα  $aBaB \dots aBabc \dots c$ , δηλαδή ο αριθμός των  $a$  ισούται με τον αριθμό των  $c$  και είναι κατά 1 μεγαλύτερος από τον αριθμό των  $B$ . Ο κανόνας  $Ba \rightarrow aB$  χρησιμοποιείται για την αντικατάσταση των  $B$  που εμφανίζονται στο δεξιότερο μέρος των προτασιακών τύπων ενώ ο κανόνας  $Bb \rightarrow bb$  χρησιμοποιείται για την εξάλειψη των  $B$  και την αντικατάστασή τους με  $b$ .

**Παράδειγμα 3.1.10.** Στη συνέχεια, κατασκευάζουμε μια CFG που να παράγει όλες τις δυαδικές συμβολοσειρές με άρτιο αριθμό 0. Στόχος είναι να αποσυνθέσουμε τέτοιες συμβολοσειρές σε μικρότερα τμήματα. Αν το πρώτο σύμβολο είναι 1, μένει άρτιος αριθμός 0 που πρέπει να παραχθούν, ενώ αν το πρώτο σύμβολο είναι 0 τότε πρέπει να βρούμε το επόμενο 0 και μετά να παράγουμε πάλι άρτιο πλήθος 0. Η σύντομη αυτή ανάλυση οδηγεί στην ακόλουθη γραμματική χωρίς συμφραζόμενα:

$$S \rightarrow 1S|0A0S|\epsilon$$

$$A \rightarrow 1A|\epsilon$$

Φυσικά, μπορούν να υπάρχουν περισσότερες από μία CFG για μία γλώσσα. Για παράδειγμα, όταν το πρώτο σύμβολο είναι 0, πρέπει μετά να παράγουμε άρτιο αριθμό 0. Χρησιμοποιώντας τη μεταβλητή  $T$  για να παράγουμε όλες τις δυαδικές συμβολοσειρές με άρτιο αριθμό 0, μπορούμε να κατασκευάσουμε την ακόλουθη CFG:

$$S \rightarrow 1S|0T|\epsilon$$

$$T \rightarrow 1T|0S$$

**Παράδειγμα 3.1.11.** Στη συνέχεια, κατασκευάζουμε μια CFG για τη γλώσσα που περιγράφεται από την κανονική έκφραση  $00^*11^*$ . Η ιδέα είναι ότι η γλώσσα προκύπτει από την παράθεση δύο γλωσσών: της γλώσσας που περιέχει όλες τις συμβολοσειρές που αποτελούνται από 0 και της γλώσσας που περιέχει όλες τις συμβολοσειρές που αποτελούνται από 1.

$$S \rightarrow CD$$

$$C \rightarrow 0C|0$$

$$D \rightarrow 1D|1$$

Το επόμενο παράδειγμα αναφέρεται στο συμπλήρωμα της γλώσσας του προηγούμενου παραδείγματος. Δεν υπάρχει κάποιος προφανής τρόπος να μετατρέψουμε μια γραμματική για κάποια γλώσσα στην αντίστοιχη για τη συμπληρωματική της. Στόχος μας είναι να κατασκευάσουμε μια γραμματική που να παράγει όλες τις συμβολοσειρές που δεν είναι της μορφής  $0^i1^j$  με  $i, j > 0$ . Μία προσέγγιση θα ήταν να χωρίσουμε το σύνολο των συμβολοσειρών σε τρεις κατηγορίες:

- συμβολοσειρές που περιέχουν 1 που ακολουθείται από 0
- συμβολοσειρές που περιέχουν μόνο 0
- συμβολοσειρές που περιέχουν μόνο 1

Έτσι, κατασκευάζουμε την παρακάτω CFG:

**Παράδειγμα 3.1.12.** CFG για τη συμπληρωματική γλώσσα της  $\{0^i1^j : i, j > 0\}$ :

$$S \rightarrow A|B|C$$

$$A \rightarrow D10D$$

$$D \rightarrow 0D|1D|\epsilon$$

$$B \rightarrow 0B|0$$

$$C \rightarrow 1C|1$$

Η μεταβλητή  $A$  παράγει όλες τις συμβολοσειρές με ένα 0 και 1 με λάθος διάταξη, η  $B$  παράγει τις συμβολοσειρές που περιέχουν μόνο 0, η  $C$  παράγει τις συμβολοσειρές που περιέχουν μόνο 1 στρινγς ενώ η  $D$  παράγει όλες τις συμβολοσειρές της γλώσσας.

Για να βεβαιωθούμε ότι μια γραμματική πράγματι παράγει τη ζητούμενη γλώσσα πρέπει να ελέγξουμε δύο συνθήκες: (1) η γραμματική διαθέτει συνέπεια «consistency»: ό,τι παράγει η γραμματική ταιριάζει με την περιγραφή της γλώσσας και (2) ότι η γραμματική διαθέτει πληρότητα (completeness): παράγει δηλαδή όλες τις συμβολοσειρές της δοσμένης γλώσσας.

**Παράδειγμα 3.1.13.** Έστω η CFG:  $S \rightarrow 0S1S|1S0S|\epsilon$ . Η συμβολοσειρά 011100 ανήκει στη γλώσσα που παράγεται από την πιο πάνω γραμματική. Μια παραγωγή είναι η:

$$S \Rightarrow 0S1S \Rightarrow 01S \Rightarrow 011S0S \Rightarrow 0111S0S0S \Rightarrow 01110S0S \Rightarrow 011100S \Rightarrow 011100.$$

Τι είδους συμβολοσειρές περιέχει η γλώσσα αυτή; Σίγουρα κάθε συμβολοσειρά με ίδιο αριθμό από 0 και 1. Επιπλέον, οποιαδήποτε συμβολοσειρά με ίσο αριθμό 0 και 1 μπορεί να παραχθεί από τη γραμματική. Γιατί;

Ανεξάρτητα από το αρχικό σύμβολο, σε κάποιο σημείο θα πρέπει να έχουν παραχθεί ίδιο πλήθος από 0 και 1. Αν το πρώτο σύμβολο της συμβολοσειράς είναι 0 τότε έχουμε ισότητα όταν παραχθεί το επόμενο 1. Αλλά, ισότητα 0 και 1 έχουμε και για τα τμήματα ανάμεσα στο 0 και το 1, μετά το 1 και φυσικά για το ίδιο το 1. Δηλαδή, μπορούμε χωρίσουμε τη συμβολοσειρά σε  $0w1x$  όπου και τα δύο τμήματα  $w$  και  $x$  ανήκουν στη γλώσσα. Για παράδειγμα, μπορούμε να χωρίσουμε τη συμβολοσειρά 00101101 ως εξής:

$$0 \left[ \begin{array}{cccc} 0 & 1 & 0 & 1 \end{array} \right] 1 \left[ \begin{array}{cc} 0 & 1 \end{array} \right]$$

$w \qquad x$

Το επιχείρημα αυτό αποδεικνύει την πληρότητα: κάθε συμβολοσειρά με ίδιο αριθμό 0 και 1 μπορεί να παραχθεί από τη γραμματική.

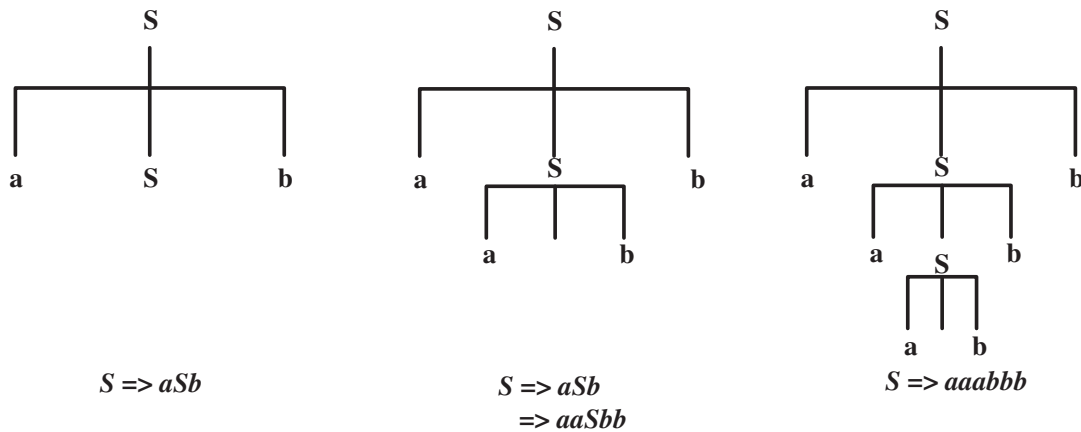
### 3.1.1 Γραφήματα Παραγωγής

Οι παραγωγές μπορούν να περιγραφούν με τις εκφράσεις που χρησιμοποιήσαμε μέχρι τώρα καθώς επίσης και με γραφήματα παραγωγής. Ένα γράφημα παραγωγής έχει ρίζα, υπάρχει διάταξη των κορυφών του και δεν περιέχει κύκλους. Κάθε κορυφή φέρει μια ετικέτα – label – που είναι ένα μη τερματικό σύμβολο ή ένα τερματικό σύμβολο ή η κενή συμβολοσειρά. Το γράφημα παραγωγής που περιγράφει την παραγωγή  $S \Rightarrow g_0 \Rightarrow \dots \Rightarrow g_n$  ορίζεται αναδρομικά με τον ακόλουθο τρόπο. Το γράφημα παραγωγής  $D_0$  που αντιστοιχεί στο αρχικό σύμβολο  $S$  περιέχει μία μόνο κορυφή που έχει σαν ετικέτα το αρχικό σύμβολο  $S$ . Αν  $a \rightarrow b$  είναι ο κανόνας που χρησιμοποιείται για την



παραγωγή  $g_i \Rightarrow g_i + 1$ , με  $0 \leq i < n$  και  $g_0 = S$ , τότε το γράφημα παραγωγής  $D_i + 1$  που αντιστοιχεί στην παραγωγή  $g_0 \Rightarrow \dots \Rightarrow g_i + 1$  προκύπτει από τον  $D_i$  με την πρόσθεση  $\max(|b|, 1)$  νέων κορυφών. Οι νέες κορυφές φέρουν σαν ετικέτες τους χαρακτήρες του  $b$ , και τοποθετούνται σαν παιδιά κάθε κορυφής στο  $D_i$  που αντιστοιχεί σε ένα χαρακτήρα του  $a$ . Επομένως τα φύλλα του δένδρικού γραφήματος  $D_i + 1$  φέρουν ετικέτες  $g_i + 1$ . Τα γραφήματα παραγωγής που έχουν μορφή κατευθυνόμενων δένδρων λέγονται και δένδρα παραγωγής (derivation trees) ή δένδρα συντακτικής ανάλυσης (parse trees).

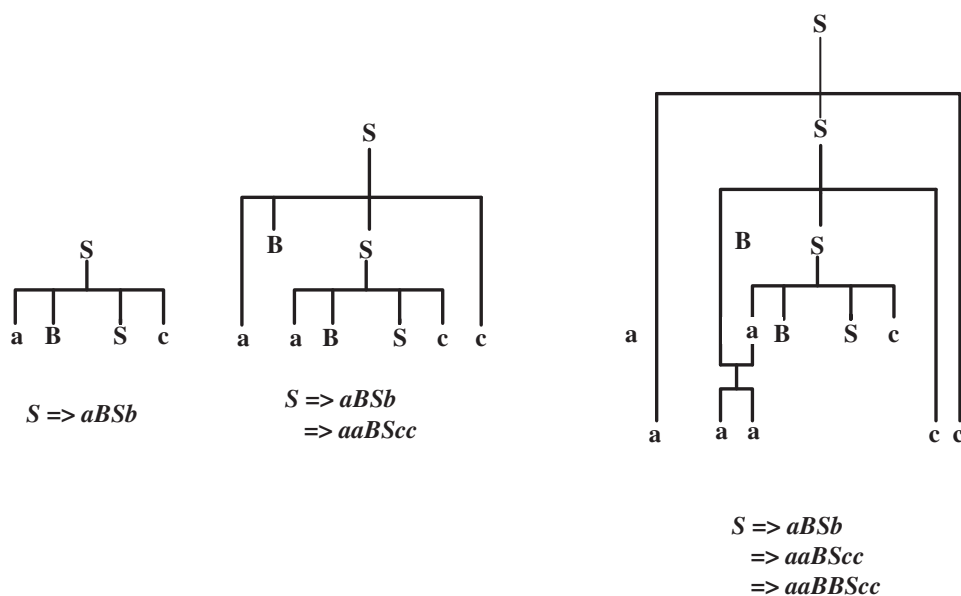
**Παράδειγμα 3.1.14.** Στο σχήμα (a) παρουσιάζονται παραδείγματα δένδρων παραγωγής για παραγωγές σύμφωνα με τη γραμματική  $G = \langle N, \Sigma, P, S \rangle$  με  $N = \{S\}$ ,  $\Sigma = \{a, b\}$  και  $P = \{S \rightarrow aSb, S \rightarrow \epsilon\}$ . Στο σχήμα (b) δίνονται γραφήματα παραγωγής για παραγωγές στη γραμματική  $G = \langle N, \Sigma, P, S \rangle$  με  $N = \{S, B\}$ ,  $\Sigma = \{a, b, c\}$  και  $P = \{S \rightarrow aBSc, S \rightarrow abc, S \rightarrow \epsilon, Ba \rightarrow aB, Bb \rightarrow bb\}$ .



### 3.1.2 Αριστερότερες Παραγωγές (Leftmost derivations)

Μια παραγωγή  $g_0 \Rightarrow \dots \Rightarrow g_n$  λέγεται αριστερότερη (leftmost derivation) αν το  $a_1$  αντικαθίσταται πριν το  $a_2$  σε μια παραγωγή και ισχύουν οι δύο παρακάτω συνθήκες. Το  $a_1$  εμφανίζεται αριστερά του  $a_2$  στη  $g_i$ ,  $0 \leq i < n$ . Τα  $a_1$  και  $a_2$  αντικαθίστανται στην παραγωγή με βάση κανόνες της μορφής  $a_1 \rightarrow b_1$  και  $a_2 \rightarrow b_2$ , αντίστοιχα.

**Παράδειγμα 3.1.15.** Το γράφημα παραγωγής που παρουσιάζεται στο παραπάνω σχήμα δείχνει τη σειρά με την οποία χρησιμοποιούνται οι κανόνες στην παραγωγή της συμβολοσειράς  $a_3b_3c_3$  στο προηγούμενο παράδειγμα. Η αντικατάσταση  $a_1 = aB$  που γίνεται στο βήμα 7 της παραγωγής είναι στην ίδια προτασιακή μορφή με την αντικατάσταση  $a_2 = Bb$  που πραγματοποιείται στο βήμα 6. Η παραγωγή δεν είναι αριστερότερη γιατί το  $a_1$  εμφανίζεται στο αριστερό μέρος του  $a_2$  αλλά αντικαθίσταται μετά το  $a_2$ . Αντίθετα, η ακόλουθη παραγωγή της  $a_3b_3c_3$  με βάση τη  $G$  είναι αριστερότερη.



Η σειρά χρησιμοποίησης των κανόνων είναι ίδια με αυτή που περιγράφεται στο παραπάνω σχήμα με τη μόνη διαφορά ότι οι δείκτες 6 και 7 έχουν αλλάξει θέσεις μεταξύ τους.

$$\begin{aligned}
 \underline{S} &\Rightarrow \overline{aBS}c \\
 &\Rightarrow \underline{aB}\overline{aBS}cc \\
 &\Rightarrow \overline{aaB}\underline{BS}cc \\
 &\Rightarrow \underline{aaB}\underline{B}\overline{aB}cc \\
 &\Rightarrow \underline{aaB}\underline{aB}cc \\
 &\Rightarrow \overline{aaaB}\underline{B}bcc \\
 &\Rightarrow \underline{aaaB}\underline{b}bcc \\
 &\Rightarrow \underline{aaaB}\underline{bb}cc
 \end{aligned}$$

### 3.1.3 Ιεραρχική Δόμηση Γραμματικών

Οι ακόλουθες κλάσεις γραμματικών προκύπτουν με βαθμιαία αύξηση των περιορισμών στους οποίους θα πρέπει να υπακούουν οι κανόνες τους. Μια γραμματική Τύπου 1 (Type 1 grammar) είναι μια γραμματική τύπου 0,  $\langle N, S, P, S \rangle$  που ικανοποιεί τις ακόλουθες δύο συνθήκες. Για κάθε κανόνα  $a \rightarrow b$  στο  $P$  που δεν είναι της μορφής  $S \rightarrow \epsilon$ , ισχύει  $|a| \leq |b|$ . Αν ο κανόνας  $S \rightarrow \epsilon$  ανήκει στο  $P$ , τότε το  $S$  δεν εμφανίζεται στο δεξιότερο μέρος κάποιου κανόνα.

Μια γλώσσα χαρακτηρίζεται ως γλώσσα τύπου 1 αν υπάρχει γραμματική τύπου 1 που παράγει τη γλώσσα αυτή.



πρόκειται για γραμματική  $\langle N, \Sigma, P, S \rangle$  τύπου 2 στην οποία για κάθε κανόνα  $a \rightarrow b$ , που δεν είναι της μορφής  $S \rightarrow \epsilon$ , ισχύει μια από τις ακόλουθες συνθήκες. Το  $b$  είναι ένα τερματικό σύμβολο. Το  $b$  είναι τερματικό σύμβολο που ακολουθείται από ένα μη τερματικό σύμβολο.

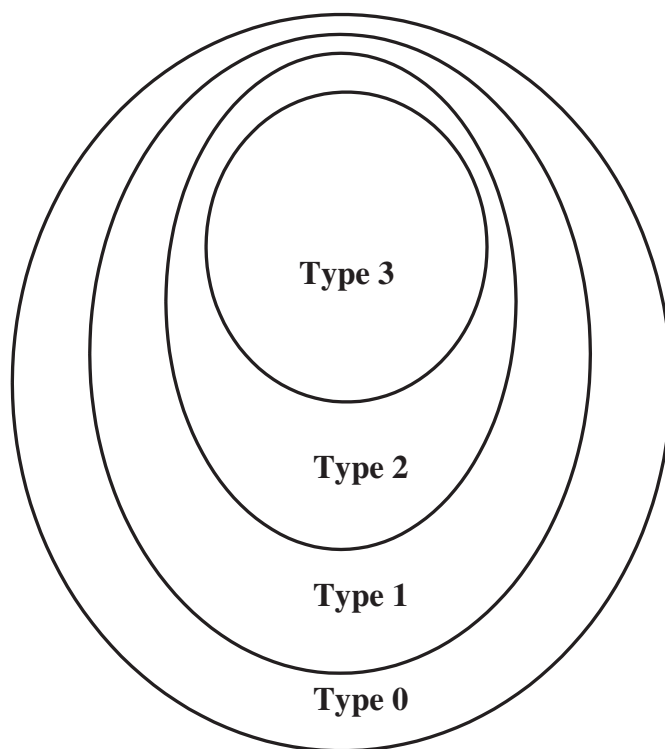
Μια γλώσσα που παράγεται από γραμματική τύπου 3 λέγεται γλώσσα τύπου 3.

**Παράδειγμα 3.1.18.** Έστω η γραμματική  $G = \langle N, \Sigma, P, S \rangle$ , με τους ακόλουθους κανόνες.

$$\begin{aligned} S &\rightarrow \epsilon \\ &\rightarrow aA \\ &\rightarrow bB \\ &\rightarrow b \\ A &\rightarrow bB \\ &\rightarrow b \\ B &\rightarrow aA \\ &\rightarrow bB \\ &\rightarrow b \end{aligned}$$

Η παραπάνω γραμματική είναι τύπου 3. Πρόσθεση στη γραμματική αυτή κανόνων της μορφής  $A \rightarrow Ba$  ή της μορφής  $B \rightarrow bb$ , θα οδηγούσαν σε μια γραμματική που δεν θα ήταν τύπου 3. Το σχήμα που ακολουθεί δείχνει την ιεραρχική δόμηση των διαφόρων τύπων γραμματικών.

Οι γλώσσες χωρίς συμφραζόμενα (context free languages) αποτελούν ένα υπερσύνολο των κανονικών γλωσσών (regular languages): αν είναι δυνατό να προσδιοριστεί μια γλώσσα μέσω μιας κανονικής έκφρασης, τότε η γλώσσα αυτή μπορεί να εκφραστεί και με μια γραμματική χωρίς συμφραζόμενα. Επιπλέον με χρήση γραμματικών χωρίς συμφραζόμενα μπορούν να προσδιοριστούν και γλώσσες που δεν είναι κανονικές. Οι γλώσσες χωρίς συμφραζόμενα χαρακτηρίζονται από ισορροπία (balance) και εμφώλευση (nesting). Για παράδειγμα οι αριθμητικές εκφράσεις περιέχουν ίδιο αριθμό παρενθέσεων (άνοιγμα-κλείσιμο παρένθεσης). Εντολές γλωσσών υψηλού επιπέδου όπως repeat ... until επιτρέπουν εμφώλευση (nesting) και επιπλέον παρουσιάζουν ισορροπία (για κάθε repeat υπάρχει ένα until που του αντιστοιχεί). Η επέκταση των κανονικών γλωσσών στις γλώσσες χωρίς συμφραζόμενα προϋποθέτει την πρόσθεση αναδρομικών κλήσεων κανονικών εκφράσεων. Μια γραμματική χωρίς συμφραζόμενα περιέχει δύο ειδών σύμβολα: τερματικά και μη τερματικά. Ως τερματικά σύμβολα θεωρούνται ανεξάρτητοι χαρακτήρες, συμβολοσειρές που παράγονται από τη γραμματική και η κενή συμβολοσειρά. Τα μη τερματικά σύμβολα χρησιμοποιούνται για την κλήση συναρτήσεων και την εφαρμογή ορισμών. Προκειμένου να αναλύσουμε συντακτικά τη συμβολοσειρά  $7+5*(2+1)$ ,



καλούμε αρχικά τη συνάρτηση αρχικού συμβόλου και στη συνέχεια τις συναρτήσεις για, εκφράσεις (expressions), με τον ορισμό  $expression \Rightarrow expression + factor$ . Το σύμβολο “συν” έχει την έννοια ότι ο όρος  $expression$  αντιστοιχεί στο “7” και ο όρος  $factor$  αντιστοιχεί στη συμβολοσειρά “5\*(2+1)”. Στη συνέχεια καλούμε τη συνάρτηση  $expression$  με τον κανόνα

$$expression \rightarrow factor.$$

Οπότε παίρνουμε:

$$\begin{aligned} expression &\Rightarrow expression + factor \\ &\Rightarrow factor + factor. \end{aligned}$$

Το σύμβολο  $\Rightarrow$  δηλώνει την εφαρμογή μιας συνάρτησης αντικατάστασης μη τερματικών συμβόλων. Στη συνέχεια καλούμε τη συνάρτηση  $factor$ , χρησιμοποιώντας τον κανόνα

$$factor \rightarrow term$$

ως εξής:

$$\begin{aligned} expression &\Rightarrow expression + factor \\ &\Rightarrow factor + factor \\ &\Rightarrow term + factor. \end{aligned}$$

Μετά, καλούμε τη συνάρτηση

$$\begin{aligned} term : expression &\Rightarrow expression + factor \\ &\Rightarrow factor + factor \\ &\Rightarrow term + factor \\ &\Rightarrow IntegerConstant + factor. \end{aligned}$$

Και στη συνέχεια, καλούμε τη συνάρτηση IntegerConstant:

$$\begin{aligned} expression &\Rightarrow expression + factor \\ &\Rightarrow factor + factor \\ &\Rightarrow term + factor \\ &\Rightarrow IntegerConstant + factor \\ &\Rightarrow 7 + factor. \end{aligned}$$

Στο σημείο αυτό, τα πρώτα δύο σύμβολα της συμβολοσειράς που έχουμε παράγει ταιριάζουν με τους δύο πρώτους χαρακτήρες της συμβολοσειράς εισόδου, οπότε επικεντρώνουμε την προσοχή μας σε ό,τι ακολουθεί. Καλούμε διαδοχικά τη συνάρτηση factor, και παίρνουμε

$$7 + factor * term$$

και κατόπιν καλούμε τις συναρτήσεις factor, term, και IntegerConstant λαμβάνοντας

$$7 + 5 * term.$$

Όμοια, αναλύουμε τον όρο term σαν “(expression)” και κατά συνέπεια λαμβάνουμε “2 + 1”. Η

πλήρης παραγωγή για τη συμβολοσειρά εισόδου είναι

$$\begin{aligned}
 \text{expression} &\Rightarrow \text{expression} + \text{factor} \\
 &\Rightarrow \text{factor} + \text{factor} \\
 &\Rightarrow \text{term} + \text{factor} \\
 &\Rightarrow \text{IntegerConstant} + \text{factor} \\
 &\Rightarrow 7 + \text{factor} \\
 &\Rightarrow 7 + \text{factor} * \text{term} \\
 &\Rightarrow 7 + \text{term} * \text{term} \\
 &\Rightarrow 7 + \text{IntegerConstant} * \text{term} \\
 &\Rightarrow 7 + 5 * \text{term} \\
 &\Rightarrow 7 + 5 * (\text{expression}) \\
 &\Rightarrow 7 + 5 * (\text{expression} + \text{factor}) \\
 &\Rightarrow 7 + 5 * (\text{factor} + \text{factor}) \\
 &\Rightarrow 7 + 5 * (\text{IntegerConstant} + \text{factor}) \\
 &\Rightarrow 7 + 5 * (2 + \text{factor}) \\
 &\Rightarrow 7 + 5 * (2 + \text{term}) \\
 &\Rightarrow 7 + 5 * (2 + \text{IntegerConstant}) \\
 &\Rightarrow 7 + 5 * (2 + 1).
 \end{aligned}$$

#### 3.1.4 Γραμματικές Χωρίς Συμφραζόμενα

Γραμματική Χωρίς Συμφραζόμενα είναι μια τετράδα  $(V, \Sigma, R, S)$ , όπου:

- $V$ : ένα αλφάβητο,
- $\Sigma$ : το σύνολο των τερματικών συμβόλων, υποσύνολο του  $V$ ,
- $R$ : το σύνολο των κανόνων, υποσύνολο του  $(V - \Sigma) \times V^*$ ,
- $S$ : το αρχικό σύμβολο που είναι στοιχείο του  $V - \Sigma$ .

