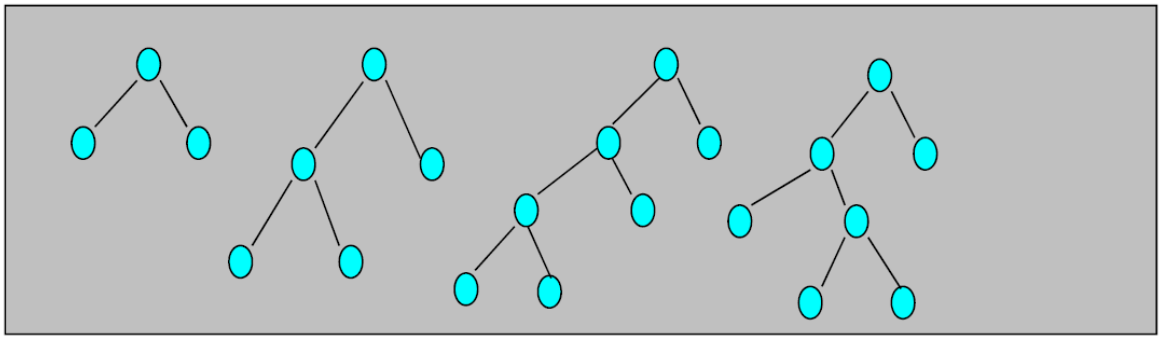
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

ΚΛΑΣΕΙΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ ΧΡΟΝΟΥ – ΑΣΥΜΠΤΩΤΙΚΗ ΑΝΑΛΥΣΗ



ΑΣΤΡΟΠΕΚΑΚΗ ΔΕΣΠΟΙΝΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΚΟΤΙΝΗ ΙΣΑΒΕΛΛΑ

ΘΕΣΣΑΛΟΝΙΚΗ 2010

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Τα προβλήματα που μπορούν να υπολογιστούν μέσω των υπολογιστών είναι διαφόρων ειδών, κάποια από αυτά είναι εύκολα και κάποια δύσκολα. Το πρόβλημα της ταξινόμησης, παραδείγματος χάριν, είναι εύκολο. Έστω ότι θέλετε να διατάξετε ένα σύνολο αριθμών κατ' αύξουσα σειρά. Ακόμα και ένας μικρός υπολογιστής είναι σε θέση να ταξινομήσει ένα εκατομμύριο αριθμούς σχετικά γρήγορα. Από την άλλη πλευρά, θεωρήστε το πρόβλημα του χρονοπρογραμματισμού. Έστω π.χ. ότι θέλετε να καταρτίσετε ένα ωρολόγιο πρόγραμμα για τα μαθήματα όλων των τμημάτων ενός πανεπιστημίου το οποίο να ικανοποιεί κάποιους εύλογους περιορισμούς, όπως λόγου χάριν ότι κανένα ζεύγος μαθημάτων δεν είναι δυνατόν να πραγματοποιηθεί ταυτόχρονα σε μια αίθουσα. Αυτό το πρόβλημα του χρονοπρογραμματισμού φαίνεται να είναι πολύ δυσκολότερο από το πρόβλημα της ταξινόμησης. Ακόμη και αν έχετε χίλια μόνο μαθήματα, η εύρεση του καλύτερου ωρολογίου προγράμματος πιθανόν απαιτεί αιώνες, ακόμη και με έναν υπερυπολογιστή.

Τι είναι αυτό που κάνει κάποια προβλήματα υπολογιστικώς δύσκολα και κάποια άλλα εύκολα;

Αυτό είναι το κεντρικό ερώτημα της θεωρίας πολυπλοκότητας. Είναι εντυπωσιακό ότι, παρά την εντατική σχετική έρευνα που έχει πραγματοποιηθεί από το 1995 και μετά, η απάντηση εξακολουθεί να μας είναι άγνωστη. Παρακάτω στην εργασία αυτή θα μελετήσουμε αυτό το συναρπαστικό ερώτημα και κάποιες από τις συνέπειες του.

Όταν αντιμετωπίζουμε κάποιο πρόβλημα που φαίνεται να είναι υπολογιστικά δύσκολο, έχουμε διάφορες επιλογές. Πρώτον, εάν κατανοήσουμε ποιο χαρακτηριστικό του προβλήματος προκαλεί τη δυσκολία, ίσως καταφέρουμε να το τροποποιήσουμε ώστε το πρόβλημα να επιλύεται ευκολότερα. Δεύτερον, ίσως να μπορούμε να αρκεστούμε σε μια κάπως ατελή λύση. Σε ορισμένες περιπτώσεις, η εύρεση λύσεων που απλώς προσεγγίζουν την τέλεια λύση είναι σχετικά εύκολη. Τρίτον, ορισμένα προβλήματα είναι δύσκολα μόνο στη χειρότερη περίπτωση, ενώ στις περισσότερες περιπτώσεις είναι εύκολα. Ανάλογα με την εφαρμογή, πιθανόν να μας αρκεί μια διαδικασία που, παρ' ότι μερικές φόρες είναι αργή, συνήθως εκτελείται γρήγορα. Τέλος, μπορούμε να εξετάσουμε εναλλακτικούς τύπους

υπολογισμού, όπως πχ πιθανοκρατικοί, οι οποίοι μπορούν να επιταχύνουν κάποιες εργασίες.