Τα στοιχεία του  $V - \Sigma$  ονομάζονται μη τερματικά (nonterminals). Για κάθε  $A \in V - \Sigma$  και  $u \in V^*$  γράφουμε  $A \xrightarrow{G} u$  αν  $(A, u) \in R$ . Για όλες τις συμβολοσειρές  $u, v \in V^*$  γράφουμε  $u \xRightarrow{G} v$  αν και μόνον αν υπάρχουν συμβολοσειρές  $x, y, v' \in V^*$  και  $A \in V - \Sigma$  έτσι ώστε  $u = xAy$ ,  $v = xv'y$  και  $A \xrightarrow{G} v'$ . Η γλώσσα που παράγεται από τη γραμματική  $G$  είναι η  $L(G) = \{w \in \Sigma^* : S \xRightarrow{*}_G w\}$  και λέμε ότι η  $G$  παράγει κάθε συμβολοσειρά της  $L(G)$ . Μια γλώσσα  $L$  ονομάζεται γλώσσα χωρίς συμφραζόμενα αν ισούται με  $L(G)$  για κάποια γραμματική  $G$  χωρίς συμφραζόμενα.

### 3.1.5 Κανονικές Γλώσσες και Γραμματικές Χωρίς Συμφραζόμενα

Μια κανονική γραμματική είναι μια γραμματική χωρίς συμφραζόμενα η οποία στο δεξί μέρος κάθε κανόνα περιέχει ένα το πολύ μη τερματικό σύμβολο, που αν υπάρχει θα είναι το τελευταίο σύμβολο της λέξης. Τυπικά, ο παραπάνω ορισμός εκφράζεται ως εξής:

**Ορισμός 3.1.1.** Μια γραμματική χωρίς συμφραζόμενα  $G = (V, \Sigma, R, S)$  είναι κανονική αν και μόνον αν  $R \in (V - \Sigma) \times \Sigma^*((V - \Sigma) \cup \epsilon)$ .

**Θεώρημα 3.1.1.** Μια γλώσσα είναι κανονική αν και μόνον αν μπορεί να παραχθεί από μια κανονική γραμματική.

*Απόδειξη. Ευθύ:* Έστω η  $L$  κανονική. Τότε θα γίνεται δεκτή από ένα πεπερασμένο αυτόματο  $M = (K, \Sigma, \delta, s, F)$ . Έστω  $G$  η κανονική γραμματική  $(V, \Sigma, R, S)$  με  $V = \Sigma \cup K$ ,  $R = \{q \rightarrow aq : \delta(q, a) = p\} \cup \{q \rightarrow \epsilon : q \in F\}$ ,  $S = s$ .

Υποθέτουμε, χωρίς βλάβη της γενικότητας, ότι τα σύνολα  $K$  και  $\Sigma$  είναι ξένα μεταξύ τους, και θα δείξουμε ότι  $L(G) = L(M)$ . Οι κανόνες της  $G$  έχουν σχεδιαστεί για να μιμούνται ακριβώς τις κινήσεις του  $M$ , δηλαδή για κάθε

$$\sigma_1, \dots, \sigma_n \in \Sigma$$

και

$$p_0, \dots, p_n \in K, (p_0, \sigma_1, \dots, \sigma_n) + M(p_1, \sigma_1, \dots, \sigma_n) + M \dots + M(p_n, \epsilon)$$

αν και μόνον αν

$$p_0 \xRightarrow{G} \sigma_1 p \xRightarrow{G} \sigma_1 \sigma_2 p_2 \xRightarrow{G} \dots \xRightarrow{G} \sigma_1 \dots \sigma_n p_n.$$

Αυτό συμβαίνει διότι:  $\delta(q, a \rightarrow p)$  αν και μόνον  $q \rightarrow ap$ . Για να διαπιστώσουμε ότι  $L(M) \subseteq L(G)$  υποθέτουμε ότι  $w \in L(M)$ . Τότε  $(s, w) \vdash_M^* (p, \epsilon)$  για κάποιο  $p \in F$ . Τότε  $s \xRightarrow{*} wp$  και εφόσον  $p \rightarrow \epsilon$  είναι επίσης κανόνας της  $G$  θα ισχύει  $s \xRightarrow{*} w$  και  $w \in L(G)$ . Για να διαπιστώσουμε ότι  $L(G) \subseteq L(M)$  υποθέτουμε ότι  $w \in L(G)$ . Τότε  $S \xRightarrow{*}_G w$  δηλαδή  $s \xRightarrow{*}_G w$ . Ο κανόνας που χρησιμοποιήθηκε στο τελευταίο βήμα της παραγωγής πρέπει να ήταν της μορφής  $p \rightarrow \epsilon$ , για κάποιο  $p \in F$  έτσι ώστε  $S \xRightarrow{*}_G wp \xRightarrow{G} w$ . Όμως τότε  $(s, w) \vdash_M^* (p, \epsilon)$  και  $w \in L(M)$ .

**Αντίστροφο:** Έστω  $G = (V, \Sigma, R, E)$  μια κανονική γραμματική. Ορίζουμε ένα μη ντετερμινιστικό πεπερασμένο αυτόματο  $M$  έτσι ώστε  $L(G) = L(M)$ . Έστω  $M = (K, \Sigma, \Delta, s, F)$  με

$$K = (V - \Sigma) \cup \{f\} \text{ όπου } f \text{ ένα νέο στοιχείο,}$$

$$s = S,$$

$$F = \{f\},$$

$$\Delta =$$

$$\{(A, w, B) : A \rightarrow wB \in R, A, B \in V - \Sigma, w \in \Sigma^*\} \cup \{(A, w, f) : A \rightarrow w \in R, A \in V - \Sigma, w \in \Sigma^*\}.$$



Κι εδώ οι κινήσεις μιμούνται τις παραγωγές, με την έννοια ότι για κάθε  $A_1, \dots, A_n \in V - \Sigma$ ,  $w_1, \dots, w_n \in \Sigma^*$ ,

$$A_1 \xRightarrow{G} w_1 A_2 \xRightarrow{G} w_1 w_2 A_3 \xRightarrow{G} \dots \xRightarrow{G} w_1 w_2 \dots w_{n-1} A_n \xRightarrow{G} w_1 \dots w_n$$

αν και μόνον αν

$$(A_1, w_1 \dots w_n) \vdash_M (A_2, w_2 \dots w_n) \vdash_M \dots \vdash_M (A_n, w_n) \vdash_M (f, \epsilon).$$

Επομένως, αν  $w \in L(G)$  τέτοια ώστε  $w \in \Sigma^*$  και  $S \xRightarrow{*}_G w$ , τότε  $(S, w) \vdash_M^* (f, \epsilon)$  κι έτσι  $w \in L(M)$ . Αντίστροφα, αν  $w \in L(M)$  τέτοια ώστε  $(S, w) \vdash_M^* (f, \epsilon)$ , τότε  $S \xRightarrow{*}_G w$ , έτσι ώστε  $w \in L(G)$ .  $\square$

### 3.2 Αυτόματα στοίβας

Είδαμε ότι η κλάση των κανονικών γλωσσών είναι ακριβώς το σύνολο των γλωσσών που γίνονται αποδεκτές από τα πεπερασμένα αυτόματα DFAs και NFAs. Με δεδομένο ότι κάποιες γλώσσες χωρίς συμφραζόμενα δεν είναι κανονικές, συμπεραίνουμε ότι κάθε κανονική γλώσσα δεν γίνεται κατ' ανάγκη αποδεκτή από κάποιο πεπερασμένο αυτόματο. Για παράδειγμα, προκειμένου η μη κανονική γλώσσα  $\{ww^R : w \text{ ανήκει στο } \{a, b\}^*\}$  να είναι αναγνωρίσιμη από κάποιο πεπερασμένο αυτόματο θα πρέπει με κάποιο τρόπο να προστεθεί στο αυτόματο ένας μηχανισμός ώστε αυτό να έχει τη δυνατότητα να “θυμάται” ό,τι διάβασε στο πρώτο μέρος της συμβολοσειράς για να μπορεί στη συνέχεια να το συγκρίνει με ό,τι υπάρχει στο δεύτερο μέρος της. Ο μηχανισμός αυτός θα μπορούσε να είναι μια δομή με τις λειτουργίες μια στοίβας (stack) που ονομάζεται και “pushdown store”. Αυτόματα που λειτουργούν όπως τα NFAs και έχουν επιπλέον βοηθητικό αποθηκευτικό χώρο λέγονται αυτόματα στοίβας (pushdown automata). Τα αυτόματα στοίβας μπορούν να θεωρηθούν σα γενικεύσεις των NFAs, μιας και τα τελευταία επιτελούν τις ίδιες ακριβώς λειτουργίες με τα αυτόματα στοίβας χωρίς όμως να αλληλεπιδρούν με τη στοίβα τους. Τα αυτόματα στοίβας αποδέχονται οποιαδήποτε γλώσσα χωρίς συμφραζόμενα και κατά συνέπεια είναι πολύ σημαντικά τόσο από μαθηματική όσο και από πρακτική άποψη: από μαθηματική άποψη γιατί συσχετίζουν δύο όψεις της ίδιας τάξης γλωσσών και από πρακτική γιατί θεμελιώνουν τη μελέτη των συντακτικών αναλυτών (syntax analysers).

Η κλάση των γλωσσών χωρίς συμφραζόμενα είναι ακριβώς το σύνολο των γλωσσών που γίνονται δεκτές από τα αυτόματα στοίβας (Pushdown Automata, PDAs). Αν και υπάρχουν δύο τύποι αυτομάτων στοίβας, ντετερμινιστικά και μη ντετερμινιστικά, σε αντίθεση με το γεγονός ότι κάθε NFA μπορεί να μετατραπεί σε ένα ισοδύναμο DFA, τα μη ντετερμινιστικά αυτόματα στοίβας είναι αυστηρά πιο ισχυρά από τα ντετερμινιστικά. Παρόλ' αυτά υπάρχουν γλώσσες χωρίς συμφραζόμενα που δεν γίνονται αποδεκτές από ντετερμινιστικά αυτόματα στοίβας. Επομένως, το φυσικό μοντέλο μηχανής για τις γλώσσες χωρίς συμφραζόμενα είναι μη ντετερμινιστικό και γι' αυτό λέγοντας αυτόματα στοίβας εννοούμε τα μη ντετερμινιστικά.

Για τα αυτόματα στοίβας υιοθετούμε έναν ορισμό με βάση τον οποίο η στοίβα δεν πρέπει να είναι άδεια προκειμένου να λάβει χώρα μια κίνηση. Τα αυτόματα στοίβας αποτελούνται από μια ταινία εισόδου δεδομένων, έναν μη ντετερμινιστικό πεπερασμένο έλεγχο και μια στοίβα.

Δεδομένου ενός συνόλου  $Q$  πιθανώς άπειρου, καλούμε  $P_{fin}(X)$  το σύνολο όλων των πεπερασμένων υποσυνόλων του  $X$ .

**Ορισμός 3.2.1.** Ένα αυτόματο στοίβας είναι μια ακολουθία 7 στοιχείων  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ :

- $Q$ : πεπερασμένο σύνολο καταστάσεων,
- $\Sigma$ : πεπερασμένο αλφάβητο εισόδου,
- $\Gamma$ : πεπερασμένο αλφάβητο με τα σύμβολα της στοίβας,
- $q_0 \in Q$ : η αρχική κατάσταση,
- $Z_0 \in \Gamma$ : το αρχικό σύμβολο της στοίβας,
- $F \subseteq Q$ : το σύνολο των τελικών καταστάσεων,
- $\delta : Q \times (\Sigma \cup \{\epsilon\} \times \Gamma \rightarrow P_{fin}(Q \times \Gamma^*))$ : η συνάρτηση μετάβασης.

Μια μετάβαση του αυτομάτου στοίβας είναι της μορφής  $(q, \gamma) \in \delta(p, \alpha, Z)$ , όπου  $p, q \in Q, Z \in \Gamma$  και  $\alpha \in \Sigma \cup \{\epsilon\}$ . Μια μετάβαση της μορφής  $(q, \gamma) \in \delta(p, \epsilon, Z)$  καλείται  $\epsilon$ -μετάβαση. Ο τρόπος λειτουργίας ενός αυτόματου στοίβας εξηγείται μέσω περιγραφής στιγμιότυπων του, ή συνολικών καταστάσεων που είναι της μορφής  $(p, u, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ . Η ιδέα είναι ότι  $p$  είναι η αρχική κατάσταση,  $u$  είναι το αδιάβαστο τμήμα της εισόδου και  $\alpha$  δείχνει τη στοίβα και ειδικότερα το κορυφαίο σύμβολο της στοίβας. Δεδομένου ενός PDA ορίζουμε τη σχέση  $\vdash_M$  μεταξύ συνολικών καταστάσεων του αυτομάτου. Η σχέση  $\vdash_M$  είναι όμοια με τη σχέση παραγωγής  $\xRightarrow{G}$  που σχετίζεται με τις γραμματικές χωρίς συμφραζόμενα. Ένα αυτόματο στοίβας διαβάζει την ταινία εισόδου, σύμβολο προς σύμβολο, από αριστερά προς τα δεξιά, κάνει κινήσεις που προκαλούν μετάβαση σε κάποια άλλη κατάσταση, ανανεώνει το περιεχόμενο της στοίβας (επηρεάζοντας μόνο το κορυφαίο στοιχείο της) και είτε μετακινεί την κεφαλή ανάγνωσης εισόδου στο επόμενο σύμβολο, είτε δεν την μετακινεί κατά τη διάρκεια μιας  $\epsilon$ -μετάβασης.

**Ορισμός 3.2.2.** Δεδομένου ενός PDA  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  η σχέση  $\vdash_M$  ορίζεται ως εξής:

1. Για κάθε κίνηση  $(q, \gamma) \in \delta(p, \alpha, Z)$ , όπου  $p, q \in Q, Z \in \Gamma$  και  $\alpha \in \Sigma$ , και για κάθε συνολική κατάσταση της μορφής  $(p, \alpha u, Z\alpha)$  είναι:

$$(p, \alpha u, Z\alpha) \vdash_M (q, u, \gamma\alpha).$$

2. Για κάθε κίνηση  $(q, \gamma) \in \delta(p, \epsilon, Z)$ , όπου  $p, q \in Q, Z \in \Gamma$ , και για κάθε συνολική κατάσταση της μορφής  $(p, u, Z\alpha)$  είναι:

$$(p, u, Z\alpha) \vdash_M (q, u, \gamma\alpha).$$

$\vdash_M^+$  είναι η μεταβατική κλειστότητα (transitive closure) της σχέσης  $\vdash_M$  και  $\vdash_M^*$  είναι η ανακλαστική, μεταβατική κλειστότητα της  $\vdash_M$ . Μια μετάβαση της μορφής

$$(p, \alpha u, Z\alpha) \vdash_M (q, u, \alpha)$$

όπου  $\alpha \in \Sigma \cup \{\epsilon\}$  καλείται “pop-κίνηση” (δηλαδή κίνηση κατά την οποία εξάγεται το κορυφαίο στοιχείο της στοίβας). Στην περίπτωση που

$$(p, u, \alpha) \vdash_M^* (q, v, \beta)$$

λέμε ότι έχουμε έναν υπολογισμό (computation).

**Ορισμός 3.2.3.** Δεδομένου ενός PDA  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  ορίζονται οι ακόλουθες γλώσσες:

1.  $T(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_M^* (f, \epsilon, \alpha), \text{ όπου } f \in F, \text{ και } \alpha \in \Gamma^*\}$ . Λέμε ότι η  $T(M)$  είναι μια γλώσσα που γίνεται αποδεκτή από το αυτόματο  $M$  λόγω τελικής κατάστασης.
2.  $N(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_M^* (f, \epsilon, \epsilon), \text{ όπου } q \in Q\}$ . Λέμε ότι η  $T(M)$  είναι μια γλώσσα που γίνεται αποδεκτή από το αυτόματο  $M$  λόγω άδειας στοίβας.
3.  $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_M^* (f, \epsilon, \alpha), \text{ όπου } f \in F\}$ . Λέμε ότι η  $T(M)$  είναι μια γλώσσα που γίνεται αποδεκτή από το αυτόματο  $M$  λόγω τελικής κατάστασης και άδειας στοίβας.

Σε όλες τις περιπτώσεις η είσοδος  $w$  πρέπει να καταναλωθεί πλήρως. Το ακόλουθο λήμμα δείχνει ότι ο τρόπος αποδοχής δεν έχει σημασία για τα αυτόματα στοίβας (ντετερμινιστικά και μη).

**Λήμμα 3.2.1.** Για κάθε γλώσσα  $L$  ισχύουν τα ακόλουθα:

1. Αν  $L = T(M)$  για κάποιο PDA  $M$ , τότε  $L = L(M')$  για κάποιο PDA  $M'$ .
2. Αν  $L = N(M)$  για κάποιο PDA  $M$ , τότε  $L = L(M')$  για κάποιο PDA  $M'$ .
3. Αν  $L = L(M)$  για κάποιο PDA  $M$ , τότε  $L = T(M')$  για κάποιο PDA  $M'$ .
4. Αν  $L = L(M)$  για κάποιο PDA  $M$ , τότε  $L = N(M')$  για κάποιο PDA  $M'$ .

**Παράδειγμα 3.2.1.** Το ακόλουθο αυτόματο στοίβας δέχεται τη γλώσσα

$$L = \{\alpha^n b^n \mid n \geq 1\}$$

λόγω άδειας στοίβας.

$$Q = \{1, 2\}, \Gamma = \{Z_0, \alpha\}$$

$$(1, \alpha) \in \delta(1, \alpha, Z_0),$$

$$(1, \alpha\alpha) \in \delta(1, \alpha, \alpha),$$

$$(2, \epsilon) \in \delta(1, b, \alpha),$$

$$(2, \epsilon) \in \delta(2, b, \alpha).$$

**Παράδειγμα 3.2.2.** Το ακόλουθο αυτόματο στοίβας δέχεται τη γλώσσα

$$L = \{\alpha^n b^n | n \geq 1\}$$

λόγω τελικής κατάστασης και άδειας στοίβας.

$$Q = \{1, 2, 3\}, \Gamma = \{Z_0, A, \alpha\}, F = \{3\}$$

$$(1, A) \in \delta(1, \alpha, Z_0),$$

$$(1, \alpha A) \in \delta(1, \alpha, A),$$

$$(1, \alpha\alpha) \in \delta(1, \alpha, \alpha),$$

$$(2, \epsilon) \in \delta(1, b, \alpha),$$

$$(2, \epsilon) \in \delta(1, b, \alpha),$$

$$(3, \epsilon) \in \delta(1, b, A),$$

$$(3, \epsilon) \in \delta(2, b, A).$$

**Παράδειγμα 3.2.3.** Στη συνέχεια δίνουμε ένα αυτόματο στοίβας που δέχεται τη γλώσσα  $\{0^n 1^n | n \geq 0\}$ . Έστω  $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$  όπου:

$$Q = \{q_1, q_2, q_3, q_4\},$$

$$\Sigma = \{0, 1\},$$

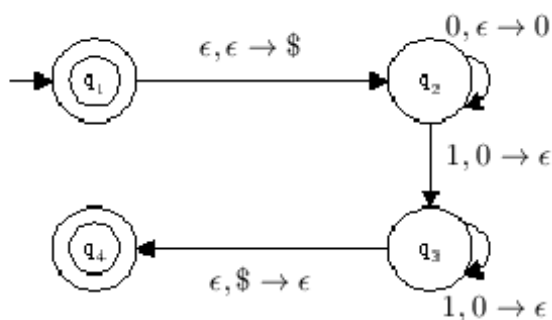
$$\Gamma = \{0, \$\},$$

$$F = \{q_1, q_4\},$$

η συνάρτηση  $\delta$  δίνεται από τον ακόλουθο πίνακα, στον οποίο όπου υπάρχει κενή είσοδος σημαίνει το  $\emptyset$ .

Input	0			1			ε		
Stack	0	\$	e	0	\$	e	0	\$	e
q <sub>1</sub>	{(q <sub>2</sub> , \$)}								
q <sub>2</sub>	{(q <sub>2</sub> , 0)}			{(q <sub>3</sub> , e)}					
q <sub>3</sub>				{(q <sub>3</sub> , e)}			{(q <sub>4</sub> , e)}		
q <sub>4</sub>									

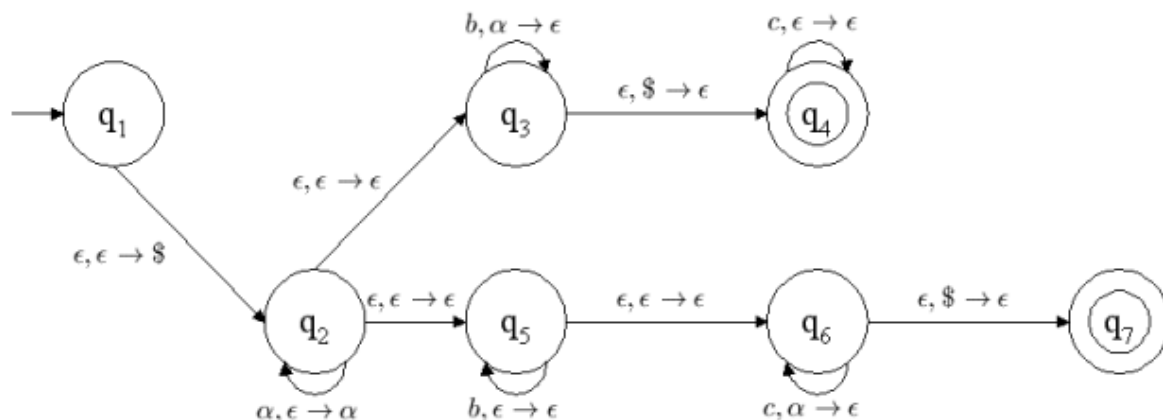
Μπορούμε επίσης να χρησιμοποιήσουμε διαγράμματα καταστάσεων για να περιγράψουμε τη λειτουργία ενός αυτομάτου στοίβας. Γράφουμε “ $\alpha, b \rightarrow c$ ” για να δηλώσουμε ότι όταν το αυτόματο διαβάσει  $\alpha$  στην είσοδο μπορεί να αντικαταστήσει το σύμβολο  $b$  στην κορυφή της στοίβας με το σύμβολο  $c$ . Καθένα από τα  $\alpha, b, c$  μπορεί να είναι  $\epsilon$ . Αν  $\alpha$  είναι  $\epsilon$  το αυτόματο μπορεί να πραγματοποιήσει αυτή τη μετάβαση χωρίς να διαβάσει κάποιο σύμβολο σαν είσοδο. Αν το  $b$  είναι  $\epsilon$  τότε το αυτόματο μπορεί να πραγματοποιήσει μια μετάβαση χωρίς να τροποποιήσει το περιεχόμενο της στοίβας του. Αν το  $c$  είναι  $\epsilon$  τότε το αυτόματο μπορεί να πραγματοποιήσει μια μετάβαση χωρίς να γράψει κάποιο σύμβολο στη στοίβα του. Ο τυπικός ορισμός του αυτόματου στοίβας δεν δηλώνει ρητά κάποιο μηχανισμό αναγνώρισης άδειας στοίβας. Το αυτόματο μπορεί να πραγματοποιήσει αυτή την αναγνώριση αν αρχικά τοποθετηθεί ένα ειδικό σύμβολο \$ στη στοίβα. Αυτό σημαίνει ότι όταν το αυτόματο διαβάσει αυτό το ειδικό σύμβολο η στοίβα είναι άδεια.



**Παράδειγμα 3.2.4.** Στη συνέχεια παρουσιάζουμε ένα αυτόματο στοίβας που αναγνωρίζει τη γλώσσα

$$\{\alpha^i b^j c^k \mid i, j, k \geq 0 \text{ ή } i = k\}.$$

Το αυτόματο δουλεύει διαβάζοντας και περνώντας στη στοίβα τα “α”. Όταν ολοκληρωθεί η ανάγνωση των “α” το αυτόματο τα έχει τοποθετήσει στη στοίβα και μπορεί πλέον να τα συγκρίνει με τα “b” ή τα “c”. Το τρικ είναι ότι χρησιμοποιείται μη ντετερμινισμός αφού το αυτόματο δεν ξέρει αν πρέπει να συγκρίνει τα “α” με τα “b” ή τα “c”. Έτσι το αυτόματο μπορεί να μαντέψει ποια σύγκριση να πραγματοποιήσει, επιλέγοντας ένα από τα πιθανά μονοπάτια, όπως φαίνεται στο σχήμα. Αν ένα από τα δυο μονοπάτια δώσει θετική απάντηση στη σύγκριση το αυτόματο αποδέχεται.



**Παράδειγμα 3.2.5.** Δίνουμε τώρα ένα αυτόματο στοίβας που να αναγνωρίζει τη γλώσσα

$$\{ww^R \mid w \in \{0, 1\}^*\}.$$

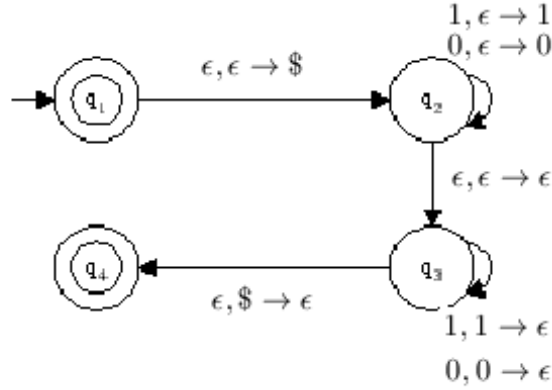
Αρχικά τοποθετεί στη στοίβα όλα τα σύμβολα που διαβάζει από την είσοδο. Σε κάθε σημείο το αυτόματο μαντεύει μη ντετερμινιστικά τη μέση της συμβολοσειράς και αλλάζει κατάσταση αρχίζοντας να εξάγει σύμβολα από τη στοίβα συγκρίνοντάς τα με αυτά που διαβάζει από την είσοδο. Αν πάντα το αποτέλεσμα της σύγκρισης είναι θετικό, και η στοίβα αδειάζει όταν έχει τελειώσει και η είσοδος το αυτόματο δέχεται. Το αυτόματο δίνεται σχηματικά παρακάτω.

**Ορισμός 3.2.4.** Ένα PDA

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

είναι ντετερμινιστικό, αν και μόνον αν κάποια από τις ακόλουθες συνθήκες ισχύει για κάθε  $(p, Z) \in Q \times \Gamma$ :

1.  $|\delta(p, \alpha, Z)| = 1$  για κάθε  $\alpha \in \Sigma$  και  $\delta(p, \epsilon, Z) = \emptyset$ ,
2.  $|\delta(p, \alpha, Z)| = \emptyset$  για κάθε  $\alpha \in \Sigma$  και  $\delta(p, \epsilon, Z) = 1$ .



Ένα ντετερμινιστικό αυτόματο στοίβας λειτουργεί σε πραγματικό χρόνο αν και μόνον αν δεν έχει  $\epsilon$ -μεταβάσεις.

Στη συνέχεια δείχνουμε ότι κάθε γλώσσα χωρίς συμφραζόμενα γίνεται δεκτή από κάποιο αυτόματο στοίβας.

### 3.3 Ισοδυναμία αυτομάτων στοίβας με γραμματικές χωρίς συμφραζόμενα

Δείχνουμε πώς μπορεί ένα Αυτόματο Στοίβας να κατασκευαστεί από μια Γραμματική Χωρίς Συμφραζόμενα. Με δεδομένη μια γραμματική χωρίς συμφραζόμενα  $G = (V, \Sigma, P, S)$  ορίζουμε ένα αυτόματο στοίβας με μια κατάσταση ως εξής:

$$Q = \{q_0\}, \Gamma = V, q_0 = S, Z_0 = S, F = \emptyset.$$

Για κάθε κανόνα  $(A \rightarrow \alpha) \in P$  υπάρχει μετάβαση

$$(q_0, \alpha) \in \delta(q_0, \epsilon, A).$$

Για κάθε  $\alpha \in \Sigma$  υπάρχει μετάβαση

$$(q_0, \epsilon) \in \delta(q_0, \alpha, \alpha).$$

Η διαίσθηση είναι ότι ένας υπολογισμός του αυτομάτου  $M$  μιμείται μια αριστερότερη παραγωγή της γραμματικής  $G$ .

**Λήμμα 3.3.1.** Δεδομένης κάθε γραμματικής χωρίς συμφραζόμενα  $G = (V, \Sigma, P, S)$ , το αυτόματο στοίβας  $M$  που μόλις περιγράψαμε δέχεται τη γλώσσα  $L(G)$  λόγω άδειας στοίβας, δηλαδή  $L(G) = N(M)$ .

Απόδειξη. Οι ακόλουθοι δύο ισχυρισμοί αποδεικνύονται με επαγωγή.

**Πρόταση 3.3.1.** Για κάθε  $u, v \in \Sigma^*$  και για κάθε  $\alpha \in NV^* \cup \{\epsilon\}$ , αν είναι  $S \xRightarrow{lm}^* u\alpha$  τότε

$$(q_0, uv, S) \vdash^* (q_0, v, \alpha).$$

**Πρόταση 3.3.2.** Για κάθε  $u, v \in \Sigma^*$  και για κάθε  $\alpha \in V^*$ , αν είναι  $(q_0, uv, S) \vdash^* (q_0, v, \alpha)$  τότε

$$S \xRightarrow{lm}^* u\alpha.$$

□

Η κατασκευή μιας γραμματικής χωρίς συμφραζόμενα από ένα αυτόματο στοίβας αν και δεν είναι δύσκολη είναι μια περίπλοκη διαδικασία. Η κατασκευή απλουστεύεται αν πρώτα μετατρέψουμε το αυτόματο στοίβας σε ένα ισοδύναμο τέτοιο ώστε για κάθε κίνηση  $(q, \gamma) \in \delta(p, \alpha, Z)$  (όπου  $\alpha \in \Sigma \cup \{\epsilon\}$ ) είναι  $|\gamma| \leq 2$ .

**Λήμμα 3.3.2.** Δεδομένου ενός αυτομάτου στοίβας

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

μπορούμε να κατασκευάσουμε ένα άλλο αυτόματο στοίβας

$$M' = (Q', \Sigma, \Gamma', \delta', q'_0, Z'_0, F')$$

τέτοιο ώστε  $L(M) = L(M')$  και ισχύουν οι ακόλουθες συνθήκες:

1. Δεν υπάρχει ένα-προς-ένα αντιστοιχία ανάμεσα στους υπολογισμούς των αυτομάτων  $M$  και  $M'$  που οδηγούν σε αποδοχή.
2. Αν το αυτόματο  $M$  δεν έχει  $\epsilon$ -μεταβάσεις, τότε ούτε το  $M'$  δεν έχει.
3. Για κάθε  $p \in Q'$ , κάθε  $\alpha \in \Sigma \cup \{\epsilon\}$  και κάθε  $Z \in \Gamma'$ , αν  $(q, \gamma) \in \delta'(p, \alpha, Z)$  τότε  $q \neq q'_0$  και  $|\gamma| \leq 2$ .

Το κρίσιμο σημείο της κατασκευής είναι ότι οι υπολογισμοί που οδηγούν σε αποδοχή ενός αυτομάτου στοίβας που αποδέχεται λόγω άδειας στοίβας και τελικής κατάστασης μπορεί να αποσυντεθεί σε απλούστερους υπολογισμούς της μορφής

$$(p, uv, Z\alpha) \vdash^* (q, v, \alpha),$$

όπου για κάθε ενδιάμεση συνολική κατάσταση  $(s, w, \beta)$  είναι  $\beta = \gamma\alpha$  για κάποιο  $\gamma \neq \epsilon$ .

Τα μη τερματικά σύμβολα της γραμματικής που κατασκευάζεται από το αυτόματο στοίβας  $M$  είναι τριπλέτες της μορφής  $[p, Z, q]$  έτσι ώστε

$$(p, u, Z) \vdash^+ (q, \epsilon, \epsilon)$$



για κάποιο  $u \in \Sigma^*$ .

Δεδομένου ενός αυτόματου στοίβας

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

που ικανοποιεί τις συνθήκες του προηγούμενου λήμματος κατασκευάζουμε μια γραμματική χωρίς συμφραζόμενα  $G = (V, \Sigma, P, S)$  ως εξής:

$$V = \{[p, Z, q] \mid p, q \in Q, Z \in \Gamma\} \cup \Sigma \cup \{S\},$$

όπου  $S$  είναι ένα νέο σύμβολο και οι παραγωγές ορίζονται ως εξής: για κάθε  $p, q \in Q$ , κάθε  $\alpha \in \Sigma \cup \{\epsilon\}$ , και κάθε  $X, Y, Z \in \Gamma$  έχουμε:

1.  $S \rightarrow \epsilon \in P$ , αν  $q_0 \in F$ ,
2.  $S \rightarrow \alpha \in P$ , αν  $(f, \epsilon) \in \delta(q_0, \alpha, Z_0)$  και  $f \in F$ ,
3.  $S \rightarrow \alpha[p, X, f] \in P$ , για κάθε  $f \in F$ , αν  $(p, X) \in \delta(q_0, \alpha, Z_0)$ ,
4.  $S \rightarrow \alpha[p, X, f][s, Y, f] \in P$ , για κάθε  $f \in F$ , για κάθε  $s \in Q$ , αν  $(p, XY) \in \delta(q_0, \alpha, Z_0)$ ,
5.  $[p, Z, q] \rightarrow \alpha \in P$ , αν  $(q, \epsilon) \in \delta(p, \alpha, Z)$  και  $p \neq q_0$ ,
6.  $[p, Z, s] \rightarrow \alpha[q, X, s] \in P$ , για κάθε  $s \in Q$ , αν  $(q, X) \in \delta(p, \alpha, Z)$  και  $p \neq q_0$ ,
7.  $[p, Z, t] \rightarrow \alpha[q, X, s][s, Y, t] \in P$ , για κάθε  $s, t \in Q$ , αν  $(q, XY) \in \delta(p, \alpha, Z)$  και  $p \neq q_0$ ,

**Λήμμα 3.3.3.** Δεδομένου ενός αυτόματου στοίβας

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

που ικανοποιεί τις συνθήκες του προηγούμενου λήμματος η γραμματική χωρίς συμφραζόμενα  $G = (V, \Sigma, P, S)$  που κατασκευάστηκε με τον παραπάνω τρόπο παράγει τη γλώσσα  $L(M)$ , δηλαδή  $L(G) = L(M)$ .

Απόδειξη. Πρέπει να αποδείξουμε ότι

$$L(G) = \{w \in \Sigma^+ \mid (q_0, w, Z_0) \vdash^+ (f, \epsilon, \epsilon), f \in F\}.$$

Για το σκοπό αυτό αποδεικνύουμε με επαγωγή την ακόλουθη πρόταση.

**Πρόταση 3.3.3.** Για κάθε  $p, q \in Q$ , για κάθε  $Z \in \Gamma$ , για κάθε  $k \geq 1$  και για κάθε  $w \in \Sigma^*$ ,

$$[p, Z, q] \xrightarrow[k]{lm} w \text{ αν και μόνον αν } (p, w, Z) \vdash^+ (q, \epsilon, \epsilon).$$

Χρησιμοποιώντας την παραπάνω πρόταση μπορούμε να δείξουμε ότι  $L(G) = L(M)$ . □

### 3.4 Ιδιότητες κλειστότητας των γλωσσών χωρίς συμφραζόμενα

Κάποιες ιδιότητες κλειστότητας των γλωσσών χωρίς συμφραζόμενα είναι ανάλογες με εκείνες των κανονικών γλωσσών.

**Θεώρημα 3.4.1.** *Οι γλώσσες χωρίς συμφραζόμενα είναι κλειστές ως προς τις πράξεις ένωση, παράθεση και Kleene\*.*

*Απόδειξη.* Έστω  $G_1 = (V_1, \Sigma_1, R_1, S_1)$  και  $G_2 = (V_2, \Sigma_2, R_2, S_2)$  και χωρίς απώλεια της γενικότητας υποθέτουμε ότι τα  $V_1 - \Sigma_1$  και  $V_2 - \Sigma_2$  είναι ξένα μεταξύ τους.

**Ένωση:** Έστω  $S$  ένα νέο σύμβολο και έστω

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S).$$

Τότε  $L(G_1) \cup L(G_2) = L(G)$ . Επειδή οι μοναδικοί κανόνες που αφορούν το  $S$  είναι οι  $S \rightarrow S_1$  και  $S \rightarrow S_2$  κι έτσι  $S \xrightarrow{*}_G w$ , όπου  $w \in (\Sigma_1 \cup \Sigma_2)^*$ , αν και μόνον αν είτε  $S_1 \xrightarrow{*}_G w$  ή  $S_2 \xrightarrow{*}_G w$ . Κι εφόσον οι  $G_1$  και  $G_2$  έχουν ξένα μεταξύ τους σύνολα μη τερματικών συμβόλων,  $S_1 \xrightarrow{*}_G w$  αν και μόνον αν  $S_1 \xrightarrow{*}_{G_1} w$ . Ομοίως και για τη γραμματική  $G_2$ .

**Παράθεση:** Η κατασκευή είναι όμοια: η  $L(G_1)L(G_2)$  παράγεται από τη γραμματική

$$(V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}, S).$$

**Kleene\*:** Η  $L(G_1)^*$  παράγεται από την

$$(V_1, \Sigma_1, R_1 \cup \{S_1 \rightarrow \epsilon_1, S_1 \rightarrow S_1 S_1\}, S_1).$$

□

Η κλάση των γλωσσών χωρίς συμφραζόμενα δεν είναι κλειστή ως προς την τομή και τη συμπλήρωση. Είναι όμως κλειστή ως προς την τομή με κανονικά σύνολα.

**Θεώρημα 3.4.2.** *Η τομή μιας γλώσσας χωρίς συμφραζόμενα με μια κανονική γλώσσα είναι γλώσσα χωρίς συμφραζόμενα.*

*Απόδειξη.* Αν η  $L$  είναι μια γλώσσα χωρίς συμφραζόμενα και  $R$  είναι ένα κανονικό σύνολο, τότε  $L = L(M_1)$  για κάποιο αυτόματο στοίβας  $M_1 = (K_1, \Sigma_1, \Gamma_1, \Delta_1, s_1, F_1)$  και  $R = L(M_2)$  για κάποιο ντετερμινιστικό πεπερασμένο αυτόματο  $M_2 = (K_2, \Sigma_2, \delta_2, s_2, F_2)$ . Η ιδέα είναι να συνδυάσουμε αυτές τις δύο μηχανές σε ένα μόνο αυτόματο στοίβας  $M$  που να εκτελεί παράλληλα τους υπολογισμούς των  $M_1$  και  $M_2$  και να δέχεται τη συμβολοσειρά αν και μόνον αν και τα δύο αυτόματα θα τη δέχονταν. Ειδικότερα, έστω  $M = (K, \Sigma, \Gamma, \Delta, s, F)$  όπου

$K = K_1 \times K_2$  το καρτεσιανό γινόμενο των συνόλων καταστάσεων των  $M_1$  και  $M_2$ .

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$\Gamma = \Gamma_1$$

$$s = (s_1, s_2)$$

$$F = F_1 \times F_2$$

και η συνάρτηση μετάβασης  $\Delta$  ορίζεται ως εξής:

$$(((q_1, q_2), u, \beta), ((p_1, p_2), \gamma)) \in \Delta$$

αν και μόνον αν

$$((q_1, u, \beta), (p_1, \gamma)) \in \Delta_1$$

και

$$(q_2, u) \vdash_M^* (p_2, \epsilon).$$

Δηλαδή το  $M$  περνά πρώτα από την κατάσταση  $(q_1, q_2)$  στην κατάσταση  $(p_1, p_2)$  με τον ίδιο τρόπο που το  $M_1$  περνά από την κατάσταση  $q_1$  στην κατάσταση  $p_1$ , κι επιπλέον το  $M$  σημειώνει και τις αλλαγές καταστάσεων του  $M_2$  αφού αυτό διαβάζει την ίδια είσοδο. Η συνθήκη  $(q_2, u) \vdash_M^* (p_2, \epsilon)$  είναι εύκολο να ελεγχθεί αφού ένα ντετερμινιστικό πεπερασμένο αυτόματο διαβάζει ένα μόνο σύμβολο σε κάθε κίνηση. Έτσι στην πράξη θα κατασκευάσαμε το  $M$  διαλέγοντας κάθε φορά μια μετάβαση  $((q_1, u, \beta)(p_1, \gamma))$  του  $M_1$  και μια κατάσταση  $q_2$  του  $M_2$  και προσομοιώνοντας το  $M_2$  για  $|u|$  βήματα με είσοδο  $u$  για να αποφασίσουμε σε ποια κατάσταση  $p_2$  θα φτάσει αφού διαβάσει την είσοδο αυτή.  $\square$

### 3.5 Το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα

Στη συνέχεια παρουσιάζουμε το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα. Συγκεκριμένα θα δείξουμε ότι αν η  $L$  είναι μια γλώσσα χωρίς συμφραζόμενα, τότε συμβολοσειρές της  $L$  με τουλάχιστο  $m$  σύμβολα μπορούν να “αντληθούν” για να παράγουν επιπλέον συμβολοσειρές που ανήκουν στη  $L$ . (Η τιμή του  $m$  εξαρτάται από την εκάστοτε γλώσσα). Το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα αναφέρει τα εξής:

**Θεώρημα 3.5.1.** *Αν  $L$  μια άπειρη γλώσσα χωρίς συμφραζόμενα, τότε υπάρχει θετικός ακέραιος  $m$  τέτοιος ώστε, αν  $w$  είναι μια συμβολοσειρά της  $L$  με μήκος τουλάχιστο  $m$ , τότε  $w = uvxyz$  (για κάποια  $u, v, x, y, z$ )  $|vxy| \neq m$ ,  $|vy| \neq 1$ ,  $uv^i xy^i z \in L$  για όλες τις μη αρνητικές τιμές του  $i$ .*

Απλούστερα το παραπάνω θεώρημα αναφέρει τα ακόλουθα: Αν  $w$  είναι μια αρκετά μεγάλη συμβολοσειρά, τότε υπάρχουν δύο υποσυμβολοσειρές  $v$  και  $x$ , κάπου στην  $w$ . Υπάρχει συμβολοσειρά  $u$  πριν τη  $v$ , συμβολοσειρά  $x$  ανάμεσα στις  $v$  και  $y$ , και συμβολοσειρά  $z$  μετά την  $y$ . Ό,τι παρεμβάλλεται ανάμεσα τις  $v$  και  $y$  δεν θα έχει μεγάλο μέγεθος, αφού  $|vxy|$  μπορεί να είναι το πολύ  $m$ . Οι υποσυμβολοσειρές  $v$  και  $y$  αποκλείεται να είναι και οι δύο κενές (αν και μια από τις δύο μπορεί να είναι). Αν επαναλάβουμε την υποσυμβολοσειρά  $v$  για  $i$  φορές και επαναλάβουμε τη  $y$  για τον ίδιο αριθμό φορών, η συμβολοσειρά που θα προκύψει θα ανήκει επίσης στη γλώσσα  $L$ .

Η απόδειξη του ακόλουθου θεωρήματος χρησιμοποιεί την έννοια των γραμματικών χωρίς συμφραζόμενα και μπορεί να αναχθεί στο αντίστοιχο θεώρημα για πεπερασμένης μνήμης μηχανές (αυτόματα) αν θεωρήσουμε  $u = v = \epsilon$ . Θα παρουσιάσουμε δύο αποδείξεις του Θεωρήματος.