Ακόμα και όταν ένα πρόβλημα είναι επιλύσιμο, και άρα μπορεί να αντιμετωπιστεί υπολογιστικά, πιθανόν να είναι πρακτικά ανεπίλυτο εάν η επίλυση του απαιτεί εξωπραγματικό χρόνο ή υπερβολικά πολλή μνήμη. Η εργασία αυτή αποτελεί μια έρευνα στην υπολογιστική πολυπλοκότητα χρόνου η οποία μελετά το χρόνο που απαιτείται για την επίλυση υπολογιστικών προβλημάτων. Επίσης, θα γίνει αναφορά στα μαθηματικά μοντέλα της υπολογιστικής επιστήμης όπως τα πεπερασμένα αυτόματα και η γραμματική «ανεξάρτητη συμφραζομένων» που θα μας επιτρέψουν να εξοικειωθούμε με τους τυπικούς ορισμούς της υπολογιστικής πολυπλοκότητα

ΚΕΦΑΛΑΙΟ 1

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

Στο κεφάλαιο αυτό θα ασχοληθούμε με κάποια βασικά στοιχεία που είναι απαραίτητα για την μετέπειτα κατανόηση του υπολογισμού της πολυπλοκότητας του χρόνου των υπολογιστικών προβλημάτων.

Μερικά απ'αυτά είναι, η έννοια της ασυμπτωτικής ανάλυσης και παράλληλα οι έννοιες αλφάβητο, γλώσσα, γραφήματα, που θα μας βοηθήσουν να κατανοήσουμε τα υπολογιστικά μοντέλα που θα παρουσιαστούν για τον σκοπό αυτής της εργασίας.

1.1 ΑΣΥΜΠΤΩΤΙΚΗ ΑΝΑΛΥΣΗ

ΟΙ ΣΥΜΒΟΛΙΣΜΟΙ ΚΕΦΑΛΑΙΟΥ ΚΑΙ ΠΕΖΟΥ ΟΜΙΚΡΟΝ

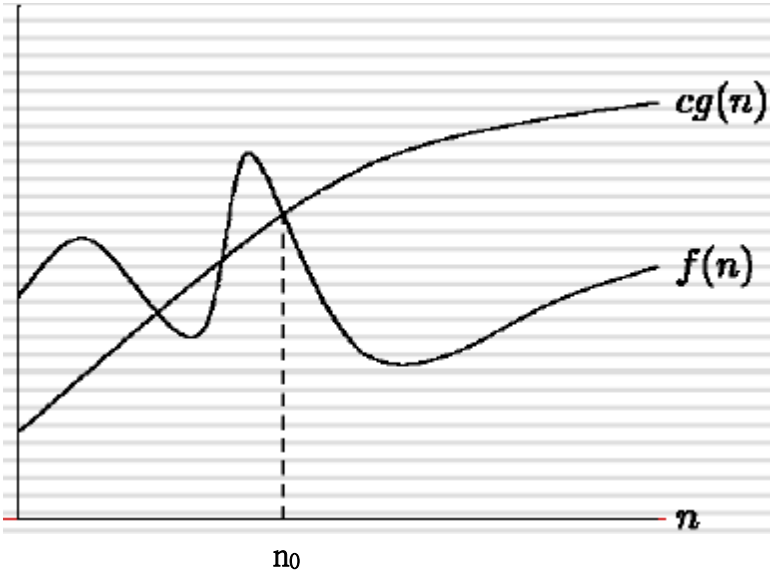
Όπως ξέρουμε, αλγόριθμος είναι μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, με σκοπό την επίλυση ενός προβλήματος.

Η έκφραση που αναπαριστά τον ακριβή χρόνο εκτέλεσης ενός αλγορίθμου συχνά είναι αρκετά περίπλοκη. Για τον λόγο αυτό, συνήθως περιοριζόμαστε σε κάποια εκτίμηση του χρόνου αυτού. Μια απλή μορφή τέτοιας εκτίμησης είναι η λεγόμενη ασυμπτωτική ανάλυση, όπου το ζητούμενο είναι να κατανοήσουμε τον χρόνο που απαιτεί ο αλγόριθμος όταν εκτελείται για εισόδους μεγάλου μήκους. Αυτό σημαίνει ότι στην έκφραση που αναπαριστά τον ακριβή χρόνο εκτέλεσης του αλγορίθμου μας ενδιαφέρει μόνο ο μεγιστοβάθμιος όρος, και αγνοούμε τόσο τον συντελεστή του όρου αυτού όσο και τους όρους χαμηλότερου βαθμού, αφού στις εισόδους μεγάλου μήκους ο μεγιστοβάθμιος όρος υπερισχύει έναντι όλων των υπολοίπων.(Sipser, 1996)

Παραδείγματος χάριν, η συνάρτηση $f(n) = 6n^3+2n^2+20n +45$ έχει τέσσερεις όρους, από τους οποίους ο μεγιστοβάθμιος είναι ο $6n^3$. Αδιαφορώντας για τον συντελεστή 6, λέμε ότι η f είναι ασυμπτωτικά το πολύ n^3. Η σχέση αυτή αναπαρίσταται συμβολικά ως $f(n) = O(n^3)$. Πρόκειται για τον λεγόμενο ασυμπτωτικά συμβολισμό ή συμβολισμό κεφαλαίου όμικρον, που ορίζεται τυπικά

αμέσως παρακάτω. Το σύμβολο R^+ αναπαριστά το σύνολο των μη αρνητικών πραγματικών αριθμών.

Έστω f και g δύο συναρτήσεις από το σύνολο N στο σύνολο R^+. Λέμε ότι f(n) = O(g(n)), εάν υπάρχουν θετικοί ακέραιοι c και n_0 τέτοιοι ώστε, για κάθε ακέραιο $n \geq n_0$, να ισχύει f(n) < cg(n). Όταν f(n) = O(g(n)), λέμε ότι η g(n) αποτελεί άνω φράγμα της f(n) ή, ακριβέστερα, ασυμπτωτικό άνω φράγμα της f(n), για να τονίσουμε ότι αγνοούμε τους σταθερούς συντελεστές.