**Θεώρημα 3.5.2** «Pumping lemma) για γλώσσες χωρίς συμφραζόμενα). Για κάθε γλώσσα χωρίς συμφραζόμενα  $L$  υπάρχει μια θετική ακέραια σταθερά  $m$  με την ακόλουθη ιδιότητα. Αν συμβολοσειρά  $w$  ανήκει στη  $L$  και  $|w| \geq m$ , τότε η  $w$  μπορεί να γραφεί σαν  $uvxyz$ , με  $uv^kxy^kz$  να ανήκει στη  $L$  για κάθε  $k \geq 0$ . Επιπλέον,  $|vxy| \leq m$  και  $|vy| > 0$ .

Απόδειξη. Έστω  $G = \langle N, \Sigma, P, S \rangle$  μια γραμματική χωρίς συμφραζόμενα. Έστω  $t$  ο αριθμός των συμβόλων στο δεξί μέρος της μεγαλύτερης παραγωγής (κανόνα) της  $G$ . Χωρίς βλάβη της γενικότητας υποθέτουμε ότι  $t \geq 2$ . Συμβολίζουμε με  $|N|$  τον αριθμό των μη τερματικών συμβόλων στο  $N$  και επιλέγουμε  $m = t^{|N|+1}$ . Έστω συμβολοσειρά  $w$  που ανήκει στη  $L(G)$  τέτοια ώστε  $|w| \geq m$ . Συμβολίζουμε με  $T$  το δένδρο παραγωγής για την  $w$  που είναι το ελάχιστο για την  $w$  όσον αφορά τον αριθμό των κόμβων. Έστω  $\pi$  το μεγαλύτερο μονοπάτι από τη ρίζα προς κάποιο φύλλο του  $T$  και  $n$  ο αριθμός των κόμβων του  $\pi$ . Ο αριθμός φύλλων του  $T$  είναι το πολύ  $t^{n-1}$ . Επομένως,  $t^{n-1} \geq |w|$  και  $|w| \geq m = t^{|N|+1}$  που συνεπάγεται ότι  $n \geq |N| + 2$ . Δηλαδή, το μονοπάτι  $\pi$  πρέπει να έχει δύο κόμβους των οποίων τα αντίστοιχα μη τερματικά σύμβολα, έστω  $E$  και  $F$ , είναι τα ίδια. Σαν αποτέλεσμα, η  $w$  μπορεί να γραφεί σαν  $uvxyz$ , όπου  $vxy$  και  $x$  οι συμβολοσειρές που αντιστοιχούν στα φύλλα των υποδένδρων του  $T$  με ρίζες  $E$  και  $F$ , αντίστοιχα. Βλέπε σχήμα (a).

(α) Ένα δένδρο παραγωγής  $T$  με  $E = F$ . (β) Το δένδρο παραγωγής  $T_k$ .

Έστω  $T_k$  το δένδρο παραγωγής  $T$  τροποποιημένο ώστε το υποδένδρο του  $E$ , εκτός από το υποδένδρο του  $F$ , να επαναλαμβάνεται (pumped)  $k$  φορές (βλέπε σχήμα (b)). Τότε το  $T_k$  είναι επίσης ένα δένδρο παραγωγής στη γραμματική  $G$  για κάθε  $k \geq 0$ . Συνεπάγεται επομένως ότι η συμβολοσειρά  $uv^kxy^kz$ , που αντιστοιχεί στα φύλλα του υποδένδρου  $T_k$ , ανήκει επίσης στη γλώσσα  $L(G)$  για κάθε  $k \geq 0$ . Η επιλογή των  $E$  και  $F$  από τα τελευταία  $|N| + 1$  μη τερματικά σύμβολα στο μονοπάτι  $\pi$  σημαίνει ότι  $|vxy| \leq t^{|N|+1} = m$ , αφού κάθε μονοπάτι από το  $E$  σε ένα φύλλο περιέχει το πολύ  $|N| + 2$  κόμβους. Όμως,  $|vy| > 0$ , γιατί διαφορετικά το  $T_0$  θα ήταν επίσης δένδρο παραγωγής για τη συμβολοσειρά  $w$ , γεγονός που έρχεται σε αντίφαση με την υπόθεση ότι το  $T$  είναι το ελάχιστο δένδρο παραγωγής για τη συμβολοσειρά  $w$ .  $\square$

**Παράδειγμα 3.5.1.** Έστω  $G = \langle N, \Sigma, P, S \rangle$  μια γραμματική χωρίς συμφραζόμενα της οποίας οι

κανόνες φαίνονται παρακάτω.

$$\begin{aligned} S &\rightarrow AA \\ &\rightarrow ab \\ A &\rightarrow SS \\ &\rightarrow a \end{aligned}$$

Χρησιμοποιώντας την ορολογία του προηγούμενου θεωρήματος για τη  $G$ , είναι,  $t = 2$ ,  $|N| = 2$ , και  $m = 8$ . Στη συμβολοσειρά  $w = (ab)^3a(ab)^2$  αντιστοιχεί το δένδρο παραγωγής που φαίνεται στην παρακάτω εικόνα.

Ένα μέγιστο μονοπάτι στο δένδρο, από τη ρίζα σε κάποιο φύλλο, περιέχει 6 κόμβους. Η συμβολοσειρά  $w$  μπορεί να γραφεί με δύο διαφορετικούς τρόπους ώστε να ικανοποιούνται οι περιορισμοί του Pumping Lemma. Σύμφωνα με τον πρώτο τρόπο επιλέγουμε  $u, v, y, z$  τέτοια ώστε:  $u = ab$ ,  $v = \epsilon$ ,  $x = ab$ ,  $y = aba$ ,  $z = abab$ . Σύμφωνα με το δεύτερο τρόπο:  $u = ab$ ,  $v = ab$ ,  $x = ab$ ,  $y = a$ ,  $z = abab$ . Οι συμβολοσειρές  $(ab)^2(aba)^k(ab)^2$  και  $ab(ab)^kaba^k(ab)^2$  είναι οι νέες συμβολοσειρές που ανήκουν στη γλώσσα για  $k \geq 0$ , οι οποίες προκύπτουν από τη  $w$  με βάση το θεώρημα. Τα σχήματα (b) και (c), αντίστοιχα, δείχνουν τα δένδρα παραγωγής  $T_k$  για αυτές τις συμβολοσειρές.

(b) Δένδρο παραγωγής  $T_k$  για τη συμβολοσειρά  $(ab)^2(aba)^k(ab)^2$ .

(c) Δένδρο παραγωγής  $T_k$  για τη συμβολοσειρά  $ab(ab)^kaba^k(ab)^2$ .

Απόδειξη 2: Έστω  $L$  μια γλώσσα χωρίς συμφραζόμενα. Τότε υπάρχει κάποια γραμματική  $G$  χωρίς συμφραζόμενα που παράγει τη γλώσσα  $L$ . Επιπλέον υποθέτουμε ότι: Η  $L$  είναι άπειρη, επομένως δεν υπάρχει άνω φράγμα στο μήκος των συμβολοσειρών που ανήκουν στη  $L$ . Η  $L$  δεν περιέχει την κενή συμβολοσειρά  $\epsilon$ . Η γραμματική  $G$  δεν έχει μοναδιαίες παραγωγές ή  $\epsilon$ -παραγωγές. Υπάρχει πεπερασμένος αριθμός μεταβλητών σε μια γραμματική και οι παραγωγές για κάθε μεταβλητή έχουν πεπερασμένο μήκος. Ο μόνος τρόπος για να παράγει μια γραμματική τυχαία μεγάλες συμβολοσειρές είναι μια ή περισσότερες μεταβλητές να είναι αναδρομικές. Έστω ότι στη γραμματική δεν υπάρχουν αναδρομικές μεταβλητές. Αφού το αρχικό σύμβολο είναι μη αναδρομικό (nonrecursive), θα πρέπει να ορίζεται με χρήση μόνο τερματικών συμβόλων και άλλων μεταβλητών. Όμοια και κάθε μη αναδρομική μεταβλητή. Όταν εξαντληθούν όλες οι μη αναδρομικές μεταβλητές η συμβολοσειρά που θα έχει προκύψει είναι πεπερασμένη. Δηλαδή υπάρχει κάποιο άνω φράγμα στο μέγεθος της συμβολοσειράς που μπορεί να παραχθεί από το αρχικό σύμβολο. Αυτό όμως έρχεται σε αντίθεση με την υπόθεση ότι η γλώσσα είναι άπειρη. Κατά συνέπεια η υπόθεση ότι εν υπάρχουν αναδρομικές μεταβλητές δεν είναι σωστή. Έστω τώρα μια συμβολοσειρά  $X$  που ανήκει στη γλώσσα  $L$ . Αν η  $X$  είναι αρκετά μεγάλη συμβολοσειρά, τότε η παραγωγή της  $X$  θα πρέπει να έχει προκύψει με επαναληπτική χρήση κάποιας μεταβλητής  $A$ . Αφού η  $A$  χρησιμοποιήθηκε στην παραγωγή, η παραγωγή θα πρέπει να ξεκίνησε ως εξής:  $S \Rightarrow uAy$  για κάποιες  $u$  και  $y$ . Αφού η  $A$  χρησιμοποιήθηκε επαναληπτικά, η

παραγωγή θα πρέπει να συνεχίζει ως εξής:  $S \Rightarrow uAy \Rightarrow uvAxy$ . Τελικά, η παραγωγή θα πρέπει να έχει αντικαταστήσει όλες τις μεταβλητές για να δώσει τη συμβολοσειρά  $X$  που ανήκει στη γλώσσα:

$$\begin{aligned} S &\Rightarrow uAy \\ &\Rightarrow uvAxy \\ &\Rightarrow uvwxy = X. \end{aligned}$$

Αυτό σημαίνει ότι τα βήματα  $A \Rightarrow vAx$  και  $A \Rightarrow w$  είναι πιθανά. Επομένως πιθανή είναι και η παραγωγή:  $A \Rightarrow vwx$ . (Τα παραπάνω δεν σημαίνουν ότι η μεταβλητή χρησιμοποιήθηκε επαναληπτικά μια φορά μόνο. Το “\*” του “ $\Rightarrow$ ” μπορεί να καλύψει πολλές φορές χρησιμοποίησης της  $A$ , καθώς και άλλων μεταβλητών). Θα πρέπει να υπάρχει ένα τελευταίο αναδρομικό βήμα. Θεωρούμε τις μεγαλύτερες συμβολοσειρές που μπορούν να παραχθούν από τις  $v$ ,  $w$ , και  $x$  χωρίς τη χρησιμοποίηση αναδρομής (recursion). Τότε υπάρχει αριθμός  $m$  τέτοιος ώστε να είναι  $|vwx| < m$ . Αφού η γραμματική (με βάση την υπόθεση) δεν περιέχει  $\epsilon$ -παραγωγές ούτε μοναδιαίες παραγωγές, κάθε βήμα της παραγωγής είτε εισάγει κάποιο τερματικό σύμβολο ή αυξάνει το μέγεθος του προτασιακού τύπου. Επειδή  $A \Rightarrow vAx$ , συνεπάγεται ότι  $|vx| > 0$ . Τέλος, αφού  $uvAxy$  εμφανίζεται στην παραγωγή, και  $A \Rightarrow vAx$  και  $A \Rightarrow w$  είναι εξίσου πιθανές παραγωγές, συνεπάγεται ότι η συμβολοσειρά  $uv^iwx^i y$  ανήκει επίσης στη γλώσσα  $L$ . Έτσι ολοκληρώθηκε η απόδειξη του Pumping Lemma.  $\square$

### 3.6 Εφαρμογές του Pumping Lemma

Το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα μπορεί να χρησιμοποιηθεί για να αποδείξουμε ότι μια γλώσσα δεν είναι χωρίς συμφραζόμενα. Η μεθοδολογία είναι όμοια με αυτή της χρησιμοποίησης του Pumping Lemma για κανονικές γλώσσες, προκειμένου να αποδείξουμε ότι μια γλώσσα δεν είναι κανονική.

**Παράδειγμα 3.6.1.** Η γλώσσα  $L = \{\alpha^n b^n c^n | n \geq 0\}$  δεν είναι χωρίς συμφραζόμενα. Αφού η  $L$  είναι άπειρη μπορούμε να εφαρμόσουμε το Pumping Lemma. Η απόδειξη προχωράει με εις άτοπον απαγωγή. Αν η  $L$  ήταν χωρίς συμφραζόμενα, θα υπήρχε μια γραμματική  $G$  χωρίς συμφραζόμενα τέτοια ώστε  $L = L(G)$  και κάποια σταθερά  $K > 1$  κατάλληλη για το Pumping Lemma. Έστω  $w = \alpha^K b^K c^K$  και επιλέγουμε το “ $b$ ” να είναι το επαναλαμβανόμενο σύμβολο. Τότε από το Pumping Lemma το τμήμα  $x$  περιέχει μέρος του επαναλαμβανόμενου τμήματος και είτε τα  $u$  και  $v$  είτε τα  $y$  και  $z$  περιέχουν επίσης μέρος του επαναλαμβανόμενου τμήματος. Υποθέτουμε ότι και το  $u$  και το  $v$  περιέχουν κάποια “ $b$ ”. Τότε έχουμε την ακόλουθη κατάσταση:

$$\underbrace{\alpha \cdots \alpha b \cdots b}_u \underbrace{b \cdots b}_v \underbrace{b \cdots b c \cdots c}_{xyz}.$$

Αν θεωρήσουμε τη συμβολοσειρά  $uv^2xy$  ο αριθμός των “α” είναι  $K$  αλλά ο αριθμός των “b” είναι αυστηρά μεγαλύτερος από  $K$  αφού το  $v$  περιέχει τουλάχιστο ένα “b” κι επομένως η συμβολοσειρά  $uv^2xy \notin L$ . Δηλαδή, καταλήγουμε σε άτοπο.

Αν τα  $y$  και  $z$  περιέχουν κάποια “b” επίσης καταλήγουμε σε άτοπο αφού στη συμβολοσειρά  $uv^2xy$  ο αριθμός των “c” είναι  $K$  αλλά ο αριθμός των “b” είναι αυστηρά μεγαλύτερος από  $K$ . Έχοντας καταλήξει σε άτοπο για όλες τις περιπτώσεις καταλήγουμε στο συμπέρασμα ότι η γλώσσα  $L$  δεν είναι χωρίς συμφραζόμενα.

Όπως και στην περίπτωση της εφαρμογής του Pumping Lemma για κανονικές γλώσσες η επιλογή της συμβολοσειράς  $w$  είναι καθοριστικής σημασίας προκειμένου να αποδείξουμε ότι η γλώσσα είναι χωρίς συμφραζόμενα.

#### Το ίδιο παράδειγμα με άλλο τρόπο

Θα δείξουμε ότι η γλώσσα  $L = \{a^i b^i c^i : i > 0\}$  δεν είναι χωρίς συμφραζόμενα. Έστω η γλώσσα  $L$  είναι χωρίς συμφραζόμενα. Αν μια συμβολοσειρά  $X \in L$ , με  $|X| > m$ , συνεπάγεται ότι  $X = uvwxy$ , με  $|vwx| \leq m$ . Επιλέγουμε ένα  $i$  που να είναι μεγαλύτερο του  $m$ . Τότε, όποτε εμφανίζεται το  $uvw$  στη συμβολοσειρά  $a^i b^i c^i$ , δεν μπορεί να περιέχει περισσότερες από δύο διαφορετικές τάξεις γραμμάτων – μπορεί να έχει μόνο  $a$ , μόνο  $b$ , μόνο  $c$ , ή μπορεί να έχει κάποια  $a$  και κάποια  $b$ , ή κάποια  $b$  και κάποια  $c$ . Επομένως η συμβολοσειρά  $vx$  δεν μπορεί να περιέχει πάνω από δύο διαφορετικές τάξεις γραμμάτων, και επιπλέον, εξαιτίας του Pumping Lemma δεν μπορεί ούτε να είναι κενή. Άρα, περιέχει τουλάχιστο ένα γράμμα. Αφού η  $uvwxy$  ανήκει στη  $L$ , και η  $uv^2wx^2y$  πρέπει να ανήκει στη  $L$ . Όμως οι  $v$  και  $x$  δεν γίνεται να είναι και οι δύο κενές, και είναι  $|uv^2wx^2y| > |uvwxy|$ , επομένως υπάρχουν επιπλέον γράμματα. Επιπλέον, η  $vx$  δεν περιέχει γράμματα και από τις τρεις δυνατές τάξεις, άρα δεν είναι δυνατό να έχει προστεθεί ο ίδιος αριθμός αντιγράφων από κάθε γράμμα. Και κατά συνέπεια η  $uv^2wx^2y$  δεν ανήκει στη γλώσσα  $L$ . Έχουμε επομένως φτάσει σε άτοπο. Άρα η αρχική υπόθεση, ότι δηλαδή η γλώσσα  $L$  είναι χωρίς συμφραζόμενα, είναι λάθος.

#### Παράδειγμα 3.6.2. Θεωρούμε τη γλώσσα $L = \{aa^j | a \text{ ανήκει στο } \{a, b\}^*\}$ .

Προκειμένου να αποδείξουμε ότι η γλώσσα  $L$  δεν είναι χωρίς συμφραζόμενα κάνουμε την αντίθετη υπόθεση, και θεωρούμε μια σταθερά  $m$  σύμφωνα με το Θεώρημα για τη γλώσσα  $L$ . Επιλέγουμε  $w = a^m b^m a^m b^m$ . Τότε υπάρχουν  $u, v, x, y, z$  με  $w = uvxyz$  και  $|vxy| \leq m$ ,  $|vy| > 0$ . Τότε  $uv^0xy^0z = uxz = a^i b^j a^s b^t$  όπου είτε  $i \Rightarrow s$  είτε  $j \Rightarrow t$ . Και στις δύο περιπτώσεις, η συμβολοσειρά  $uxz$  δεν ανήκει στη γλώσσα  $L$ . Άρα η  $L$  δεν είναι χωρίς συμφραζόμενα. Αν επιλέξουμε συμβολοσειρά  $w = a^m b a^m b$  μια διάσπαση της  $w$  σε  $uvxyz$  που να ικανοποιούν τις συνθήκες  $|vxy| \leq m$  και  $|vy| > 0$  μπορεί να είναι της μορφής  $v = y = a_j$  με  $b$  στη  $x$  για κάποιο  $j > 0$ . Τότε συμβολοσειρές  $uv^k xy^k z = a^{m+(k-1)j} b a^{m+(k-1)j} b$  ανήκουν επίσης στη γλώσσα  $L$  για κάθε  $k \geq 0$ . Άρα, η συγκεκριμένη επιλογή της συμβολοσειράς  $w$  δεν μας οδηγεί σε άτοπο, όπως εμείς θέλουμε.

#### Παράδειγμα 3.6.3. Η γλώσσα $L = \{a^m b^n c^m d^n | m, n \geq 1\}$ δεν είναι χωρίς συμφραζόμενα. Αφού η

$L$  είναι άπειρη μπορούμε να εφαρμόσουμε το *Pumping Lemma*. Η απόδειξη προχωράει με εις άτοπον απαγωγή. Αν η  $L$  ήταν χωρίς συμφραζόμενα, θα υπήρχε μια γραμματική  $G$  χωρίς συμφραζόμενα τέτοια ώστε  $L = L(G)$  και κάποια σταθερά  $K > 1$  κατάλληλη για το *Pumping Lemma*. Έστω  $w = \alpha^K b^K c^K d^K$  και επιλέγουμε το επαναλαμβανόμενο τμήμα να περιέχει τα σύμβολα “b” και “c”. Τότε από το *Pumping Lemma* το τμήμα  $x$  περιέχει μέρος του επαναλαμβανόμενου τμήματος και είτε τα  $u$  και  $v$  είτε τα  $y$  και  $z$  περιέχουν επίσης μέρος του επαναλαμβανόμενου τμήματος. Υποθέτουμε ότι και το  $u$  και το  $v$  περιέχουν μέρος του επαναλαμβανόμενου τμήματος.

Αν το  $v$  περιέχει κάποια “b”, δεδομένου ότι  $uvxyz \in L$ , το  $v$  πρέπει να περιέχει μόνο “b” γιατί διαφορετικά θα προέκυπτε συμβολοσειρά που δεν θα άνηκε στη γλώσσα  $L$  και θα είχαμε την ακόλουθη κατάσταση:

$$\underbrace{\alpha \cdots ab \cdots b}_{u} \underbrace{b \cdots b}_{v} \underbrace{bc \cdots cd \cdots c}_{xyz}.$$

Επειδή όμως έχουμε υποθέσει ότι  $uvxyz \in L$ , ο μόνος τρόπος να διατηρηθεί ίδιος ο αριθμός των “b” και  $d$  είναι να ισχύει  $y \in d^+$ . Τότε όμως το τμήμα  $vxy$  περιέχει  $c^K$  γεγονός που σημαίνει ότι δεν ισχύει η τέταρτη συνθήκη του *Pumping Lemma*, ότι δηλαδή είναι  $|vxy| > K$ .

Αν το  $v$  περιέχει κάποια “c”, δεδομένου ότι  $uvxyz \in L$ , το  $v$  πρέπει να περιέχει μόνο “c” γιατί διαφορετικά θα προέκυπτε συμβολοσειρά που δεν θα άνηκε στη γλώσσα  $L$  και θα είχαμε την ακόλουθη κατάσταση:

$$\underbrace{\alpha \cdots ab \cdots bc \cdots c}_{u} \underbrace{c \cdots c}_{v} \underbrace{cd \cdots d}_{xyz}.$$

Επειδή όμως έχουμε υποθέσει ότι  $uvxyz \in L$ , και ο αριθμός των “α” παραμένει  $K$  ενώ ο αριθμός των “c” είναι αυστηρά μεγαλύτερος από  $K$ , αυτή η περίπτωση είναι αδύνατη.

Θεωρούμε τώρα την περίπτωση που τα  $y$  και  $z$  περιέχουν μέρος του επαναλαμβανόμενου τμήματος. Αιτιολογώντας όπως και για τις προηγούμενες περιπτώσεις η μόνη πιθανότητα είναι να ισχύει  $v \in \alpha^+$  και  $y \in c^+$ :

$$\underbrace{\alpha \cdots \alpha}_{u} \underbrace{\alpha \cdots \alpha}_{v} \underbrace{\alpha \cdots ab \cdots bc \cdots c}_{x} \underbrace{c \cdots c}_{y} \underbrace{cd \cdots d}_{z}.$$

Τότε όμως το τμήμα  $vxy$  περιέχει  $b^K$  γεγονός που σημαίνει ότι δεν ισχύει η τέταρτη συνθήκη του *Pumping Lemma*, ότι δηλαδή είναι  $|vxy| > K$ .

Το Pumping Lemma για γλώσσες χωρίς συμφραζόμενα μπορεί να γενικευθεί για σχέσεις που μπορούν να γίνουν αποδεκτές από αυτόματα στοίβας (pushdown transducers). Η γενικευμένη αυτή μορφή του Pumping Lemma μπορεί να χρησιμοποιηθεί για τον καθορισμό σχέσεων που δεν γίνονται αποδεκτές από αυτόματα στοίβας.

**Θεώρημα 3.6.1.** Για κάθε σχέση  $R$  που γίνεται αποδεκτή από κάποιο αυτόματο στοίβας, υπάρχει σταθερά  $m$  έτσι ώστε να ισχύουν τα ακόλουθα για κάθε  $(w_1, w_2)$  που ανήκει στην  $R$ .