ΣΧΗΜΑ 1.1 Ασυμπτωτικός Συμβολισμός

Σε περιγραφικό επίπεδο, η σχέση f(n) = O(g(n)) σημαίνει ότι, εάν παραβλέψουμε διαφορές μέχρι κάποιον σταθερό πολλαπλασιαστικό παράγοντα, η f είναι μικρότερη ή ίση της g. Στην πράξη, οι περισσότερες από τις συναρτήσεις f που είναι πιθανόν να συναντήσουμε έχουν κάποιον προφανή μεγιστοβάθμιο όρο h. Στην περίπτωση αυτή, μπορούμε να γράφουμε f(n) = O(g(n)), όπου g ο όρος h χωρίς τον συντελεστή του.

Έστω f_1 (n) η συνάρτηση $5n^3+ 2n^2 + 22n + 6$. Επιλέγοντας τον μεγιστοβάθμιο όρο $5n^3$ και αγνοώντας τον συντελεστή 5, παίρνουμε $f_1(n) = O(n^3)$. Ας επαληθεύσουμε ότι το συμπέρασμα μας ικανοποιεί τον τυπικό ορισμό. Για τον σκοπό αυτό, επιλέγουμε ως c την τιμή 6 και ως n_0 την τιμή 10, οπότε έχουμε $5n^3+2n^2+ 22n + 6 < 6n^3$ για κάθε $n \geq 10$, το οποίο όντως ισχύει.

Επιπλέον, ισχύει και ότι f_1 (n) = $O(n^4)$, αφού η συνάρτηση n^4 είναι μεγαλύτερη της n^3 και επομένως αποτελεί και αυτή ασυμπτωτικό άνω φράγμα της f_1. Αντιθέτως, η $f_1(n)$ δεν είναι $O(n^2)$. Στην περίπτωση αυτή, ο ορισμός είναι

προφανώς αδύνατον να ικανοποιηθεί, όποιες τιμές και αν αναθέσουμε στους ακεραίους c και n_0.

Ο συμβολισμός κεφαλαίου όμικρον παρουσιάζει μια ιδιαίτερη συμπεριφορά όταν αλληλεπιδρά με λογαρίθμους. Συνήθως, όταν χρησιμοποιούμε λογαρίθμους θα πρέπει να προσδιορίζουμε τη βάση τους. Παραδείγματος χάριν, στην έκφραση $x = \log_2 n$, η βάση b = 2 υποδεικνύει ότι η συγκεκριμένη ισότητα είναι ισοδύναμη της $2^x = n$. Εάν αλλάξουμε την τιμή της βάσης b, η τιμή του $\log_b n$ μεταβάλλεται επίσης. Ωστόσο, σύμφωνα με την ταυτότητα $\log_b n = \log_2 n / \log_2 b$, η τιμή του λογαρίθμου μεταβάλλεται μόνο κατά έναν σταθερό πολλαπλασιαστικό παράγοντα. Επομένως, όταν γράφουμε f(n) = O(logn), δεν είναι απαραίτητο να διευκρινίζουμε τη βάση, αφού ο σταθερός παράγοντας που θα προκύψει θα παραβλεφθεί ούτως ή άλλως.

Έστω π.χ. η συνάρτηση $f_2(n) = 3n\log_2 n + 5n\log_2\log_2 n + 2$. Στην περίπτωση αυτή, έχουμε $f_2(n) = O(n \log n)$, αφού η συνάρτηση log n υπερισχύει της log (log n).

Ο συμβολισμός κεφαλαίου όμικρον εμφανίζεται επίσης σε αριθμητικές εκφράσεις, όπως η $f(n) = O(n^2) + O(n)$. Έτσι, δεδομένου ότι ο όρος $O(n^2)$ υπερισχύει του όρου O(n), η παραπάνω έκφραση είναι ισοδύναμη της $f(n) = O(n^2)$. Το ίδιο σκεπτικό ισχύει και όταν το σύμβολο O εμφανίζεται σε κάποιον εκθέτη. Παραδείγματος χάριν, η έκφραση $f(n) = 2^{O(n)}$ αντιπροσωπεύει ένα άνω φράγμα της μορφής 2^{cn}, για κάποια σταθερά c. Ιδιαίτερο ενδιαφέρον παρουσιάζει η έκφραση $f(n) = 2^{O(\log n)}$, που εμφανίζεται σε ορισμένες αναλύσεις. Από την ταυτότητα $n = 2^{\log_2 n}$, έπεται ότι $n^c = 2^{c\log_2 n}$, και άρα η έκφραση $2^{O(\log n)}$ αντιπροσωπεύει ένα άνω φράγμα της μορφής n^c, για κάποια σταθερά c. Το ίδιο φράγμα, αλλά με διαφορετικό τρόπο, αναπαριστά και η έκφραση $n^{O(1)}$, αφού ο συμβολισμός O(1) δηλώνει μια συνάρτηση που δεν υπερβαίνει ποτέ κάποια πάγια σταθερά.

Συχνά εμφανίζονται επίσης φράγματα της μορφής n^c, όπου c κάποια σταθερά μεγαλύτερη του 0. Τα φράγματα αυτά ονομάζονται πολυωνυμικά. Αντίστοιχα, φράγματα της μορφής $2^{(n^\delta)}$, όπου δ οποιοσδήποτε θετικός πραγματικός αριθμός, ονομάζονται εκθετικά.