Αν  $|w_1| + |w_2| \geq m$ , τότε η  $w_1$  μπορεί να γραφεί σαν  $u^1 v^1 x^1 y^1 z^1$  και η  $w_2$  μπορεί να γραφεί σαν  $u^2 v^2 x^2 y^2 z^2$ , όπου  $(u^1 v^1 k x^1 y^1 k z^1, u^2 v^2 k x^2 y^2 k z^2)$  ανήκει επίσης στη  $R$  για κάθε  $k \geq 0$ .



Επιπλέον,  $|v^1x^1y^1| + |v^2x^2y^2| \leq m$  και  $|v^1y^1| + |v^2y^2| > 0$ .

Απόδειξη. Θεωρούμε ένα αυτόματο στοίβας  $M_1$ . Έστω  $M_2$  το αυτόματο στοίβας που προκύπτει από το  $M_1$  με αντικατάσταση κάθε μετάβασης του τύπου  $(q, \alpha, \beta, p, \gamma, \varrho)$  με μια άλλη της μορφής  $(q, [\alpha, \varrho], \beta, p, \gamma)$  αν ισχύει η ανισότητα  $[\alpha, \varrho] \Rightarrow [\epsilon, \epsilon]$ , και με μια μετάβαση του τύπου  $(q, \epsilon, \beta, p, \gamma)$  αν ισχύει η ισότητα  $[\alpha, \varrho] = [\epsilon, \epsilon]$ . Έστω  $h_1$  και  $h_2$  οι συναρτήσεις που ορίζονται ως εξής:  $h_1(\epsilon) = h_2(\epsilon) = \epsilon$ ,  $h_1([\alpha, \varrho]) = \alpha$ ,  $h_2([\alpha, \varrho]) = \varrho$ ,  $h_1([\alpha, \varrho]w) = h_1([\alpha, \varrho])h_1(w)$ , και  $h_2([\alpha, \varrho]w) = h_2([\alpha, \varrho])h_2(w)$ . Από την κατασκευή του το αυτόματο  $M_2$  κωδικοποιεί στην είσοδό του την είσοδο και τις εισόδους και εξόδους του  $M_1$ , δηλαδή ό,τι διαβάζει και δίνει σαν έξοδο το  $M_1$ . Οι τιμές των  $h_1$  και  $h_2$ , αντίστοιχα, καθορίζουν τις τιμές των κωδικοποιημένων εισόδων και εξόδων. Κατά συνέπεια, το  $(w_1, w_2)$  ανήκει στο  $R(M_1)$  αν και μόνον αν η συμβολοσειρά  $w$  ανήκει στη γλώσσα  $L(M_2)$  για κάποια συμβολοσειρά  $w$  τέτοια ώστε  $h_1(w) = w_1$  και  $h_2(w) = w_2$ . Με  $m'$  συμβολίζουμε τη σταθερά που χρησιμοποιείται από το Θεώρημα Άντλησης (Pumping Lemma) για γραμματικές χωρίς συμφραζόμενα, και το εφαρμόζουμε για τη γλώσσα  $L(M_2)$ . Επιλέγουμε  $m = 2m'$ . Θεωρούμε ένα οποιοδήποτε ζεύγος  $(w_1, w_2)$  της σχέσης  $R(M_1)$  τέτοιο ώστε  $|w_1| + |w_2| \geq m$ . Τότε υπάρχει κάποια συμβολοσειρά  $w$  της γλώσσας  $L(M_2)$  τέτοια ώστε  $h_1(w) = w_1$ ,  $h_2(w) = w_2$ , και  $|w| \geq \frac{m}{2} = m'$ . Από το Θεώρημα Άντλησης (Pumping Lemma) για γλώσσες χωρίς συμφραζόμενα, η συμβολοσειρά  $w$  μπορεί να γραφεί σαν  $uvxyz$ , με  $|vxy| \leq m'$ ,  $|vy| > 0$ , και  $uv^kxy^kz$  ανήκει στη γλώσσα  $L(M_2)$  για κάθε  $k \geq 0$ . Το αποτέλεσμα προκύπτει αν επιλέξουμε  $u_1 = h_1(u)$ ,  $u_2 = h_2(u)$ ,  $v_1 = h_1(v)$ ,  $v_2 = h_2(v)$ ,  $x_1 = h_1(x)$ ,  $x_2 = h_2(x)$ ,  $y_1 = h_1(y)$ ,  $y_2 = h_2(y)$ ,  $z_1 = h_1(z)$ , και  $z_2 = h_2(z)$ .  $\square$

**Παράδειγμα 3.6.4.** Έστω  $M_1$  το αυτόματο στοίβας του οποίου το διάγραμμα μεταβάσεων φαίνεται παρακάτω.

Με βάση το παραπάνω θεώρημα,  $M_2$  είναι το αυτόματο στοίβας με το ακόλουθο διάγραμμα καταστάσεων. Ο υπολογισμός του  $M_1$  με συμβολοσειρά εισόδου  $aa\beta\beta aa$  δίνει σαν αποτέλεσμα  $\beta aa$  και αντιστοιχεί στον υπολογισμό του  $M_2$  όταν λάβει σαν είσοδο  $[a, \epsilon][a, \epsilon][b, \epsilon][b, b][a, a][a, a]$ .

### 3.7 Σύνοψη

Μια γραμματική χωρίς συμφραζόμενα (CFG) συνίσταται σε ένα σύνολο κανόνων που επιτρέπουν την αντικατάσταση μεταβλητών με ακολουθίες μεταβλητών και τερματικών συμβόλων. Η γλώσσα μιας γραμματικής είναι το σύνολο των συμβολοσειρών που παράγει. Μια γλώσσα είναι χωρίς συμφραζόμενα αν υπάρχει CFG που να την παράγει. Για κάθε συμβολοσειρά της γλώσσας υπάρχει μια αριστερότερη παραγωγή και ένα δένδρο παραγωγής. Αν αυτά είναι μοναδικά για όλες τις συμβολοσειρές, τότε η γραμματική λέγεται σαφής (unambiguous).

Αυτόματο στοίβας (PDA) είναι ένα FA με μία στοίβα για επιπλέον μνήμη. Στα αντίστοιχα διαγράμματα καταστάσεων ο χαρακτήρας  $\$$  δηλώνει την αρχή της στοίβας. Τα PDA είναι μη ντε-

τερμινιστικά εξ ορισμού.

Υπάρχει αλγόριθμος για τη μετατροπή μιας CFG σε ένα ισοδύναμο PDA: το PDA μαντεύει την αριστερότερη παραγωγή. Ο αλγόριθμος για τη μετατροπή PDA σε CFG είναι πιο πολύπλοκος.

Μια γραμματική είναι κανονική αν κάθε κανόνας της είναι της μορφής  $A \rightarrow bC$  ή  $A \rightarrow a$ . Η ιεραρχία Chomsky περιλαμβάνει επίσης γραμματικές με συμφραζόμενα και μη περιορισμένες γραμματικές.

Υπάρχουν ειδικές μορφές CFG όπως η Chomsky Normal Form, όπου κάθε κανόνας είναι της μορφής  $A \rightarrow BC$  ή  $A \rightarrow c$ . Ο αλγόριθμος για μετατροπή σε Chomsky Normal Form περιλαμβάνει (1) τον καθορισμό όλων των nullable μεταβλητών και την απομάκρυνση όλων των  $\epsilon$ -παραγωγών, (2) την απομάκρυνση όλων των μοναδιαίων κανόνων που δίνουν μεταβλητές, (3) τη διάσπαση μεγάλων παραγωγών, και (4) μεταφορά των τερματικών σε μοναδιαίους κανόνες.

Στην κλάση των γλωσσών χωρίς συμφραζόμενα ισχύει η κλειστότητα ως προς τις πράξεις Kleene αλλά όχι ως προς τη συμπλήρωση και την τομή. Το Pumping Lemma μπορεί να χρησιμοποιηθεί για να δείξουμε ότι μια γλώσσα είναι χωρίς συμφραζόμενα.

## Κεφάλαιο 4

# Μηχανές Turing και υπολογιστική θεωρία

Κατά τη διάρκεια του πρώτου μισού του εικοστού αιώνα, μαθηματικοί όπως οι Kurt Gödel, Alan Turing, Alonzo Church διαπίστωσαν ότι συγκεκριμένα βασικά προβλήματα δεν είναι δυνατό να επιλυθούν από υπολογιστές. Ένα παράδειγμα αυτού του φαινομένου είναι η απάντηση στο κατά πόσον μια μαθηματική πρόταση είναι αληθής. Άμεση συνέπεια της παραπάνω διαπίστωσης υπήρξε η ανάπτυξη της ιδέας των θεωρητικών υπολογιστικών μοντέλων (computational models) τα οποία θα μπορούσαν να διαφωτίσουν την κατασκευή πραγματικών υπολογιστικών μοντέλων.

Η θεωρία της υπολογισιμότητας (Computability Theory) και αυτή της πολυπλοκότητας (Complexity Theory) είναι άμεσα συνδεδεμένες. Στόχος της θεωρίας της πολυπλοκότητας είναι ο διαχωρισμός των προβλημάτων σε εύκολα και δύσκολα, ενώ στόχος της θεωρίας υπολογισιμότητας είναι ο διαχωρισμός των προβλημάτων σε αυτά που επιδέχονται επίλυσης και σε αυτά που δεν είναι εφικτό να επιλυθούν.

Στα 1930 (πριν την εμφάνιση των ψηφιακών υπολογιστών) αρκετοί μαθηματικοί ασχολούνταν με το τι ακριβώς σημαίνει η ύπαρξη δυνατότητας υπολογισμού μιας συνάρτησης. Μια μηχανή Turing πρόκειται για έναν απλούστατο αλλά πολύ ισχυρό υπολογιστή, που εμφανίζει μεγάλη χρησιμότητα στη μελέτη της φύσης και των ορίων της υπολογισιμότητας. Σημαντικά θεωρητικά αποτελέσματα σχετικά με το τι μπορεί να υπολογιστεί και τι όχι, εκφράζονται με χρήση του μοντέλου της μηχανής Turing, που πρώτος το επινόησε ο Alan Turing (1912 – 1954) στη δημοσίευσή του, “On Computable Numbers, with an Application to the Entscheidungs problem”. Πρακτικά του London Mathematical Society, (2nd Series, 42(1936)230 – 65). Όλες οι συναρτήσεις που μπορούν να υπολογιστούν με μηχανές Turing είναι αποτελεσματικά υπολογίσιμες, ενώ η αντίστροφη πρόταση, ότι δηλαδή κάθε υπολογίσιμη συνάρτηση μπορεί να υπολογιστεί από μηχανή Turing αποτελεί τη θέση του Church (Church’s Thesis). Ο Alonzo Church και ο Alan Turing ανεξάρτητα κατέληξαν σε ισοδύναμα συμπεράσματα, που συνοψίζονται στον ακόλουθο ορισμό:

**Θεώρημα 4.0.1 (Church Thesis).** *Μια συνάρτηση είναι υπολογίσιμη αν μπορεί να υπολογιστεί από*

μια μηχανή Turing.

Μια μηχανή Turing πρόκειται για ένα εξαιρετικά απλό είδος υπολογιστή με μεγάλη υπολογιστική ισχύ του οποίου οι λειτουργίες περιορίζονται στη ανάγνωση και εγγραφή συμβόλων σε μια ταινία και την μετακίνηση προς τα δεξιά ή προς τα αριστερά πάνω στην ταινία αυτή. Μια μηχανή Turing μοιάζει με ένα αυτόματο στοίβας μιας και τα δύο έχουν σα βασικό συστατικό ένα μηχανισμό με πεπερασμένο αριθμό καταστάσεων και επιπλέον διαθέτουν αποθηκευτικό χώρο. Η διαφορά τους βρίσκεται μεταξύ άλλων στο είδος του αποθηκευτικού χώρου: τα αυτόματα στοίβας έχουν τη στοίβα (stack) ενώ οι μηχανές Turing διαθέτουν ταινία που θεωρείται απείρου μήκους και προς τις δύο κατευθύνσεις. Η ταινία είναι χωρισμένη σε τετραγωνάκια καθένα από τα οποία περιέχει το πολύ ένα σύμβολο ενός πεπερασμένου αλφαβήτου. Η μηχανή Turing διαθέτει μια κεφαλή ανάγνωσης/εγγραφής η οποία σε κάθε χρονική στιγμή είναι τοποθετημένη σε κάποιο από τα τετραγωνίδια της ταινίας και η μηχανή Turing μπορεί μόνο να διαβάσει ή να γράψει σε ένα τετραγωνίδιο κάθε στιγμή. Μια μηχανή Turing διαθέτει έναν πεπερασμένο αριθμό καταστάσεων και βρίσκεται σε μία από αυτές σε κάθε χρονική στιγμή. Για κάθε τέτοια κατάσταση υπάρχουν αντίστοιχες εντολές που “λένε” στη μηχανή τι να κάνει όταν διαβάζει ένα συγκεκριμένο σύμβολο και σε ποια κατάσταση να μεταβεί στη συνέχεια. Για τη σχηματική περιγραφή των παραπάνω λειτουργιών χρησιμοποιούνται συνήθως διαγράμματα καταστάσεων με λογική παρόμοια με αυτή που είδαμε στα πεπερασμένα αυτόματα. Οι εντολές έχουν τη μορφή:

(παρούσα\_κατάσταση, τρέχον\_σύμβολο, νέα\_κατάσταση, νέο\_σύμβολο, αριστερά-δεξιά).

Η σημασία της εντολής αυτής είναι η ακόλουθη: αν η μηχανή Turing βρίσκεται στην κατάσταση παρούσα\_κατάσταση, και το σύμβολο κάτω από την κεφαλή ανάγνωσης – εγγραφής είναι το τρέχον\_σύμβολο, άλλαξε την εσωτερική κατάσταση της μηχανής σε νέα\_κατάσταση, αντικατάστησε το τρέχον σύμβολο στην ταινία με το νέο\_σύμβολο, και μετακίνησε την κεφαλή κατά ένα τετραγωνίδιο προς την κατεύθυνση που έχει οριστεί (αριστερά ή δεξιά). Αν η μηχανή Turing βρεθεί σε κατάσταση για την οποία δεν υπάρχει αντίστοιχη εντολή, τερματίζει – σταματάει τη λειτουργία της (halt). Το σύνολο των εντολών που υπάρχουν και καθορίζουν τη λειτουργία μιας μηχανής Turing μπορεί να αντιμετωπιστεί σαν ένα πρόγραμμα για τη μηχανή Turing. Κάθε μηχανή Turing  $M$  μπορεί να θεωρηθεί σαν ένας αφηρημένος υπολογιστής που αποτελείται από πεπερασμένο έλεγχο (finite-state control), μια ταινία εισόδου, μια κεφαλή ανάγνωσης,  $m$  βοηθητικές ταινίες για κάποιο  $m > 0$ , μια κεφαλή ανάγνωσης/εγγραφής για κάθε βοηθητική ταινία, μια ταινία εξόδου, και μια κεφαλή εγγραφής.

Υπάρχουν διάφορες παραδοχές που γίνονται σχετικά με μια μηχανή Turing όπως περιγράφηκε παραπάνω. Μια τέτοια παραδοχή αφορά στους αριθμούς για την αναπαράσταση των οποίων χρησιμοποιείται μοναδιαίος συμβολισμός, δηλαδή, ο μη αρνητικός ακέραιος  $n$  αναπαρίσταται με μια συμβολοσειρά  $n + 1$  διαδοχικών άσων “1”ς. Επιπλέον, αν θέλουμε να υπολογίσουμε τη συνάρτηση  $f(n_1, n_2, \dots, n_k)$ , υποθέτουμε ότι αρχικά η ταινία αποτελείται από τα  $n_1, n_2, \dots, n_k$ , κατάλληλα κω-

δικοποιημένα, που χωρίζονται μεταξύ τους με ένα απλό κενό, η κεφαλή βρίσκεται αρχικά πάνω από το αριστερότερο bit του πρώτου στοιχείου και η μηχανή Turing βρίσκεται σε μια αρχική κατάσταση. Η μηχανή Turing έχει υπολογίσει το  $m = f(n_1, n_2, \dots, n_k)$  αν, όταν τερματίζει τη λειτουργία της, η ταινία περιέχει τα  $n_1, n_2, \dots, n_k, m$ , κατάλληλα κωδικοποιημένα και χωρισμένα μεταξύ τους από κενά, ενώ η κεφαλή ανάγνωσης/εγγραφής βρίσκεται ξανά πάνω από το αριστερότερο bit του πρώτου ορίσματος. Έστω, για παράδειγμα, ότι θέλουμε να κατασκευάσουμε μια μηχανή Turing για τον υπολογισμό της συνάρτησης  $m = \text{multiply}(n_1, n_2) = n_1 * n_2$ .

Υποθέτουμε ότι η ταινία εισόδου διαβάσει

\_(1)111\_11111\_-----

τα οποία κωδικοποιούν τους αριθμούς 3 και 4 αντίστοιχα, με χρήση μοναδιαίου συμβολισμού. Η μηχανή Turing θα σταματήσει τη λειτουργία της ανάγνωσης των δεδομένων εισόδου μόλις αυτά εξαντληθούν

\_(1)111\_11111\_111111111111\_-----

η παραπάνω ακολουθία κωδικοποιεί τους αριθμούς 3, 4, και 12 με χρήση μοναδιαίου συμβολισμού. Οι μηχανές Turing μπορεί να είναι ντετερμινιστικές ή μη ντετερμινιστικές. Δεν διαβάζουν κάποια είσοδο όπως τα αυτόματα. Σαν είσοδός τους θεωρούνται σύμβολα τα οποία έχουν τοποθετηθεί στην ταινία τους, πριν την έναρξη της λειτουργίας και οι μηχανές ενδέχεται να διαβάσουν κάποια από αυτά, όλα ή κανένα. Μια μηχανή Turing ενδέχεται να δεχτεί ή να απορρίψει την είσοδό της, ενώ σαν αποτέλεσμα του υπολογισμού (output) θεωρούνται τα εναπομείναντα στην ταινία σύμβολα μετά το τέλος της λειτουργίας της μηχανής.

Μια μηχανή Turing  $M$  είναι μια ακολουθία 7 στοιχείων  $(Q, \Sigma, \Gamma, \delta, q_0, \#, F)$  όπου:

- $Q$ : ένα σύνολο καταστάσεων,
- $\Sigma$ : ένα πεπερασμένο σύνολο συμβόλων που καλείται αλφάβητο εισόδου,
- $\Gamma$ : ένα πεπερασμένο σύνολο συμβόλων που είναι το αλφάβητο της ταινίας,
- $\delta$ : η συνάρτηση μετάβασης, που είναι ένα πεπερασμένο σύνολο από πεντάδες

$$\delta \subseteq Q \times \Gamma \times \Gamma \times \{L, R\} \times Q,$$

έτσι ώστε για κάθε  $(p, \alpha) \in Q \times \Gamma$ , υπάρχει το πολύ μία τριάδα  $(b, m, q) \in \Gamma \times \{L, R\} \times Q$  τέτοια ώστε  $(p, \alpha, b, m, q) \in \delta$ . Κάθε τετράδα  $(p, \alpha, b, m, q) \in \delta$  καλείται εντολή (instruction). Συμβολίζεται και σαν:

$$p, \alpha \rightarrow b, m, q.$$

Το αποτέλεσμα μιας εντολής είναι η μετάβαση από την κατάσταση  $p$  στην κατάσταση  $q$ , η τοποθέτηση (εγγραφή) του συμβόλου  $b$  στη θέση του συμβόλου  $a$  που μόλις διαβάστηκε, και η μετακίνηση της κεφαλής ανάγνωσης/εγγραφής προς τα δεξιά ή προς τα αριστερά, ανάλογα με την ένδειξη του  $m$ .

- $\# \in \Gamma$ : ένα σύμβολο που καλείται κενό (blank),
- $q_0 \in Q$ : η αρχική κατάσταση,
- $F \subseteq Q$ : το σύνολο των τελικών καταστάσεων.

Επειδή μια μηχανή Turing πρέπει να είναι σε θέση να προσδιορίσει την είσοδό της και να καταλαμβάνει τότε ολοκληρώθηκε η διαδικασία επεξεργασίας της, υπάρχουν οι εξής απαιτήσεις: Η ταινία να είναι αρχικά κενή (κάθε σύμβολό της δηλαδή να είναι  $\#$ ) εκτός ίσως από μια πεπερασμένη, συνεχή ακολουθία συμβόλων. Αν υπάρχουν αρχικά μη κενά σύμβολα στην ταινία, η κεφαλή τοποθετείται σε κάποιο από αυτά.

Αξίζει να σημειωθεί ότι η είσοδος (τα μη κενά σύμβολα στην ταινία) δεν περιέχει το  $\#$ . Επίσης, ενδέχεται να υπάρχουν σύμβολα στο  $\Gamma$  που δεν εμφανίζονται σαν μέρος της εισόδου. Η συνάρτηση μετάβασης για μια μηχανή Turing ορίζεται όπως παρακάτω:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}.$$

Αυτό ερμηνεύεται ως εξής: Όταν η μηχανή είναι σε μια δοσμένη κατάσταση ( $Q$ ) και διαβάζει ένα συγκεκριμένο σύνολο ( $\Gamma$ ) από την ταινία, αντικαθιστά το σύμβολο αυτό με κάποιο άλλο ( $\Gamma$ ), μεταβαίνει σε μια άλλη κατάσταση ( $Q$ ), και μετακινεί την κεφαλή κατά ένα τετραγωνίδιο αριστερά ( $L$ ) ή δεξιά ( $R$ ). Ένα στιγμιότυπο ή συνολική κατάσταση (configuration) μιας μηχανής Turing ορίζεται από:

1. την τρέχουσα κατάσταση της μηχανής Turing,
2. τα περιεχόμενα της ταινίας, και
3. τη θέση της κεφαλής στην ταινία.

Τα παραπάνω συνθέτουν μια συμβολοσειρά της μορφής  $x_i \dots x_j q_m x_k \dots x_l$  όπου τα  $x$  είναι τα σύμβολα στην ταινία,  $q_m$  είναι η παρούσα κατάσταση, και η κεφαλή βρίσκεται στο τετραγωνίδιο που περιέχει το σύμβολο  $x_k$  (το σύμβολο που βρίσκεται αμέσως μετά την  $q_m$ ). Μια κίνηση (move) μιας μηχανής Turing μπορεί να απεικονιστεί σαν ένα ζεύγος στιγμιότυπων, μεταξύ των οποίων υπάρχει το σύμβολο “+”, που σημαίνει “παράγει σε ένα βήμα”.

Για παράδειγμα, αν  $\delta(q_5, b) = (q_8, c, R)$  τότε μια πιθανή κίνηση θα ήταν η:  $abbabq_5babb \vdash abbabcq_8abb$

Μια μηχανή Turing τερματίζει τη λειτουργία της (halts) όταν δεν υπάρχουν δυνατές κινήσεις (moves). Αν όταν τερματίσει η μηχανή βρίσκεται σε τελική κατάσταση, τότε δέχεται την είσοδό της (accepts), διαφορετικά την απορρίπτει.