Σχετικός με τον συμβολισμό κεφαλαίου όμικρον είναι και ο λεγόμενος συμβολισμός πεζού όμικρον. Όπως είδαμε, ο συμβολισμός κεφαλαίου όμικρον υποδεικνύει ότι κάποια συνάρτηση είναι ασυμπτωτικά μικρότερη ή ίση κάποιας άλλης. Εάν θέλουμε να δηλώσουμε ότι μια συνάρτηση είναι ασυμπτωτικά

μικρότερη μιας άλλης, τότε χρησιμοποιούμε τον συμβολισμό πεζού όμικρον. Με άλλα λόγια, η διαφορά μεταξύ κεφαλαίου και πεζού όμικρον είναι αντίστοιχη με τη διαφορά μεταξύ ≤ και <.

Έστω f και g δύο συναρτήσεις από το σύνολο N στο σύνολο R+. Λέμε ότι f(n) = o(g(n)), εάν ισχύει ότι $\lim_{n\to\infty}$ f(n)/g(n)=0.

Με άλλα λόγια, η έκφραση f(n) = o(g(n)) σημαίνει ότι, για οποιονδήποτε πραγματικό αριθμό c > 0, υπάρχει ακέραιος n_0 τέτοιος ώστε f(n) < cg(n) για κάθε n ≥n_0 .

1.2 ΑΛΦΑΒΗΤΑ ΚΑΙ ΓΛΩΣΣΕΣ

Για τη μελέτη των αφηρημένων μηχανών θα ήταν χρήσιμο να γίνει αναφορά στο μοντέλο των δεδομένων που επεξεργάζονται. Η αναπαράσταση των δεδομένων γίνεται με σειρές από σύμβολα, που αναφέρονται ως *αλφάβητο*. *Αλφάβητο* Σ ονομάζεται ένα πεπερασμένο μη κενό σύνολο από σύμβολα. Ένα παράδειγμα ενός αλφάβητου μπορεί να είναι της μορφής Σ = {a, b} (Χειμωνίδης, 2001).

Η παράθεση κανενός, ενός, δύο ή περισσοτέρων, αλλά πάντως πεπερασμένου πλήθους, συμβόλων από αυτό το αλφάβητο όχι κατ' ανάγκη διαφορετικών μεταξύ τους ονομάζεται *συμβολοσειρά* (string) ή πεπερασμένη ακολουθία από σύμβολα. Για την αναπαράσταση των συμβολοσειρών χρησιμοποιούνται συνήθως τα γράμματα u, v, w, x, y, z. Η συμβολοσειρά με μηδέν σύμβολα αναπαρίσταται με **“ ”** ή **ε** ή Ø.

Το *μήκος* μιας συμβολοσειράς x είναι το πλήθος των συμβόλων τα οποία την απαρτίζουν. Για παράδειγμα |abababb| = 6, |aaa| = 3, |ε| = 0. *Αριστερό ή αρχικό απόκομμα* συμβολοσειράς είναι ό,τι έχει απομείνει από την αφαίρεση κανενός, ενός ή όλων των συμβόλων της συμβολοσειράς από δεξιά. Π.χ. αν 0101 μια συμβολοσειρά, τότε θα λέγαμε ότι οι συμβολοσειρές 010, 01, 0 είναι αριστερά αποκόμματα της συμβολοσειράς 0101(Χειμωνίδης, 2001). Όμοια *δεξί ή τελικό απόκομμα* συμβολοσειράς είναι ό,τι έχει απομείνει από την αφαίρεση κανενός, ενός ή όλων των συμβόλων της συμβολοσειράς από αριστερά. Τα δεξιά και τα αριστερά αποκόμματα είναι επίσης συμβολοσειρές.

Υποσυμβολοσειρά ονομάζουμε ό,τι έχει απομείνει από την αφαίρεση ενός δεξιού και ενός αριστερού αποκόμματος. Το σύνολο όλων των δυνατών συμβολοσειρών ενός αλφαβήτου Σ συμβολίζεται με Σ*. Π.χ. αν Σ = {0,1}, τότε Σ* = {0,1}* = {ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 100, …}

Μία συμβολοσειρά ονομάζεται *παλινδρομική*, αν διαβάζεται το ίδιο από τα αριστερά προς τα δεξιά και από τα δεξιά προς τα αριστερά. Για παράδειγμα, η συμβολοσειρά : ΣΕΡΡΕΣ είναι παλινδρομική. Έστω ένα αλφάβητο Σ. Τυπικά ορίζεται μια παλινδρομική συμβολοσειρά του Σ* ως εξής :

Η κενή συμβολοσειρά ε είναι παλινδρομική.

Αν α ∈ Σ οποιοδήποτε στοιχείο του αλφαβήτου, τότε η συμβολοσειρά α είναι παλινδρομική.

Αν η συμβολοσειρά x ∈ Σ* είναι παλινδρομική και α ∈ Σ, τότε και η αχα είναι παλινδρομική.

Καμία άλλη συμβολοσειρά δεν είναι παλινδρομική, εκτός αν προκύπτει από τα παραπάνω (Δημητρίου, 2002).

Η *αντίστροφη* μιας συμβολοσειράς x είναι η συμβολοσειρά που προκύπτει αν γράψουμε τα σύμβολα της x με αντίστροφη φορά, από δεξιά προς τα αριστερά και συμβολίζεται με xR. Έτσι αν x = αυτόματα, xR = αταμοτυα.

Για τις αντίστροφες συμβολοσειρές ισχύουν οι παρακάτω ιδιότητες :

Αν x είναι η κενή συμβολοσειρά, τότε xR = x = ε. Αν x = yα, όπου α ∈ Σ και y ∈ Σ*, τότε xR= (yα)R = αyR.