Μια πιο τυπική διατύπωση του παραπάνω είναι η ακόλουθη:

Μια μηχανή Turing  $T = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$  δέχεται μια γλώσσα  $L(M)$ , με

$$L(M) = \{w \in \Sigma^+ : q_0 w \vdash^* x_i q_f x_j \text{ για κάποιο } q_f \in F, x_i, x_j \in \Gamma^*\}.$$

(Σημειώνουμε ότι αυτός ο ορισμός προϋποθέτει ότι η μηχανή Turing ξεκινάει τη λειτουργία της με την κεφαλή να βρίσκεται πάνω από το αριστερότερο σύμβολο). Προηγουμένως αναφέραμε ότι μια μηχανή Turing δέχεται την είσοδό της αν τερματίσει σε τελική κατάσταση. Υπάρχουν δύο περιπτώσεις στις οποίες αυτό ενδέχεται να μη συμβεί:

1. Η μηχανή Turing να τερματίσει σε μη τελική κατάσταση, ή
2. Η μηχανή Turing να μη τερματίσει ποτέ (δηλαδή να περιέλθει σε μια άπειρη ανακύκλωση – infinite loop).

Αν μια μηχανή Turing τερματίσει, η ακολουθία των συνολικών καταστάσεων που οδήγησαν στην κατάσταση τερματισμού καλείται υπολογισμός (computation).

Μια μηχανή Turing ορίζει μια συνάρτηση  $y = f(x)$  για συμβολοσειρές  $x, y \in \Sigma^*$  αν  $q_0 x \vdash^* q_f y$  όπου  $q_f$  είναι μια τελική κατάσταση. Μια συνάρτηση  $f$  είναι Turing – υπολογίσιμη αν υπάρχει μηχανή Turing που να μπορεί να επιτελέσει την παραπάνω λειτουργία. Μηχανές Turing με περισσότερες από μια ταινίες

Παραθέτουμε σχετικά το ακόλουθο θεώρημα, παραλείποντας την απόδειξη.

**Θεώρημα 4.0.2.** Για κάθε μηχανή Turing με  $k$  ταινίες,  $S$ , υπάρχει μια μηχανή Turing με μια ταινία,  $T$ , που μπορεί να αντικαταστήσει την  $S$  με την ακόλουθη έννοια: για κάθε λέξη που ανήκει στο  $\Sigma^*0$ , η  $S$  τερματίζει τη λειτουργία της σε πεπερασμένο αριθμό βημάτων, με είσοδο  $x$  αν και μόνον όταν η  $T$  με την ίδια είσοδο  $x$ , τερματίζει τη λειτουργία της, στην τελευταία ταινία της  $S$  έχει γραφεί ότι και στην ταινία της  $T$ . Επιπλέον, αν η  $S$  τερματίζει μετά από  $N$  βήματα τότε η  $T$  τερματίζει τη λειτουργία της μετά από  $O(N^2)$  βήματα.

## 4.1 Παραδείγματα μηχανών Turing

Προκειμένου να εξηγήσουμε πώς λειτουργεί μια μηχανή Turing, περιγράφουμε στιγμιότυπα της λειτουργίας της. Για τον ορισμό των στιγμιότυπων εκμεταλλευόμαστε το γεγονός ότι  $Q \cap \Gamma = \emptyset$ .

**Ορισμός 4.1.1.** Έστω μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$ . Ένα στιγμιότυπο είναι μια μη κενή συμβολοσειρά στο  $\Gamma^* Q \Gamma^+$ , δηλαδή μια συμβολοσειρά της μορφής

*υραυ,*

όπου  $u, v \in \Gamma^*$ ,  $p \in Q$  και  $\alpha \in \Gamma$ .

Το στιγμιότυπο  $upav$  περιγράφει μια στιγμιαία απεικόνιση της λειτουργίας της μηχανής Turing σύμφωνα με την οποία η παρούσα κατάσταση είναι η  $p$ , στην ταινία της μηχανής υπάρχει η συμβολοσειρά  $uav$  και η κεφαλή ανάγνωσης/εγγραφής δείχνει στο σύμβολο  $a$ . Επομένως, κατά το στιγμιότυπο  $upav$ , η κατάσταση  $p$  βρίσκεται αριστερά του συμβόλου που διαβάζεται από την κεφαλή ανάγνωσης/εγγραφής.

Στη συνέχεια εξηγούμε πώς λειτουργεί μια μηχανή Turing δείχνοντας πώς δουλεύει με στιγμιότυπα.

**Ορισμός 4.1.2.** Έστω μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$ . Η σχέση  $\vdash$  είναι δυαδική και ορίζεται στο σύνολο των στιγμιότυπων ως εξής: Για οποιαδήποτε δύο στιγμιότυπα (τα οποία στο εξής θα συμβολίζουμε  $ID_i$ )  $ID_1$  και  $ID_2$ , είναι  $ID_1 \vdash ID_2$  αν και μόνον αν ισχύει κάποιο από τα ακόλουθα:

1.  $(p, \alpha, b, R, q) \in \delta$  και είτε

$$i \text{ } ID_1 = upacv, c \in \Gamma, \text{ και } ID_2 = ubqcv, \text{ ή}$$

$$ii \text{ } ID_1 = upa \text{ και } ID_2 = ubqB$$

2.  $(p, \alpha, b, L, q) \in \delta$  και είτε

$$i \text{ } ID_1 = uprav, c \in \Gamma, \text{ και } ID_2 = uqcbv, \text{ ή}$$

$$ii \text{ } ID_1 = pra \text{ και } ID_2 = qBbv$$

**Ορισμός 4.1.3.** Έστω μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$ .

Παρατηρούμε από τον παραπάνω ορισμό ότι η ταινία εκτείνεται κατά μία θέση (ένα κενό (blank) μετά το δεξιότερο σύμβολο στην περίπτωση 1.ii και κατά μία θέση πριν το αριστερότερο σύμβολο στην περίπτωση 2.ii). Συμβολίζουμε με  $\vdash^+$  τη μεταβατική κλειστότητα της σχέσης  $\vdash$  και με  $\vdash^*$  την ανακλαστική, μεταβατική κλειστότητα της  $\vdash$ .

Στη συνέχεια εξηγούμε πώς μια μηχανή Turing υπολογίζει μια συνάρτηση

$$f : \underbrace{\Sigma^* \times \dots \times \Sigma^*}_m \rightarrow \Sigma^*.$$

Αφού επιτρέπεται οι συναρτήσεις να έχουν σαν είσοδο  $m \geq 1$  συμβολοσειρές, υποθέτουμε ότι το αλφάβητο  $\Gamma$  περιέχει τον ειδικό χαρακτήρα “,” που δεν ανήκει στο  $\Sigma$  προκειμένου να διαχωρίζονται οι συμβολοσειρές εισόδου. Επίσης, για λόγους απλότητας, υποθέτουμε ότι η μηχανή Turing καθαρίζει την ταινία όποτε φτάνει σε τελική κατάσταση (halts) πριν να γράψει την έξοδο - αποτέλεσμα του υπολογισμού. Για αυτό το λόγο ορίζουμε ένα “κατάλληλο στιγμιότυπο” (που σηματοδοτεί τη λήξη ενός υπολογισμού) καθώς και αρχικά (starting) στιγμιότυπα και στιγμιότυπα διακοπής λειτουργίας (blocking) της μηχανής Turing.



**Ορισμός 4.1.4.** Έστω μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$ , όπου το  $\Gamma$  περιέχει εκτός από το κενό  $B$  και τον ειδικό χαρακτήρα “,” που δεν ανήκει στο  $\Sigma$ . Τότε:

- ένα αρχικό (starting) στιγμιότυπο είναι της μορφής

$$q_0 w_1, w_2, \dots, w_m,$$

όπου  $w_1, \dots, w_m \in \Sigma^*$  και  $m \geq 2$ , ή  $q_0 w$  με  $w \in \Sigma^+$ , ή  $q_0 B$ .

- ένα στιγμιότυπο διακοπής λειτουργίας (blocking ή halting) είναι της μορφής

$$u p a v,$$

τέτοιο ώστε να μην υπάρχουν εντολές  $(p, \alpha, b, m, q) \in \delta$  για κανένα  $(b, m, q) \in \Gamma \times \{L, R\} \times Q$ .

- ένα “κατάλληλο στιγμιότυπο” (proper) είναι ένα στιγμιότυπο διακοπής λειτουργίας της μορφής

$$B^k p w B^l,$$

όπου  $w \in \Sigma^*$  και  $k, l \geq 0$ , με  $l \geq 1$  όταν  $w = \epsilon$ .

Οι ακολουθίες υπολογισμών μιας μηχανής Turing ορίζονται ως εξής:

**Ορισμός 4.1.5.** Έστω μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$ . Ένας υπολογισμός είναι μια άπειρη ή πεπερασμένη ακολουθία στιγμιότυπων

$$ID_0, ID_1, \dots, ID_i, ID_{i+1}, \dots,$$

τέτοια ώστε  $ID_i \vdash ID_{i+1}$  για κάθε  $i \geq 0$ .

Ένας υπολογισμός τερματίζει αν και μόνον αν είναι μια πεπερασμένη ακολουθία στιγμιότυπων τέτοια ώστε

$$ID_0 \vdash^* ID_n,$$

και το στιγμιότυπο  $ID_n$  είναι ένα στιγμιότυπο διακοπής λειτουργίας.

Ένας υπολογισμός αποκλίνει αν είναι μια άπειρη ακολουθία στιγμιότυπων.

Στη συνέχεια εξηγούμε πώς μια μηχανή Turing υπολογίζει μια συνάρτηση.

**Ορισμός 4.1.6.** Μια μηχανή Turing,  $(Q, \Sigma, \Gamma, \delta, q_0, \{L, R\}, F)$  υπολογίζει μια συνάρτηση

$$f : \underbrace{\Sigma^* \times \dots \times \Sigma^*}_m \rightarrow \Sigma^*,$$

αν και μόνον αν ισχύουν οι ακόλουθες συνθήκες:

1. Για κάθε  $w_1, \dots, w_m \in \Sigma^*$ , με δεδομένο ένα αρχικό στιγμιότυπο

$$ID_0 = q_0 w_1, w_2, \dots, w_m,$$

ή  $q_0 w$  με  $w \in \Sigma^+$ , ή  $q_0 B$ , η ακολουθία υπολογισμών της μηχανής Turing  $M$  από το στιγμιότυπο  $ID_0$  τερματίζει σε ένα “κατάλληλο στιγμιότυπο” αν και μόνον αν ορίζεται η τιμή  $f(w_1, \dots, w_m)$ .

2. Αν ορίζεται η τιμή  $f(w_1, \dots, w_m)$ , τότε η μηχανή Turing  $M$  τερματίζει σε ένα “κατάλληλο στιγμιότυπο” της μορφής

$$ID_n = B^k p f(w_1, \dots, w_m) B^h,$$

γεγονός που σημαίνει ότι η μηχανή υπολογίζει τη σωστή τιμή. Μια συνάρτηση  $f$  ορισμένη στο αλφάβητο  $\Sigma^*$  είναι Turing-υπολογίσιμη αν και μόνον αν μπορεί να υπολογιστεί από μια μηχανή Turing,  $M$ .

Αξίζει να σημειωθεί ότι σύμφωνα με τη συνθήκη 1, η μηχανή Turing  $M$  ενδέχεται να τερματίσει σε ένα μη “κατάλληλο στιγμιότυπο” περίπτωση για την οποία η  $f(w_1, \dots, w_m)$  δεν ορίζεται. Το γεγονός αυτό ισοδυναμεί με το ότι δεχόμαστε το αποτέλεσμα ενός υπολογισμού μόνο αν η μηχανή Turing  $M$  έχει καθαρίσει την ταινία της, δηλαδή αν έχει καταλήξει σε ένα “κατάλληλο στιγμιότυπο”. Ειδικότερα, όλοι οι ενδιαμέσοι υπολογισμοί της μηχανής θα πρέπει να έχουν σβηστεί πριν η μηχανή τερματίσει.

**Παράδειγμα 4.1.1.** Έστω η ακόλουθη μηχανή Turing:

$$K = \{q_0, q_1, q_2, q_3\},$$

$$\Sigma = \{\alpha, b\},$$

$$\Gamma = \{\alpha, b, B\},$$

Οι εντολές που ορίζονται από τη συνάρτηση  $\delta$  είναι οι ακόλουθες:

$$q_0, B \rightarrow B, R, q_3$$

$$q_0, \alpha \rightarrow b, R, q_1$$

$$q_0, b \rightarrow \alpha, R, q_1$$

$$q_1, \alpha \rightarrow b, R, q_1$$

$$q_1, b \rightarrow \alpha, R, q_1$$

$$q_1, B \rightarrow B, L, q_2$$

$$q_2, \alpha \rightarrow \alpha, L, q_2$$

$$q_2, b \rightarrow b, L, q_2$$

$$q_2, B \rightarrow B, R, q_3.$$

Εύκολα μπορεί να διαπιστώσει κανείς ότι αυτό που κάνει η παραπάνω μηχανή είναι να εναλλάσσει τα “α” με τα “b” σε μια συμβολοσειρά. Δηλαδή με είσοδο  $w = \alpha\alpha b\alpha b b$  η μηχανή επιστρέφει  $b b b \alpha b \alpha \alpha$ .

**Ορισμός 4.1.7.** Μια γλώσσα είναι *Turing-υπολογίσιμη* (*Turing-decidable*) η απλά υπολογίσιμη αν κάποια μηχανή Turing μπορεί να την υπολογίσει.

**Παράδειγμα 4.1.2.** Στη συνέχεια παρουσιάζουμε μια μηχανή Turing που αναγνωρίζει τη γλώσσα που αποτελείται από όλες τις συμβολοσειρές που περιέχουν “0” και των οποίων το μήκος είναι δύναμη του 2. Δηλαδή υπολογίζει τη γλώσσα  $A = \{0^{2^n} | n \geq 0\}$ .

$M_2 =$  “με είσοδο μια συμβολοσειρά  $w$ ”:

1. Προχώρησε από αριστερά προς τα δεξιά κατά μήκος της ταινίας και διάβασε όλα τα “0”.
2. Αν κατά το πρώτο βήμα η ταινία περιείχε μόνο ένα “0”, κάνε δεκτή τη συμβολοσειρά (*accept*).
3. Αν κατά το πρώτο βήμα η ταινία περιείχε περισσότερα από ένα “0”, και ο αριθμός των “0” ήταν περιττός μην κάνεις δεκτή τη συμβολοσειρά (*reject*).
4. Τοποθέτησε την κεφαλή ανάγνωσης/εγγραφής στο αριστερότερο άκρο της ταινίας.
5. Πήγαινε στο πρώτο βήμα.

Κάθε επανάληψη του πρώτου βήματος ελαττώνει τον αριθμό των “0” στο μισό. Καθώς η μηχανή σαρώνει την ταινία στο πρώτο βήμα, “θυμάται” αν ο αριθμός των “0” που διάβασε είναι άρτιος ή περιττός. Αν είναι περιττός και μεγαλύτερος του 1, ο αριθμός των “0” στη συμβολοσειρά εισόδου δεν μπορεί να ήταν δύναμη του 2. Ως εκ τούτου η μηχανή απορρίπτει την είσοδο. Όμως, αν ο αριθμός των “0” που έχει διαβάσει η μηχανή είναι 1, ο αριθμός τους στην αρχική συμβολοσειρά ήταν δύναμη του 2. Επομένως, σε αυτή την περίπτωση η μηχανή αποδέχεται τη συμβολοσειρά. Στη συνέχεια δίνουμε μια τυπική περιγραφή της μηχανής Turing.

$$M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}}).$$

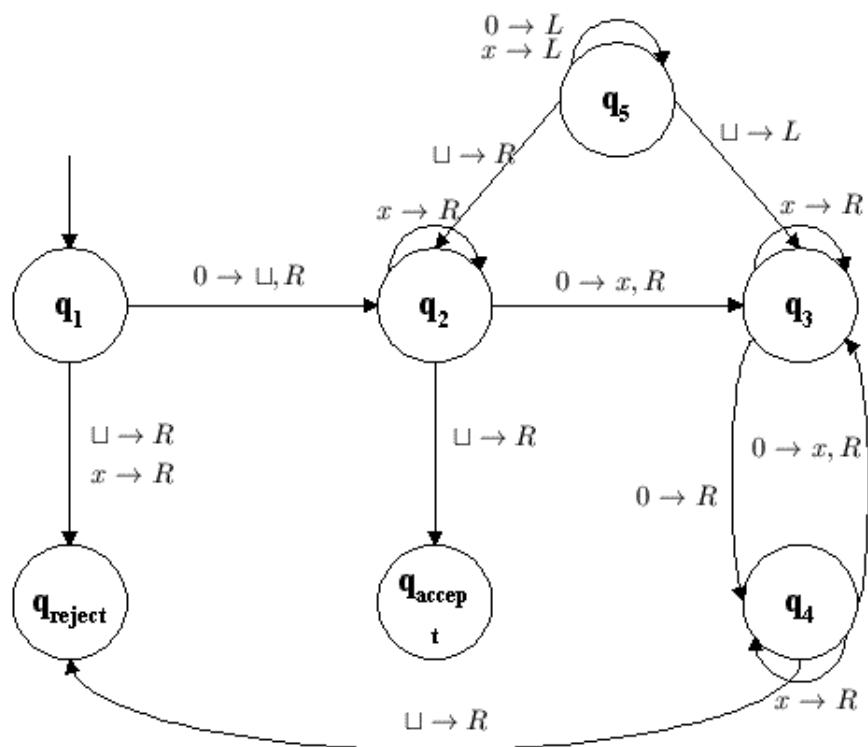
$$Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\},$$

$$\Sigma = \{0\},$$

$$\Gamma = \{0, x, \sqcup\},$$

η συνάρτηση  $\delta$  δίνεται στο διάγραμμα καταστάσεων που ακολουθεί.

οι καταστάσεις: “αρχική”, “αποδοχής”, “απόρριψης” είναι αντίστοιχα οι:  $q_1, q_{\text{accept}}, q_{\text{reject}}$ .



Στο διάγραμμα καταστάσεων η επιγραφή  $0 \rightarrow \sqcup, R$  εμφανίζεται κατά τη μετάβαση από την  $q_1$  στην  $q_2$ . Δηλώνει ότι όταν η μηχανή είναι στην κατάσταση  $q_1$  και η κεφαλή διαβάζει 0, η μηχανή μεταβαίνει στην κατάσταση  $q_2$ , γράφει  $\sqcup$  και μετακινεί την κεφαλή προς τα δεξιά. Για ευκολία χρησιμοποιούμε την επιγραφή  $0 \rightarrow R$  κατά τη μετάβαση από την  $q_3$  στην  $q_4$ , δηλώνοντας ότι η μηχανή μετακινείται προς τα δεξιά όταν διαβάζει 0 στην κατάσταση  $q_3$  αλλά δεν τροποποιεί την ταινία της κι έτσι  $\delta(q_3, 0) = (q_4, 0, R)$ . Η μηχανή ξεκινά γράφοντας ένα κενό σύμβολο πάνω από το αριστερότερο 0 στην ταινία ώστε να μπορεί να βρει το αριστερότερο άκρο της στο βήμα 4. Δίνουμε ένα παράδειγμα λειτουργίας της μηχανής αυτής με είσοδο 0000. Η αρχική συνολική κατάσταση είναι η  $q_1 0000$ . Η ακολουθία των συνολικών καταστάσεων στις οποίες περιέρχεται η μηχανή παρουσιάζονται στο ακόλουθο σχήμα, διαβάζοντας τις στήλες από πάνω προς τα κάτω και τις γραμμές από αριστερά

προς τα δεξιά.

$$\begin{array}{lll}
 q_1 0000 & \sqcup q_5 x 0 x \sqcup & \sqcup x q_5 x x \sqcup \\
 \sqcup q_2 000 & q_5 \sqcup x 0 x \sqcup & \sqcup q_5 x x x \sqcup \\
 \sqcup x q_3 00 & \sqcup q_2 x 0 x \sqcup & q_5 \sqcup x x x \sqcup \\
 \sqcup x 0 q_4 0 & \sqcup x q_2 0 x \sqcup & \sqcup q_2 x x x \sqcup \\
 \sqcup x 0 x q_3 \sqcup & \sqcup x x q_3 x \sqcup & \sqcup x q_2 x x \sqcup \\
 \sqcup x 0 q_5 x \sqcup & \sqcup x x x q_3 \sqcup & \sqcup x x q_2 x \sqcup \\
 \sqcup x q_5 0 x \sqcup & \sqcup x x q_5 x \sqcup & \sqcup x x x q_2 \sqcup \\
 & & \sqcup x x x \sqcup q_{\text{accept}}.
 \end{array}$$

## 4.2 Παραλλαγές του βασικού μοντέλου μηχανής Turing

Οι εναλλακτικές εκδοχές μηχανών Turing συνοψίζονται σε μηχανές Turing με πολλές ταινίες και σε μη ντετερμινιστικές μηχανές Turing. Τόσο το αρχικό μοντέλο όσο και οι εκδοχές της μηχανής Turing έχουν την ίδια υπολογιστική πολυπλοκότητα, δηλαδή αναγνωρίζουν την ίδια κλάση γλωσσών. Στη συνέχεια θα περιγράψουμε αυτές τις εκδοχές και θα παρουσιάσουμε αποδείξεις για την ισοδυναμία υπολογιστικής πολυπλοκότητας. Σαν πρώτο βήμα, διαφοροποιούμε τον τύπο της συνάρτησης μετάβασης. Στον αρχικό ορισμό, η συνάρτηση μετάβασης αναγκάζει την κεφαλή να κινηθεί προς τα δεξιά ή προς τα αριστερά μετά από κάθε βήμα. Υποθέτουμε, λοιπόν, ότι επιτρέπουμε στην κεφαλή να μπορεί να μην κινηθεί καθόλου. Τότε η συνάρτηση μετάβασης θα έχει τη μορφή:  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ . Αυτό το επιπλέον χαρακτηριστικό δεν επιτρέπει στις μηχανές Turing να δέχονται επιπλέον γλώσσες. Μια άλλη παραλλαγή προκύπτει αν αντικαταστήσουμε κάθε μετάβαση κατά την οποία δεν κινείται η κεφαλή με δύο μεταβάσεις, μια προς τα δεξιά και μια (πίσω) προς τα αριστερά. Προκειμένου να δείξουμε ότι δύο παραλλαγές είναι ισοδύναμες αρκεί να δείξουμε ότι η μια μπορεί να εξομοιώσει την άλλη.

### 4.2.1 Μηχανές Turing πολλαπλών ταινιών

Μια μηχανή Turing πολλαπλών ταινιών είναι μια κανονική μηχανή Turing με πολλές ταινίες, κάθε μια από τις οποίες έχει τη δική της κεφαλή ανάγνωσης/εγγραφής. Αρχικά η είσοδος εμφανίζεται στην ταινία με αριθμό 1 ενώ οι υπόλοιπες ταινίες είναι κενές. Η συνάρτηση μετάβασης τροποποιείται ώστε να επιτρέπεται ανάγνωση, εγγραφή και μετακίνηση κεφαλών για όλες τις ταινίες ταυτόχρονα. Τυπικά:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k,$$

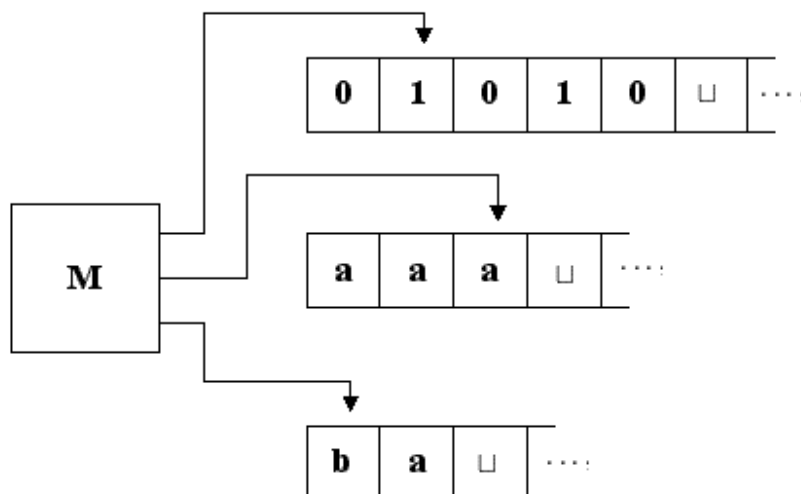
όπου  $k$  είναι το πλήθος των ταινιών. Η έκφραση

$$\delta(q_i, \alpha_1, \dots, \alpha_k) = (q_j, b_1, \dots, b_k, L, R, \dots, L),$$

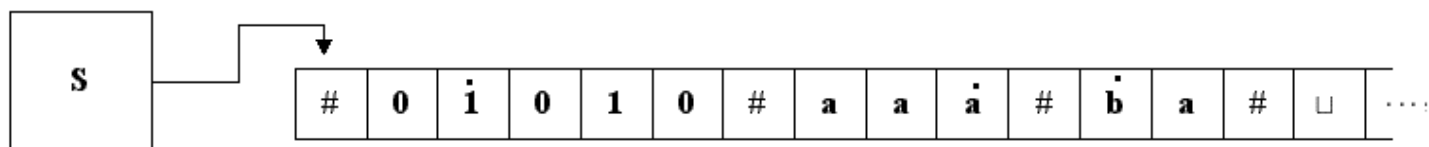
σημαίνει ότι αν η μηχανή βρίσκεται στην κατάσταση  $q_i$  και οι κεφαλές 1 έως  $k$  διαβάζουν τα σύμβολα  $a_1$  έως  $a_k$ , η μηχανή μεταβαίνει στην κατάσταση  $q_j$ , γράφει τα σύμβολα από  $b_1$  έως  $b_k$  και μετακινεί κάθε κεφαλή προς τα αριστερά ή προς τα δεξιά, ανάλογα με το τι έχει καθοριστεί. Αν και εκ πρώτης όψεως οι μηχανές Turing πολλαπλών ταινιών φαίνεται να είναι ισχυρότερες από αυτές με μια ταινία, αποδεικνύεται ότι στην πραγματικότητα οι δύο εκδοχές είναι ισοδύναμες, δηλαδή ότι αναγνωρίζουν το ίδιο σύνολο γλωσσών.

**Θεώρημα 4.2.1.** Κάθε μηχανή Turing με περισσότερες από μια ταινίες είναι ισοδύναμη με μια μηχανή Turing με μια ταινία.

*Απόδειξη.* στη συνέχεια θα δείξουμε πώς μπορούμε να μετατρέψουμε μια μηχανή Turing πολλαπλών ταινιών  $M$  σε μια ισοδύναμη μηχανή Turing με μία ταινία  $S$ . Η ιδέα είναι να εξομοιώσουμε την  $M$  με την  $S$ . Υποθέτουμε ότι η  $M$  έχει  $k$  ταινίες. Τότε η  $S$  εξομοιώνει το αποτέλεσμα των  $k$  ταινιών αποθηκεύοντας την πληροφορία που αυτές διαθέτουν στη δική της μοναδική ταινία. Χρησιμοποιείται το σύμβολο  $\#$  ως διαχωριστικό μεταξύ των περιεχομένων των διαφόρων ταινιών. Επιπλέον, η  $S$  θα πρέπει να “θυμάται” τη θέση της κάθε κεφαλής. Αυτό πετυχαίνεται με την εγγραφή ενός συμβόλου στην ταινία με μια τελεία πάνω του προκειμένου να δηλωθεί το σημείο στο οποίο θα ήταν η κεφαλή ανάγνωσης/εγγραφής αυτής της ταινίας. Τα σύμβολα με τις τελείες θεωρούνται νέα σύμβολα που έχουν προστεθεί στο αλφάβητο της ταινίας. Στο σχήμα που ακολουθεί φαίνεται ο τρόπος με τον οποίο μια ταινία μπορεί να χρησιμοποιηθεί για να εξομοιώσει τρεις ταινίες.



$S =$  με είσοδο  $w = w_1 \dots w_n$ :



1. Αρχικά η μηχανή  $S$  τροποποιεί την ταινία της ώστε να απεικονίζει τις  $k$  ταινίες της  $M$ . Η τροποποιημένη ταινία περιέχει:

$$\# \overset{\bullet}{w_1} \overset{\bullet}{w_2} \dots \overset{\bullet}{w_n} \# \square \# \square \# \dots \#$$

2. Για την εξομοίωση μιας απλής κίνησης, η  $S$  σαρώνει την ταινία της από το πρώτο  $\#$ , που δηλώνει το αριστερότερο άκρο της, μέχρι το  $k+1$ -στό  $\#$ , που δηλώνει το δεξιότερο άκρο της, προκειμένου να καθορίσει τα σύμβολα κάτω από τις υποτιθέμενες κεφαλές. Στη συνέχεια η  $S$  σαρώνει για δεύτερη φορά την ταινία της προκειμένου να ανανεώσει το περιεχόμενο των ταινιών σύμφωνα με τον τρόπο που υποδεικνύει η συνάρτηση μετάβασης της μηχανής  $M$ .
3. Αν σε οποιοδήποτε σημείο η μηχανή  $S$  μετακινεί κάποια από τις υποτιθέμενες κεφαλές προς τα δεξιά πάνω σε ένα  $a\#$ , αυτό σημαίνει ότι η μηχανή  $M$  έχει μετακινήσει την αντίστοιχη κεφαλή στο κενό τμήμα της ταινίας που δεν έχει διαβαστεί. Επομένως η  $S$  γράφει ένα κενό σύμβολο στο σημείο αυτό της ταινίας και μετακινεί κατά μια μονάδα προς τα δεξιά τα υπόλοιπα περιεχόμενά της, αυτά που βρίσκονται μεταξύ αυτού του συμβόλου και του δεξιότερου  $\#$ . Και συνεχίζει την εξομοίωση όπως προηγούμενως.

□

**Πόρισμα 4.2.1.** Μια γλώσσα είναι αναγνωρίσιμη από μια μηχανή Turing αν και μόνον αν είναι αναγνωρίσιμη από μια μηχανή Turing πολλαπλών ταινιών.

*Απόδειξη.* Μια γλώσσα αναγνωρίσιμη από μια μηχανή Turing αναγνωρίζεται από κάθε απλή (με μία ταινία) μηχανή Turing, η οποία αποτελεί ειδική περίπτωση μιας μηχανής Turing πολλαπλών ταινιών. Αυτό αποδεικνύει την μια κατεύθυνση του πορίσματος. Η άλλη συνεπάγεται από το θεώρημα 4.2.1. □

#### 4.2.2 Μη Ντετερμινιστικές Μηχανές Turing

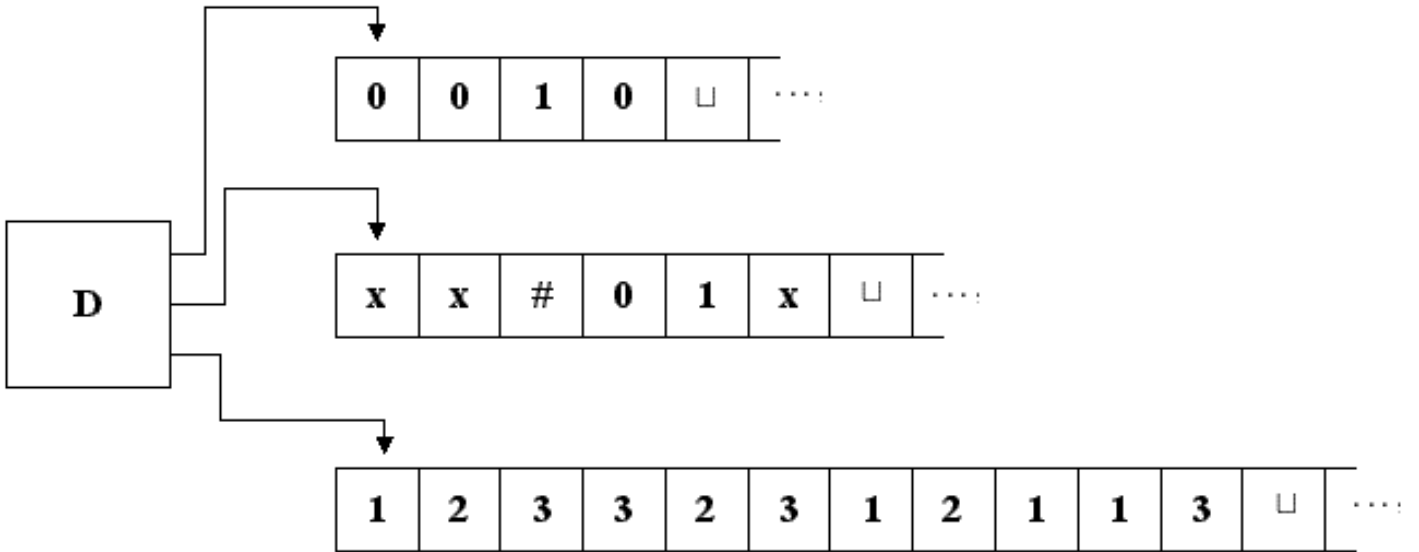
**Θεώρημα 4.2.2.** Για κάθε μη ντετερμινιστική μηχανή Turing υπάρχει μια ισοδύναμη ντετερμινιστική μηχανή Turing.

*Απόδειξη.* Μια μη ντετερμινιστική μηχανή Turing μπορεί σε κάθε σημείο του υπολογισμού της να προχωρήσει προς περισσότερα του ενός μονοπάτια. Η συνάρτηση μετάβασης μια μη ντετερμινιστικής

μηχανής Turing έχει τη μορφή:

$$Q \times \Gamma \rightarrow \Pi(Q \times \Gamma \times \{L, R\}).$$

Ο υπολογιστικός μηχανισμός μιας μη ντετερμινιστικής μηχανής Turing είναι ένα δένδρο του οποίου τα μονοπάτια δηλώνουν τα διαφορετικά υπολογιστικά μονοπάτια που μπορεί να ακολουθήσει η μηχανή. Αν κάποιο μονοπάτι οδηγεί σε κατάσταση αποδοχής, η μηχανή αποδέχεται την είσοδό της. Στη συνέχεια δείχνουμε ότι ο μη ντετερμινισμός δεν επηρεάζει την υπολογιστική ισχύ του μοντέλου της μηχανής Turing. Υποθέτουμε ότι η μηχανή Turing  $D$  που θέλουμε να εξομοιώσουμε έχει τρεις ταινίες. Αυτό όμως, με βάση το θεώρημα 4.2.1 ισοδυναμεί με την ύπαρξη μιας ταινίας. Ο τρόπος με τον οποίο χρησιμοποιεί η μηχανή  $D$  τις ταινίες της απεικονίζεται στο ακόλουθο σχήμα. Η ταινία 1 πάντα περιέχει τη συμβολοσειρά εισόδου και δεν υφίσταται καμία τροποποίηση. Η ταινία 2 διατηρεί ένα αντίγραφο της ταινίας της μη ντετερμινιστικής μηχανής Turing  $N$  για κάποιο μη ντετερμινιστικό υπολογιστικό μονοπάτι. Η ταινία 3 “θυμάται” τη θέση της  $D$  στο μη ντετερμινιστικό υπολογιστικό δένδρο της  $N$ . Μελετάμε αρχικά την αναπαράσταση δεδομένων στην ταινία 3. Κάθε κόμβος στο



δένδρο μπορεί να έχει το πολύ  $b$  παιδιά, όπου  $b$  είναι το μέγεθος του μεγαλύτερου συνόλου πιθανών επιλογών που δίνονται από τη συνάρτηση μετάβασης της  $N$ . Σε κάθε κόμβο του δένδρου αναθέτουμε μια διεύθυνση, που είναι ουσιαστικά μια συμβολοσειρά του αλφαβήτου  $\Sigma_b = \{1, 2, \dots, b\}$ . Αναθέτουμε, έτσι, τη διεύθυνση 231 στον κόμβο που φτάνουμε αν ξεκινήσουμε από τη ρίζα του δένδρου, πάμε στο δεύτερο παιδί του, μετά στο τρίτο παιδί (του δεύτερου παιδιού) και τελικά στο πρώτο παιδί του προηγούμενου κόμβου. Κάθε σύμβολο της συμβολοσειράς δηλώνει την επόμενη επιλογή κατά την εξομοίωση ενός βήματος με ένα μονοπάτι του μη ντετερμινιστικού υπολογισμού της μηχανής  $N$ . Ενδέχεται κάποιο σύμβολο να μην αντιστοιχεί σε καμία επιλογή αν υπάρχουν λίγες επιλογές διαθέσιμες για κάποια συνολική κατάσταση. Στην περίπτωση αυτή η διεύθυνση είναι άκυρη



και δεν αντιστοιχεί σε κανέναν κόμβο. Η ταινία 3 περιέχει μια συμβολοσειρά από το αλφάβητο  $\Sigma_b$ . Δηλώνει το υπολογιστικό μονοπάτι της  $N$  από τη ρίζα στον κόμβο με διεύθυνση τη συγκεκριμένη συμβολοσειρά (εκτός κι αν η τελευταία είναι άκυρη). Η κενή συμβολοσειρά είναι η διεύθυνση της ρίζας του δένδρου. Στη συνέχεια περιγράφουμε τη λειτουργία της μηχανής  $D$ .

1. Αρχικά η ταινία 1 περιέχει την είσοδο  $w$ , ενώ οι ταινίες 2 και 3 είναι κενές.
2. Αντίγραψε την ταινία 1 στη 2.
3. Χρησιμοποίησε την ταινία 2 για την εξομοίωση της  $N$  με είσοδο τη συμβολοσειρά  $w$  για κάποιο μη ντετερμινιστικό υπολογιστικό μονοπάτι της. Πριν από κάθε βήμα της μηχανής  $N$ , δες το επόμενο σύμβολο στην ταινία 3 προκειμένου να καθορίσεις την επιλογή της επόμενης κατάστασης με βάση αυτές που είναι επιτρεπτές από τη συνάρτηση μετάβασης της  $N$ . Αν δεν υπάρχουν σύμβολα στην ταινία 3 ή αν είναι άκυρη η επιλογή, μην ασχολείσαι άλλο με αυτό το μονοπάτι και πήγαινε στο βήμα 4. Αν φτάσεις σε αποδεκτή συνολική κατάσταση, κάνε αποδεκτή την είσοδο.
4. Αντικατάστησε τη συμβολοσειρά στην ταινία 3 με την επόμενη με βάση τη λεξικογραφική διάταξη. Εξομοίωσε το επόμενο υπολογιστικό μονοπάτι της  $N$ , πηγαίνοντας στο δεύτερο βήμα.

□

**Πόρισμα 4.2.2.** *Μια γλώσσα είναι αναγνωρίσιμη από μια μηχανή Turing αν και μόνον αν είναι αναγνωρίσιμη από μια μη ντετερμινιστική μηχανή Turing.*

*Απόδειξη.* Κάθε ντετερμινιστική μηχανή Turing είναι ταυτόχρονα και μη ντετερμινιστική κι επομένως αποδεικνύεται το ευθύ μέρος του πορίσματος. Το αντίστροφο προκύπτει από το θεώρημα 4.2.2. □

Μπορούμε τώρα να τροποποιήσουμε την απόδειξη του θεωρήματος 4.2.2 έτσι ώστε αν η μηχανή  $N$  πάντα τερματίζει σε κάθε μονοπάτι του υπολογισμού της, και η  $D$  επίσης θα τερματίσει.

**Πόρισμα 4.2.3.** *Μια γλώσσα είναι υπολογίσιμη από μια μηχανή Turing αν και μόνον αν είναι αναγνωρίσιμη από μια μη ντετερμινιστική μηχανή Turing.*

### 4.3 Ορισμός της έννοιας του αλγορίθμου

Κατά καιρούς διάφορα χαρακτηριστικά έχουν χρησιμοποιηθεί προκειμένου να περιγραφεί η ιδέα της υπολογισιμότητας (computability). Τα χαρακτηριστικά αυτά προέκυψαν μέσα από διαφορετικές προσεγγίσεις συμπεριλαμβανομένων και των ντετερμινιστικών μηχανών Turing. Παρόλα αυτά τα διαφορετικά αυτά χαρακτηριστικά αποδείχτηκε ότι ουσιαστικά είναι ισοδύναμα μιας και μπορεί εύκολα

και αποδοτικά να πραγματοποιηθεί μετάβαση από το ένα στο άλλο. Αυτή η ισοδυναμία εκφράζεται από την ακόλουθη εικασία (η οποία παραδοσιακά εκφράζεται σε σχέση με τις μηχανές Turing).

**Θεώρημα 4.3.1** (Church's Thesis). *Μια συνάρτηση είναι υπολογίσιμη αν και μόνον αν είναι υπολογίσιμη από μια ντετερμινιστική μηχανή Turing.*

Εξαιτίας της γενικότητας που διέπει την παραπάνω εικασία είναι δύσκολη η παρουσίαση απόδειξης για την ισχύ της. Το μόνο στοιχείο που αυξάνει την αξιοπιστία της είναι ότι μέχρι σήμερα τουλάχιστο δεν έχει βρεθεί αντιπαράδειγμα που να την αναιρεί. Η βασική συνέπεια της θέσης του Church, αν θεωρήσουμε ότι είναι αποδεκτή και ορθή, είναι ότι μπορεί να γίνει λόγος για υπολογίσιμες συναρτήσεις χωρίς απαραίτητα να αναφερθεί η μηχανή η οποία τις αποδέχεται. Η θέση του Church διατυπώθηκε το 1931.

#### 4.4 Αποφασισιμότητα: Αποφασίσιμα προβλήματα σχετικά με κανονικές γλώσσες

Όταν θέλουμε να διαπιστώσουμε αν, για παράδειγμα, κάποια συμβολοσειρά γίνεται αποδεκτή από ένα Ντετερμινιστικό Πεπερασμένο Αυτόματο, αυτό που ουσιαστικά κάνουμε είναι το εξής: εκφράζουμε τη συμβολοσειρά σαν μια γλώσσα  $A$  και να ελέγχουμε αν το DFA αποδέχεται τη γλώσσα  $A_{DFA}$ . Η γλώσσα αυτή περιέχει τις κωδικοποιήσεις για όλα τα DFAs καθώς και τις συμβολοσειρές που αυτά κάνουν αποδεκτές. Θεωρούμε:

$$A_{DFA} = \{ \langle B, w \rangle \mid \text{Το } B \text{ είναι ένα DFA που δέχεται τη συμβολοσειρά } w \}.$$

Το πρόβλημα ελέγχου αποδοχής της συμβολοσειράς  $w$  από το DFA  $B$  είναι το ίδιο με το πρόβλημα: "Είναι το  $\langle B, w \rangle$  : μέλος της γλώσσας  $A_{DFA}$ ; ". Με τον τρόπο αυτό μπορούμε να διατυπώσουμε υπολογιστικά προβλήματα ώστε τελικά να πρέπει να ελέγξουμε αν κάποια συμβολοσειρά ανήκει σε μια γλώσσα. Δείχνοντας ότι η γλώσσα είναι αποφασίσιμη εξασφαλίζουμε ότι και το υπολογιστικό πρόβλημα είναι επιλύσιμο. Το ακόλουθο θεώρημα δηλώνει ότι το πρόβλημα ελέγχου αποδοχής μιας συμβολοσειράς από ένα DFA είναι αποφασίσιμο.

**Θεώρημα 4.4.1.** *Η γλώσσα  $A_{DFA}$  είναι αποφασίσιμη.*

*Απόδειξη.* Αυτό που απαιτείται είναι η παρουσίαση μιας μηχανής Turing από την οποία η γλώσσα  $A_{DFA}$  να είναι υπολογίσιμη.

Η ζητούμενη μηχανή Turing θα πρέπει να είναι της μορφής:

$M =$  "Με είσοδο  $\langle B, w \rangle$ , όπου  $B$  είναι ένα DFA και  $w$  μια συμβολοσειρά:

1. Εξομοίωσε τη λειτουργία του  $B$  με είσοδο  $w$ .
2. Αν η εξομοίωση καταλήξει σε κατάσταση αποδοχής, τότε κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο."

Έστω  $B = (Q, \Sigma, \delta, q_0, F)$ . Τότε όταν η μηχανή  $M$  δέχεται την είσοδο  $\langle B, w \rangle$ , αρχικά ελέγχει την ορθότητα των  $B$  και  $w$  και τερματίζει απορρίπτοντας την είσοδο αν κάποιο από τα  $B$  ή  $w$  δεν ακολουθεί τον ορισμό του. Στη συνέχεια η  $M$ , εκτελεί την εξομοίωση καταγράφοντας στην ταινία της την τρέχουσα κατάσταση του αυτομάτου  $B$  και την τρέχουσα θέση του  $B$  στη συμβολοσειρά εισόδου  $w$ . Αρχικά, η τρέχουσα κατάσταση του  $B$  είναι η  $q_0$  και η τρέχουσα θέση του στη συμβολοσειρά είναι το αριστερότερο σύμβολο της  $w$ . Οι καταστάσεις και η τρέχουσα θέση καθορίζονται και τροποποιούνται με βάση τη συνάρτηση μετάβασης  $\delta$ . Όταν η μηχανή επεξεργαστεί και το τελευταίο σύμβολο της  $w$ , η  $M$  τερματίζει και είτε αποδέχεται τη συμβολοσειρά εισόδου αν το  $B$  έχει φτάσει σε κατάσταση αποδοχής, είτε απορρίπτει τη συμβολοσειρά εισόδου αν το  $B$  δεν φτάσει σε κατάσταση αποδοχής.  $\square$

Μπορούμε να αποδείξουμε ανάλογο θεώρημα για τα Μη Ντετερμινιστικά Αυτόματα. Έστω

$$A_{NFA} = \{ \langle B, w \rangle \mid \text{Το } B \text{ είναι ένα NFA που δέχεται τη συμβολοσειρά } w \}.$$

**Θεώρημα 4.4.2.** *Η γλώσσα  $A_{NFA}$  είναι αποφασίσιμη.*

*Απόδειξη.* Παρουσιάζουμε μια μηχανή Turing που να δέχεται τη γλώσσα  $A_{NFA}$ . Θα μπορούσαμε να σχεδιάσουμε τη μηχανή  $N$  να λειτουργεί όπως και η  $M$  και να εξομοιώνει ένα NFA αντί για ένα DFA. Θα ακολουθήσουμε όμως μια παραπλήσια μέθοδο: Επειδή η μηχανή  $M$  σχεδιάστηκε να λειτουργεί με DFAs, η μηχανή  $N$  αρχικά μετατρέπει το NFA που δέχεται σαν είσοδο σε DFA και δίνει το νέο αυτόματο σαν είσοδο στη μηχανή  $M$ .