Μία *γλώσσα L* πάνω σε ένα αλφάβητο Σ είναι ένα υποσύνολο όλων των δυνατών συμβολοσειρών του Σ, δηλαδή L ⊆ Σ*. Οι γλώσσες αναπαρίστανται όπως και τα σύνολα είτε παρατάσσοντας όλα τα σύμβολά τους είτε χρησιμοποιώντας κάποια ιδιότητα των συμβολοσειρών τους. Το κενό σύνολο ονομάζεται κενή γλώσσα και είναι μία για όλα τα αλφάβητα (Δημητρίου, 2002).

Παραδείγματα γλωσσών θα μπορούσαν να είναι :

{ε}

{x ∈ {0,1}* | |x| είναι άρτιος}

{x ∈ {0,1}* | |x| ≤ 15}

{x ∈ {0,1}* | x περιέχει περισσότερα 0 απ’ ότι 1}

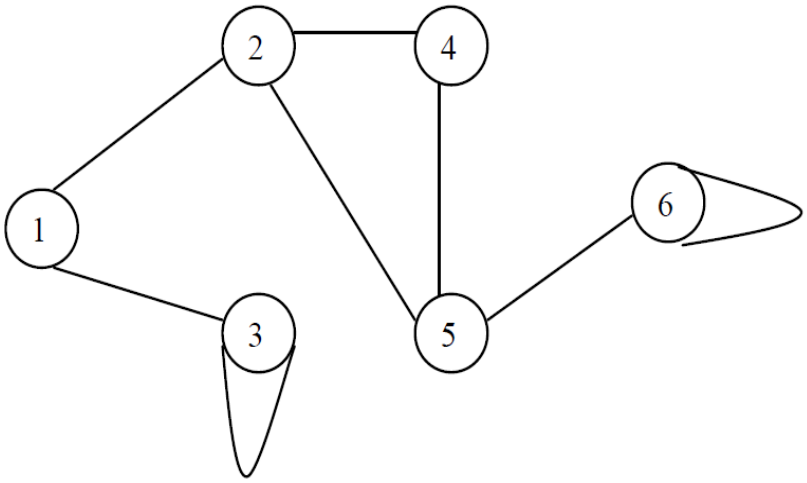
Έστω L μία γλώσσα πάνω σε ένα αλφάβητο Σ. Θα λέμε ότι δύο συμβολοσειρές x, y ∈ Σ* *διακρίνονται* ή *είναι διακρινόμενες όσον αφορά στην L*, αν

υπάρχει συμβολοσειρά z, που μπορεί να εξαρτάται από τις x και y, ώστε μία και μόνο μία από τις xz και yz να ανήκει στην L. Οι x και y είναι μη διακρινόμενες, αν δε συμβαίνει το παραπάνω, αν δηλαδή για οποιοδήποτε z, οι xz και yz ή και οι δύο ανήκουν στην L ή καμιά τους (Δημητρίου, 2002).

Εφόσον οι γλώσσες είναι απλά σύνολα συμβολοσειρών, μπορούν να δημιουργήσουν νέες γλώσσες με τη χρήση των πράξεων της ένωσης, της τομής, της διαφοράς, του συμπληρώματος, της συνένωσης, του αστεριού Kleene.

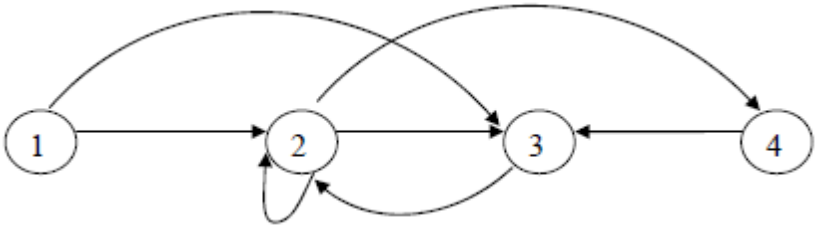
1.3 ΓΡΑΦΗΜΑΤΑ ΚΑΙ ΔΕΝΤΡΑ

Ένα *γράφημα* G(V, E) αποτελείται από ένα σύνολο από κόμβους (ή κορυφές) V και από ένα σύνολο από πλευρές (ή ακμές) E, που συνδέουν τους κόμβους μεταξύ τους (Σταμάτης, 2003). Στο γράφημα του σχήματος 1.1 η διάταξη των κόμβων δεν παίζει κανένα ρόλο.



ΣΧΗΜΑ 1.2 Γράφημα που η διάταξη των κόμβων δεν παίζει κανένα ρόλο

Στην περίπτωση κατά την οποία έχει σημασία η διάταξη των κόμβων, το γράφημα ονομάζεται κατευθυνόμενο G(V, E). Ένα παράδειγμα κατευθυνόμενου γραφήματος φαίνεται στο σχήμα 1.2.



ΣΧΗΜΑ 1 .3 Κατευθυνόμενο γράφημα

Ένα *δέντρο* είναι ένα κατευθυνόμενο γράφημα το οποίο υπακούει στους εξής κανόνες :

Υπάρχει ένας και μόνο κόμβος, που ονομάζεται ρίζα, στον οποίο δεν καταλήγει καμία ακμή.

Σε όλους τους υπόλοιπους κόμβους καταλήγει υποχρεωτικά μία και μόνο ακμή.

Τα δέντρα τα απεικονίζουμε από τη ρίζα προς τα κάτω και δε δείχνουμε τις κατευθύνσεις, γιατί υποθέτουμε ότι δείχνουν πάντα προς τα κάτω.

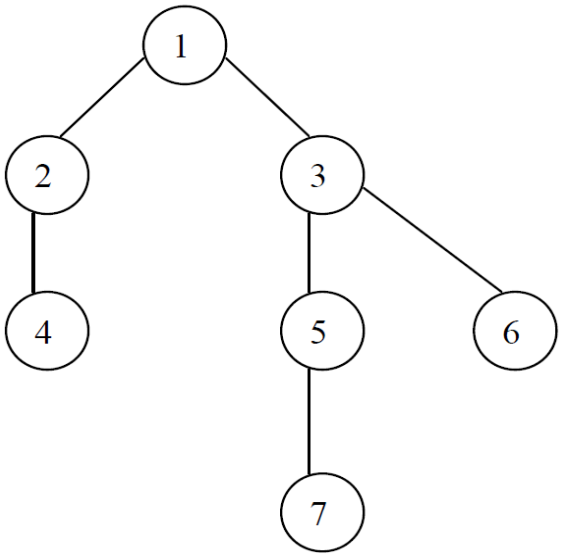
Κάθε κόμβος, εκτός από τη ρίζα, έχει από πάνω του ακριβώς έναν κόμβο που ονομάζεται *γονέας*.

Οι κόμβοι που βρίσκονται ακριβώς κάτω από έναν κόμβο και ξεκινούν από αυτόν, ονομάζονται *παιδιά* του.

Ένας κόμβος (χωρίς παιδιά) που δεν έχει κανένα μη- κενό υποδέντρο ονομάζεται *φύλλο* ή *εξωτερικός κόμβος*(Σταμάτης, 2003).

Μονοπάτι καλούμε μια ακολουθία κόμβων, όχι απαραίτητα διαφορετικών, που συνδέονται διαδοχικά μεταξύ τους με τη βοήθεια ακμών. Το *μήκος* του μονοπατιού είναι ο αριθμός των ακμών που συνδέουν τους κόμβους. *Ύψος ενός δέντρου* είναι το μήκος του μεγαλύτερου μονοπατιού από τη ρίζα προς κάποιο φύλλο, ενώ βάθος ενός κόμβου είναι το μήκος του μονοπατιού από τη ρίζα μέχρι τον κόμβο αυτό. Έτσι η ρίζα έχει πάντα βάθος μηδέν. Τέλος, ένα δέντρο ονομάζεται *δυαδικό* αν κάθε εσωτερικός κόμβος του έχει ακριβώς δύο παιδιά (Σταμάτης, 2003).

Στο επόμενο σχήμα (1.3) ο κόμβος 1 είναι η ρίζα του δέντρου, οι κόμβοι 2, 3 και 5 είναι εσωτερικοί κόμβοι ενώ οι κόμβοι 4, 6 και 7 είναι εξωτερικοί κόμβοι ή φύλλα του δέντρου. Το ύψος του συγκεκριμένου δέντρου είναι 3.



ΣΧΗΜΑ 1. 4 Ένα γράφημα δέντρου

1.5 ΓΡΑΜΜΑΤΙΚΕΣ

Στο κεφάλαιο αυτό θα πραγματοποιηθεί μια προσέγγιση μεθόδων περιγραφής γλωσσών, των *γραμματικών*. Ονομάστηκαν έτσι γιατί αρχικά εφαρμόστηκαν στη μελέτη πραγματικών γλωσσών όπως τα Αγγλικά. Με τη βοήθεια των γραμματικών είμαστε σε θέση να περιγράψουμε και να αναλύσουμε ακόμα πιο σύνθετες γλώσσες από αυτές που αναφέρθηκαν ως τώρα.

Μία *τυπική γραμματική* (*formal grammar*) αποτελείται από ένα πεπερασμένο σύνολο από τερματικά σύμβολα (τα σύμβολα των συμβολοσειρών σε μια τυπική γλώσσα), ένα πεπερασμένο σύνολο από μη τερματικά σύμβολα, ένα πεπερασμένο σύνολο από κανόνες αριστερής και δεξιάς παραγωγής και ένα αρχικό σύμβολο. Μία τέτοια γραμματική ορίζει τις τυπικές γλώσσες όλων των συμβολοσειρών που αποτελούνται μόνο από τερματικά σύμβολα και μπορούν να παραχθούν από το αρχικό σύμβολο (Δημητρίου, 2002).

Μία γραμματική ανεξάρτητη συμφραζομένων G ορίζεται ως μια 4- άδα (V, Σ, S, R) όπου:

V : ένα πεπερασμένο σύνολο μεταβλητών,

Σ : το σύνολο των τερματικών συμβόλων με $V \cap \Sigma = \emptyset$,

$S \in V$: η αρχική μεταβλητή και

R : ένα πεπερασμένο σύνολο κανόνων της μορφής $A \rightarrow \alpha$, όπου $A \in V$ και $\alpha \in (V \cup \Sigma)^*$.

Στις γραμματικές ανεξάρτητες συμφραζομένων χρησιμοποιούμε ελληνικά γράμματα για την περιγραφή των συμβολοσειρών, οι οποίες περιέχουν και τερματικά σύμβολα.

Αν το β προκύπτει από το α με αντικατάσταση κάποιας μεταβλητής, που εμφανίζεται στο αριστερό μέρος ενός κανόνα της G, θα γράφουμε $\alpha \Rightarrow_G \beta$ (ή $\alpha \Rightarrow \beta$ αν γνωρίζουμε τη γραμματική G). Για παράδειγμα αν $\alpha = \alpha_1 A \alpha_2$ και $\beta = \alpha_1 \gamma \alpha_2$, θα γράφουμε $\alpha \rightarrow \beta$ αν και μόνο αν υπάρχει ο κανόνας $A \rightarrow \gamma$ στη γραμματική. Τότε θα λέμε ότι το *α παράγει το β*. Αν το β μπορεί να παραχθεί από το α σε μηδέν ή περισσότερα βήματα δηλαδή αν $\alpha = \beta$ ή $\exists\, \kappa \geq 1$ ώστε $\alpha = \alpha_0 \Rightarrow \alpha_1 \ldots \Rightarrow \alpha_\kappa = \beta$ θα γράφουμε $\alpha \xrightarrow{*} \beta$, ενώ αν το β μπορεί να παραχθεί από το α σε ένα ή περισσότερα βήματα θα γράφουμε $\alpha \xrightarrow{+} \beta$. Κάθε ακολουθία της μορφής $\alpha 0 \Rightarrow \alpha_1 \ldots \Rightarrow \alpha_n$ ονομάζεται παραγωγή του α_n από το α_0 και λέμε ότι η παραγωγή αποτελείται από n βήματα ή ότι έχει μήκος n.