$N =$  “Με είσοδο  $\langle B, w \rangle$ , όπου  $B$  είναι ένα NFA και  $w$  μια συμβολοσειρά:

1. Μετάτρεψε το NFA  $B$  στο ισοδύναμο του DFA  $C$ .
2. Εκτέλεσε τη λειτουργία της μηχανής  $M$  (που περιγράφηκε στο θεώρημα 4.4.1) με είσοδο  $\langle C, w \rangle$ .
3. Αν η μηχανή  $M$  καταλήξει σε κατάσταση αποδοχής, τότε κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

Η εκτέλεση των λειτουργιών της μηχανής  $M$  στο βήμα 2 σημαίνει απλά ενσωμάτωση της μηχανής  $M$  στο σχεδιασμό της  $N$ , σαν υπορουτίνα της.  $\square$

Όμοια μπορούμε να διαπιστώσουμε αν μια κανονική έκφραση μπορεί να παράγει μια δοσμένη συμβολοσειρά. Έστω:

$$A_{REX} = \{ \langle R, w \rangle \mid \text{Η } R \text{ είναι μια κανονική έκφραση που δέχεται τη συμβολοσειρά } w \}.$$

**Θεώρημα 4.4.3.** *Η γλώσσα  $A_{REX}$  είναι αποφασίσιμη.*

Απόδειξη. Η ακόλουθη μηχανή Turing αποφασίζει τη γλώσσα  $A_{REX}$ .

$P =$  “Με είσοδο  $\langle R, w \rangle$ , όπου  $R$  είναι μια κανονική έκφραση και  $w$  μια συμβολοσειρά:

1. Μετάτρεψε την κανονική έκφραση  $R$  στο ισοδύναμό της DFA  $C$ .
2. Εκτέλεσε τη λειτουργία της μηχανής  $M$  (που περιγράφηκε στο θεώρημα 4.4.1) με είσοδο  $\langle C, w \rangle$ .
3. Αν η μηχανή  $M$  καταλήξει σε κατάσταση αποδοχής, τότε κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

Τα θεωρήματα 4.4.1, 4.4.2 και 4.4.3 δηλώνουν ότι στα πλαίσια της αποφασισιμότητας οι μηχανές Turing, τα DFAs, τα NFAs και οι κανονικές εκφράσεις είναι ισοδύναμα. Μέχρι τώρα το πρόβλημα που μελετήσαμε ήταν κατά πόσο κάποια συμβολοσειρά γίνεται ή όχι από ένα υπολογιστικό μοντέλο. Στη συνέχεια θα μελετήσουμε το εξής πρόβλημα: κατά πόσον ένα υπολογιστικό μοντέλο (μηχανές Turing, DFAs, NFAs, κανονικές εκφράσεις) δέχεται ή όχι συμβολοσειρές.

Έστω

$$E_{DFA} = \{ \langle A \rangle \mid \text{Το } A \text{ είναι ένα DFA και } L(A) = \emptyset \}.$$

**Θεώρημα 4.4.4.** *Η γλώσσα  $E_{DFA}$  είναι αποφασίσιμη.*

Απόδειξη. Ένα DFA αποδέχεται κάποια συμβολοσειρά αν και μόνον μπορεί να φτάσει σε κατάσταση αποδοχής, με αφετηρία την αρχική κατάσταση και ακολουθώντας τις μεταβάσεις που επιτρέπονται από τη συνάρτησή του. Για να πραγματοποιήσουμε τον παραπάνω έλεγχο σχεδιάζουμε μια μηχανή Turing που να χρησιμοποιεί έναν αλγόριθμο σημείωσης (marking algorithm).

$T =$  “Με είσοδο  $\langle A \rangle$ , όπου  $A$  είναι ένα DFA:

1. Σημείωσε την αρχική κατάσταση του  $A$ .
2. Επανάλαβε έως ότου δεν υπάρχουν νέες καταστάσεις για σημείωση.
3. Σημείωσε κάθε κατάσταση στην οποία υπάρχει μετάβαση από κατάσταση που έχει ήδη σημειωθεί.
4. Αν καμία κατάσταση αποδοχής δεν έχει σημειωθεί κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

Το ακόλουθο θεώρημα δηλώνει ότι η διαδικασία ελέγχου για το αν δύο DFAs που αναγνωρίζουν την ίδια γλώσσα είναι ισοδύναμα, είναι μια αποφασίσιμη διαδικασία. Έστω:

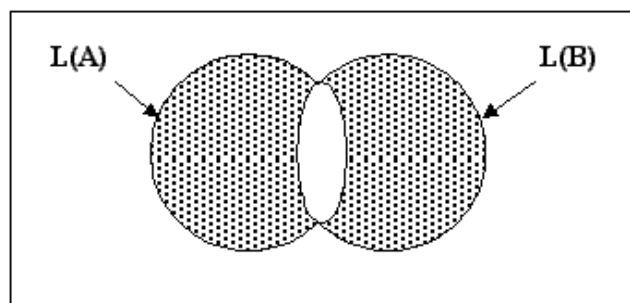
$$EQ_{DFA} = \{ \langle A, B \rangle \mid \text{Τα } A \text{ και } B \text{ είναι DFAs και } L(A) = L(B) \}.$$

**Θεώρημα 4.4.5.** Η γλώσσα  $EQ_{DFA}$  είναι αποφασίσιμη.

*Απόδειξη.* Για την απόδειξη χρησιμοποιούμε το θεώρημα 4.4.4. Κατασκευάζουμε ένα νέο DFA  $C$ , από τα  $A$  και  $B$ , που να δέχεται μόνο εκείνες τις συμβολοσειρές που γίνονται δεκτές είτε από το  $A$  είτε από το  $B$  αλλά όχι και από τα δύο. Επομένως, αν τα  $A$  και  $B$  αναγνωρίζουν την ίδια γλώσσα τότε το  $C$  δεν δέχεται τίποτα. Η γλώσσα που γίνεται δεκτή από το  $C$  είναι η

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)).$$

Η έκφραση αυτή καλείται και συμμετρική διαφορά (symmetric difference) των γλωσσών  $L(A)$  και  $L(B)$  και σχηματικά φαίνεται παρακάτω.



Με  $\overline{L(A)}$  συμβολίζουμε το συμπλήρωμα της  $L(A)$ . Η συμμετρική διαφορά είναι σημαντική για την απόδειξη μιας και είναι  $L(C) = \emptyset$  αν και μόνον αν  $L(A) = L(B)$ . Οι αλγόριθμοι κατασκευής του  $C$  είναι εκτελέσιμοι από μηχανές Turing. Αφού έχουμε κατασκευάσει το  $C$ , χρησιμοποιούμε το θεώρημα 4.4.4 για να διαπιστώσουμε αν  $L(C) = \emptyset$  και στην περίπτωση που είναι ισχύει  $L(A) = L(B)$ .

$F =$  “Με είσοδο  $\langle A, B \rangle$ , όπου  $A$  και  $B$  είναι DFAs:

1. Κατασκεύασε το DFA  $C$ .
2. Εκτέλεσε τη λειτουργία της μηχανής  $T$  του θεωρήματος 4.4.4.
3. Αν η μηχανή  $T$  φτάσει σε κατάσταση αποδοχής, κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

## 4.5 Αποφασίσισιμα προβλήματα σχετικά με γλώσσες χωρίς συμφραζόμενα

Στη συνέχεια περιγράφουμε αλγορίθμους για τον έλεγχο του αν μια συγκεκριμένη συμβολοσειρά μπορεί να παραχθεί από μια γραμματική χωρίς συμφραζόμενα καθώς και του αν η γλώσσα που παράγεται από μια γραμματική χωρίς είναι κενή. Έστω:

$A_{CFG} = \{ \langle G, w \rangle \mid \text{Η } G \text{ είναι μια γραμματική χωρίς συμφραζόμενα που παράγει τη συμβολοσειρά } w \}.$

**Θεώρημα 4.5.1.** *Η γλώσσα  $A_{CFG}$  είναι αποφασίσιμη.*

*Απόδειξη.* Για μια γραμματική χωρίς συμφραζόμενα  $G$  και μια συμβολοσειρά  $w$ , επιθυμούμε να διαπιστώσουμε αν η  $G$  μπορεί να δώσει την  $w$ . Μια μη αποδοτική ιδέα είναι να δοκιμάσουμε όλους τους κανόνες της  $G$  ώστε να διαπιστώσουμε αν τελικά η  $G$  μπορεί να δώσει την  $w$ . Το πρόβλημα είναι ότι μπορεί να υπάρχουν άπειροι κανόνες και στην περίπτωση που η  $G$  δεν μπορεί να δώσει την  $w$ , ο αλγόριθμος δεν θα τερμάτιζε ποτέ. Σημειώνουμε ότι η παραπάνω ιδέα καταλήγει σε μια μηχανή Turing που μπορεί να αναγνωρίσει αλλά δεν μπορεί να αποφασίσει για τη γλώσσα  $A_{CFG}$ . Πρέπει επομένως να φτιάξουμε έναν αλγόριθμο που να πραγματοποιεί τον έλεγχο με πεπερασμένο αριθμό κανόνων. Ισχύει ότι αν μια γραμματική  $G$  είναι σε κανονική μορφή Chomsky, τότε κάθε παραγωγή μιας συμβολοσειράς  $w$  διαρκεί  $2n - 1$  βήματα, όπου  $n$  είναι το μήκος της  $w$ . Οπότε για να διαπιστώσουμε αν η  $G$  παράγει τη  $w$  ελέγχουμε μόνο κανόνες  $2n - 1$  βημάτων. Και υπάρχει πεπερασμένος αριθμός τέτοιων κανόνων. Η  $G$  μπορεί εύκολα να τροποποιηθεί ώστε να έχει κανονική μορφή Chomsky. Στη συνέχεια παρουσιάζουμε τη μηχανή Turing για τη γλώσσα  $A_{CFG}$ .

$F =$  “Με είσοδο  $\langle G, w \rangle$ , όπου  $G$  είναι μια γραμματική χωρίς συμφραζόμενα και  $w$  είναι μια συμβολοσειρά:

1. Μετάτρεψε τη  $G$  στην ισοδύναμη της που είναι σε κανονική μορφή Chomsky.
2. Βρες όλους τους κανόνες  $2n - 1$  βημάτων, όπου  $n$  το μήκος της  $w$ . Στην περίπτωση που  $n = 0$ , βρες όλους τους κανόνες 1 βήματος.
3. Αν κάποιος από αυτούς τους κανόνες παράγει την  $w$ , κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

Το πρόβλημα του ελέγχου αν μια γραμματική χωρίς συμφραζόμενα παράγει μια συγκεκριμένη συμβολοσειρά σχετίζεται με το πρόβλημα μετάφρασης (compiling) γλωσσών προγραμματισμού. Ο αλγόριθμος που υλοποιεί η μηχανή Turing που μόλις περιγράψαμε αν και είναι απλός και κατάλληλος για την απόδειξη είναι μη αποδοτικός και δεν χρησιμοποιείται στην πράξη. Σημειώνουμε, επίσης, ότι όλα όσα δείχνουμε για γραμματικές χωρίς συμφραζόμενα ισχύουν και για τα αυτόματα στοίβας μιας και έχουμε δείξει ότι γραμματικές χωρίς συμφραζόμενα και αυτόματα στοίβας είναι ισοδύναμα υπολογιστικά μοντέλα. Στη συνέχεια μελετάμε το πρόβλημα του αν κάποια γραμματική χωρίς συμφραζόμενα δέχεται συμβολοσειρές. Έστω:

$$E_{CFG} = \{ \langle G \rangle \mid \text{Η } G \text{ είναι μια γραμματική χωρίς συμφραζόμενα και } L(G) = \emptyset \}.$$

**Θεώρημα 4.5.2.** *Η γλώσσα  $E_{CFG}$  είναι αποφασίσιμη.*

Απόδειξη. Προκειμένου να κατασκευάσουμε έναν αλγόριθμο γι' αυτό το πρόβλημα μπορούμε να χρησιμοποιήσουμε τη μηχανή Turing του Θεωρήματος 4.5.1 η οποία μπορεί να ελέγχει αν μια γραμματική χωρίς συμφραζόμενα μπορεί να δώσει μια συγκεκριμένη συμβολοσειρά. Για να διαπιστώσουμε αν  $L(G) = \emptyset$  ο αλγόριθμος μπορεί να δοκιμάσει όλες τις πιθανές συμβολοσειρές  $w$ . Αλλά υπάρχουν άπειρες συμβολοσειρές με αποτέλεσμα ο αλγόριθμος ενδεχομένως να εκτελείται επ' άπειρον. Κατά συνέπεια ακολουθούμε μια διαφορετική προσέγγιση. Για να διαπιστώσουμε αν μια γλώσσα είναι κενή, πρέπει να ελέγξουμε αν η αρχική μεταβλητή μπορεί να δώσει μια ακολουθία τερματικών συμβόλων. Ο αλγόριθμος τότε σημειώνει κάθε τέτοια μεταβλητή. Αρχικά ο αλγόριθμος σημειώνει τα τερματικά σύμβολα της γραμματικής και στη συνέχεια σαρώνει όλους τους κανόνες της. Αν βρει κάποιον κανόνα αντικατάστασης μιας μεταβλητής με μια συμβολοσειρά της οποίας τα σύμβολα είναι σημειωμένα, σημειώνει τη μεταβλητή αυτή. Ο αλγόριθμος συνεχίζει μέχρις ότου υπάρχουν μεταβλητές για σημείωση.

$R =$  “Με είσοδο  $\langle G \rangle$ , όπου  $G$  είναι μια γραμματική χωρίς συμφραζόμενα:

1. Σημείωσε όλα τα τερματικά σύμβολα της  $G$ .
2. Επανάλαβε έως ότου δεν υπάρχουν νέες μεταβλητές για σημείωση.
3. Σημείωσε κάθε μεταβλητή  $A$  για την οποία υπάρχει στη  $G$  κανόνας  $A \rightarrow U_1 U_2 \dots U_k$  και όλα τα σύμβολα  $U_i$  έχουν ήδη σημειωθεί.
4. Αν το αρχικό σύμβολο δεν έχει σημειωθεί κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

Στη συνέχεια μελετάμε το πρόβλημα του αν δύο γλώσσες χωρίς συμφραζόμενα παράγουν την ίδια γλώσσα. Έστω:

$$EQ_{CFG} = \{ \langle G, H \rangle \mid \text{Οι } G \text{ και } H \text{ είναι γλώσσες χωρίς συμφραζόμενα και } L(G) = L(H) \}.$$

Σύμφωνα με το Θεώρημα 4.4.5 υπάρχει αλγόριθμος που αποφασίζει για το ίδιο πρόβλημα αλλά σχετικά με πεπερασμένα αυτόματα. Με βάση αυτόν τον αλγόριθμο χρησιμοποιήθηκε η διαδικασία απόφασης  $EDFA$  για να αποδειχθεί ότι το σύνολο  $EQ_{DFA}$  είναι αποφασίσιμο. Εντούτοις η ιδέα ότι μιας και το  $E_{CFG}$  είναι αποφασίσιμο θα μπορούσαμε να το χρησιμοποιήσουμε για να δείξουμε ότι και το  $EQ_{CFG}$  είναι αποφασίσιμο δεν είναι σωστή μιας και η κλάση των γλωσσών χωρίς συμφραζόμενα δεν είναι κλειστή ως προς την τομή και τη συμπλήρωση. Μάλιστα, το σύνολο  $EQ_{CFG}$  δεν είναι αποφασίσιμο. Στη συνέχεια θα δείξουμε ότι η κλάση των γλωσσών χωρίς συμφραζόμενα είναι αποφασίσιμη από κάποια μηχανή Turing.

**Θεώρημα 4.5.3.** *Κάθε γλώσσα χωρίς συμφραζόμενα είναι υπολογίσιμη.*

Απόδειξη. Έστω  $A$  μια γραμματική χωρίς συμφραζόμενα. Θέλουμε να δείξουμε ότι το σύνολο  $A$  είναι αποφασίσιμο. Η ιδέα να κατασκευάσουμε ένα αυτόματο στοίβας για την  $A$  και στη συνέχεια να το μετατρέψουμε σε μια ισοδύναμη μηχανή Turing είναι μεν απλή και εκ πρώτης όψεως φαίνεται να δουλεύει ακόμα και στην περίπτωση που το αυτόματο στοίβας είναι μη ντετερμινιστικό, μιας και μπορούμε να το μετατρέψουμε σε μια μη ντετερμινιστική μηχανή Turing και στη συνέχεια να μετατρέψουμε αυτή στην ισοδύναμή της ντετερμινιστική. Ενδεχομένως όμως κάποια υπολογιστικά μονοπάτια του αυτόματου στοίβας να εκτελούν άπειρες επαναλήψεις ανάγνωσης/εγγραφής και να μην οδηγούν σε τελική κατάσταση, με συνέπεια η ισοδύναμη μηχανή Turing να έχει επίσης υπολογιστικά μονοπάτια που δεν οδηγούν σε τελική κατάσταση. Για την απόδειξη επομένως δεν χρησιμοποιούμε την παραπάνω τεχνική αλλά τη μηχανή Turing του θεωρήματος 4.5.1. Έστω  $G$  μια γραμματική χωρίς συμφραζόμενα. Θα σχεδιάσουμε μια μηχανή Turing  $M_G$  που να μπορεί να αποφασίσει τη γλώσσα  $A$ . Κατασκευάζουμε ένα αντίγραφο της  $G$  στη μηχανή Turing  $M_G$ .

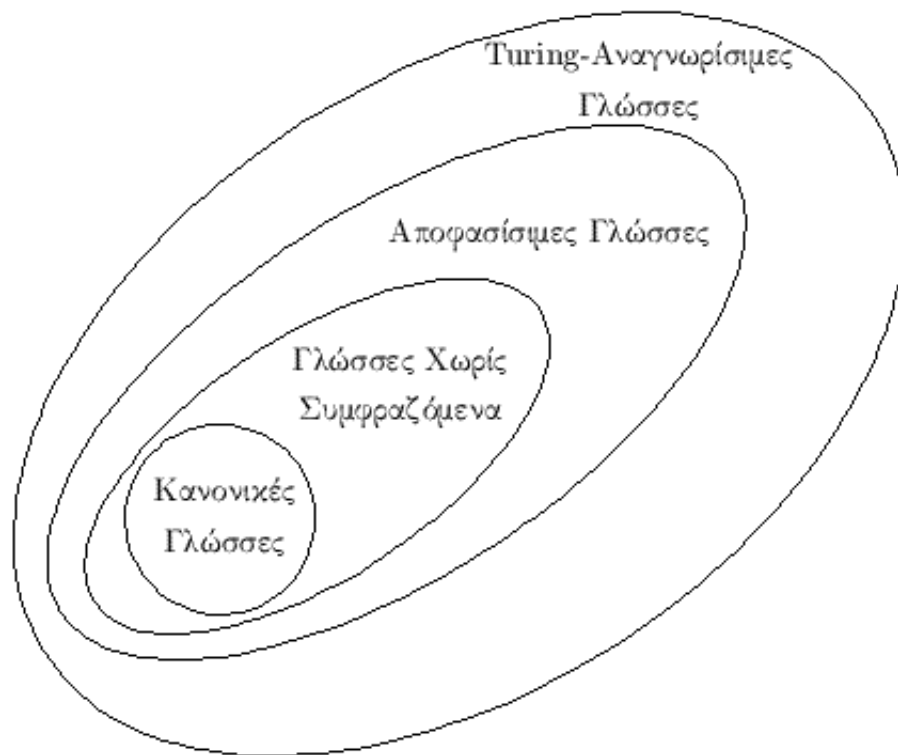
$M_G =$  “Με είσοδο μια συμβολοσειρά  $w$ :

1. Εκτέλεσε τη λειτουργία της μηχανής  $S$  του θεωρήματος 4.5.1 με είσοδο  $\langle G, w \rangle$ .
2. Αν η μηχανή  $S$  φτάσει σε κατάσταση αποδοχής, κάνε δεκτή τη συμβολοσειρά εισόδου. Διαφορετικά, απόρριψε την είσοδο.”

□

Το τελευταίο Θεώρημα παρέχει τον τελικό συσχετισμό των τεσσάρων βασικών κλάσεων γλωσσών που έχουμε περιγράψει μέχρι τώρα: κανονικές, χωρίς συμφραζόμενα, αποφασίσιμες και αναγνωρίσιμες από μηχανές Turing. Η σχέση μεταξύ απεικονίζεται στο παρακάτω σχήμα.







# Βιβλιογραφία

- [1] ADLEMAN L. *Two theorems on random polynomial time.* IN *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science(FOCS '78)*, PP. 75–83, 1978.
- [2] ADLEMAN L. AND HUANG M. A. *Recognizing primes in random polynomial time.* IN *Proceedings of the 19th IEEE annual ACM Symposium on the Theory of Computing (STOC '87)*, PP. 462–469, 1987.
- [3] CHOMSKY N. *Three models for the description of language.* *IRE Trans. on Information Theory 2 (1956)*, PP. 113–124, 1956.
- [4] CORMEN T.H. , LEISERSON C.E. , AND RIVEST R.L., **Introduction to Algorithms**, MIT PRESS, 1991.
- [5] GÖDEL K. *On formally undecidable propositions* IN *Principia Mathematica* AND RELATED SYSTEMS I. *The undecidable*, M. DAVIS, ED. RAVEN PRESS, 1965, PP. 4–38.
- [6] FEYNMAN. R. P., HEY A. J., AND ALLEN R. W. *Feynman lectures on Computation*, ADDISON-WESLEY, 1996.
- [7] GARAY M. R. AND JOHNSON D. S., **Computers and Intractability**, W.H. FREEMAN AND COMPANY, 1979.
- [8] GOLDWASSER S. AND MICHALI S. *Probabilistic Encryption*, *Journal of Computer and System Sciences.* (1984), 170–229.
- [9] GOLDWASSER S., MICHALI S. AND RACKOFF C. *The knowledge Complexity of Interactive Proof-Systems.* *SIAM Journal of Computing.* (1989), 186–208.
- [10] HARRARY F. *Graph Theory.* 2ND ED. ADDISON-WESLEY, 1971.
- [11] HOPCROFT J. E. AND ULLMAN J. D., **Introduction to Automata Theory, Languages, and Computation**, ADDISON-WESLEY, 1979.
- [12] LEIGHTON F. T. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes .* MORGAN KAUFMANN, 1991.

- [13] LEWIS H. R. ΚΑΙ ΠΑΠΑΔΗΜΗΤΡΙΟΥ Χ. Χ., **ΣΤΟΙΧΕΙΑ ΘΕΩΡΙΑΣ ΥΠΟΛΟΓΙΣΜΟΥ**, ΤΕΕ, ΒΙΒΛΙΟΘΗΚΗ ΠΛΗΡΟΦΟΡΙΚΗΣ, 1992.
- [14] LEWIS H. AND PAPADIMITRIOU C. *Elements of the Theory of Computation*. PRENTICE-HALL, 1991.
- [15] PAPADIMITRIOU C. H. *Computational Complexity*. ADDISON-WESLEY, 1994.
- [16] RABIN M. *Probabilistic Algorithms*. IN *algorithms and Complexity: New Directions and Recent Results*., J. F. TRAUB, ED. ACADEMIC PRESS, 1976, PP. 21-39.
- [17] SISPER M. *Introduction to the Theory of Computation*. PWS PUBLISHING COMPANY, 1997.
- [18] TURING A. M. *On computable numbers, with an application to the Entscheidungs Problem*. IN *Proceedings, London Mathematical Society*, 1936.