Έστω G = (V, Σ, S, R) μία γραμματική ανεξάρτητη συμφραζομένων. Η γλώσσα που παράγεται από τη γραμματική G είναι το σύνολο των συμβολοσειρών που μπορούν να παραχθούν από την αρχική μεταβλητή S. Δηλαδή :

$L(G)=\{x \in \Sigma^* \mid S \xrightarrow{*} x\}$

Αριστερή παραγωγή ονομάζεται η παραγωγή κατά την οποία μόνο το ακροαριστερό μη τερματικό σύμβολο αντικαθίσταται σε κάθε βήμα παρόμοια ορίζεται και η δεξιά παραγωγή.

Μία γραμματική ονομάζεται **διφορούμενη ή ασαφής ή διττή** αν υπάρχουν δύο ή περισσότερες αριστερές παραγωγές που παράγουν την ίδια συμβολοσειρά (Δημητρίου, 2002). Το πρόβλημα της διφορουμενολογίας μια τέτοιας γραμματικής γίνεται εντονότερο κατά τη συντακτική ανάλυση, όπου σκοπός είναι η εύρεση της ακολουθίας των κανόνων που χρησιμοποιήθηκαν για την παραγωγή της συμβολοσειράς. Μια λύση είναι να γίνει μετατροπή της διφορούμενης γραμματικής σε μία ισοδύναμη μη διφορούμενη. Η λύση αυτή όμως δεν μπορεί να δοθεί ώστε να καλυφθούν όλες οι περιπτώσεις· υπάρχουν δηλαδή γραμματικές, που όσες μετατροπές και να κάνουμε, παραμένουν ασαφείς.

Μία γραμματική ανεξάρτητη συμφραζομένων G = (V, Σ, S, R) ονομάζεται κανονική :

- αν κάθε κανόνας έχει μία από τις παρακάτω μορφές :

$A \rightarrow \alpha$
$A \rightarrow \varepsilon$

$A \rightarrow \alpha B$
όπου $\alpha \in \Sigma$ και $A, B \in V$.

- αν στο δεξιό μέρος κάθε κανόνα της υπάρχει το πολύ ένα μη τερματικό σύμβολο, και αυτό είναι πάντα στην τελευταία θέση.
- αν τα σύνολα των γλωσσών των κανονικών γραμματικών και των γλωσσών των κανονικών εκφράσεων (ή των πεπερασμένων αυτομάτων) ταυτίζονται.

ΚΕΦΑΛΑΙΟ 2

ΠΕΠΕΡΑΣΜΕΝΑ ΑΥΤΟΜΑΤΑ – ΚΑΝΟΝΙΚΕΣ ΓΛΩΣΣΕΣ

Στο προηγούμενο κεφάλαιο ασχοληθήκαμε με ορισμένες βασικές ορολογίες όπως το αλφάβητο, η συμβολοσειρά και η γλώσσα. Σε αυτό κεφάλαιο θα ασχοληθούμε με τη μελέτη των μηχανών που αναγνωρίζουν τις γλώσσες.

Τα πεπερασμένα αυτόματα ανήκουν σε μια κατηγορία τέτοιων μηχανών αποτελούν μια ικανοποιητική αναπαράσταση υπολογιστών με εξαιρετικά περιορισμένη μνήμη. Πρέπει να τονίσουμε ότι έχουν μεγάλη χρησιμότητα στην καθημερινότητά μας, καθώς αποτελούν τη βάση της λειτουργίας πολλών ηλεκτρομηχανικών συσκευών.

2.1 ΠΕΠΕΡΑΣΜΕΝΑ ΑΥΤΟΜΑΤΑ

Τα βασικά χαρακτηριστικά των πεπερασμένων αυτομάτων είναι το σύνολο των καταστάσεων, στις οποίες μπορεί να βρεθούν και η συνάρτηση, που καθορίζει, για κάθε συνδυασμό κατάστασης και εισόδου, ποια είναι η επόμενη κατάσταση. Ένα ακόμη χαρακτηριστικό των πεπερασμένων αυτομάτων είναι ότι υπάρχει περιορισμός στη διαθέσιμη μνήμη, γιατί κάθε κατάσταση του αυτόματου αντιστοιχεί και σε μια θέση μνήμης. Για το λόγο αυτό ονομάζονται και μηχανές πεπερασμένων καταστάσεων.

Ένα πεπερασμένο αυτόματο ή μηχανή πεπερασμένων καταστάσεων είναι ένα μαθηματικό μοντέλο (Δημητρίου, 2002) που ορίζεται ως μια 5- άδα

$(Q, \Sigma, q_0, \delta, F)$, όπου :

Q : πεπερασμένο σύνολο τα στοιχεία του οποίου αντιστοιχούν στις καταστάσεις της μηχανής

Σ : αλφάβητο των συμβόλων εισόδου

$q_0 \in Q$: αρχική κατάσταση

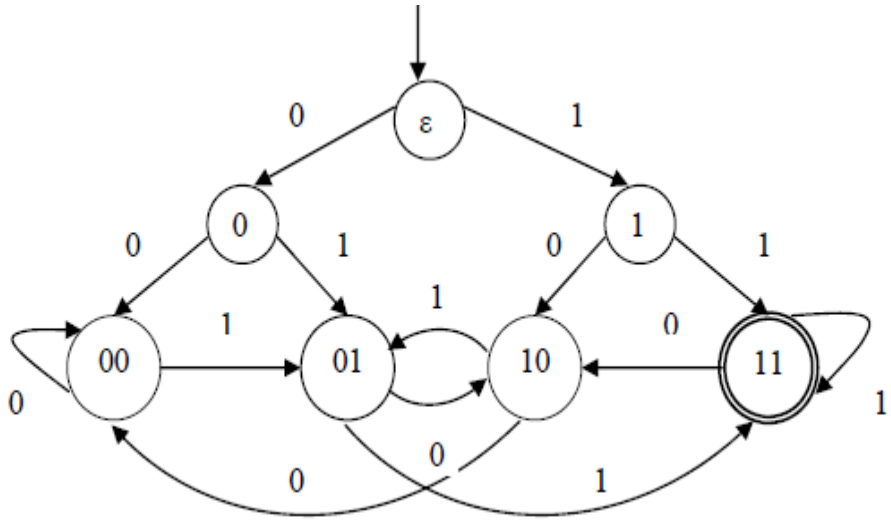
$\delta : Q \times \Sigma \rightarrow Q$ συνάρτηση μετάβασης

$F \subseteq Q$: σύνολο τελικών καταστάσεων.

Ένα πεπερασμένο αυτόματο όταν δεχθεί ως είσοδο μια συμβολοσειρά x, διαβάζει τα σύμβολα της x ένα προς ένα από τα αριστερά προς τα δεξιά. Μετά

από την ανάγνωση κάθε συμβόλου οδηγείται σε μία από τις πεπερασμένες καταστάσεις του. Μετά το τέλος της ανάγνωσης και του τελευταίου συμβόλου αν η κατάσταση, στην οποία έχει οδηγηθεί, είναι τελική κατάσταση, τότε η συμβολοσειρά ανήκει στη γλώσσα, αλλιώς απορρίπτεται. Το αυτόματο δηλαδή δέχεται ως είσοδο μια συμβολοσειρά και παράγει ως έξοδο ένα «ναι» / True ή ένα «όχι» / False.

Η αναπαράσταση ενός πεπερασμένου αυτόματου μπορεί να γίνει είτε με ένα γράφημα είτε με έναν πίνακα αλλαγής κατάστασης. Για παράδειγμα : έστω η γλώσσα L με συμβολοσειρές {x ∈ {0,1}* | x να λήγει σε 11}. Το γράφημα του ΠΑ παρουσιάζεται στο σχήμα 2.1:



ΣΧΗΜΑ 2.1 Η αναπαράσταση ενός πεπερασμένου αυτόματου

Το πάνω βέλος του γραφήματος δείχνει το σημείο εκκίνησης, το οποίο είναι το ε (στην αρχή το αυτόματο δεν έχει κανένα σύμβολο). Τα βέλη δείχνουν την κατάσταση στην οποία θα οδηγηθεί το αυτόματο μετά την είσοδο του συμβόλου που βρίσκεται από πάνω τους. Στα κυκλάκια υπάρχουν οι πεπερασμένες καταστάσεις στις οποίες μπορεί να βρεθεί το αυτόματο, ενώ με διπλό κυκλάκι συμβολίζεται η τελική κατάσταση.

		ΚΑΤΑΣΤΑΣΗ						
		ε	0	1	00	01	10	11
ΕΙΣΟΔΟΣ	0	0	00	10	00	10	00	10
	1	1	01	11	01	11	01	11

Κάθε ζεύγος κατάστασης – συμβόλου εισόδου δείχνει ποια θα είναι η επόμενη κατάσταση. Για παράδειγμα αν το αυτόματο δεχόταν ως είσοδο τη συμβολοσειρά 10011, τότε θα υπήρχε η εξής ακολουθία βημάτων :

1. Όντας το αυτόματο στην κατάσταση ε και έχοντας είσοδο το 1 θα μετέβαινε στην κατάσταση 1 (κελί (1,ε))

2. κατόπιν όντας στην κατάσταση 1 και με την είσοδο του 0 θα μετέβαινε στην κατάσταση 10 (κελί (0,1))

3. με την είσοδο του επόμενου 0 θα μετέβαινε στην κατάσταση 00 (κελί (0,10))

4. με την είσοδο του τέταρτου συμβόλου που είναι 1 θα μετέβαινε στην κατάσταση 01 (κελί (1,00)) και

5. τέλος με την είσοδο του τελευταίου συμβόλου δηλαδή του 1 από την κατάσταση 01 θα μετέβαινε στην κατάσταση 11 (κελί (1,01)) που είναι και τελική κατάσταση.

Μετά από την είσοδο της συμβολοσειράς 10011 το αυτόματο θα έδινε ως έξοδο «ναι / yes / true» δηλαδή η δοθείσα συμβολοσειρά ανήκει στη γλώσσα

$$\{x \in \{0,1\}^* \mid x \text{ να λήγει σε } 11\}.$$

Οι κανόνες σύμφωνα με τους οποίους ένα αυτόματο επιλέγει την επόμενη κατάσταση, κωδικοποιούνται στη *συνάρτηση μετάβασης*. Έτσι, αν το αυτόματο βρίσκεται στην κατάσταση $q \in Q$ και το επόμενο σύμβολο είναι το $\alpha \in \Sigma$, τότε $\delta(q, \alpha)$ είναι η μοναδική κατάσταση στην οποία μπορεί να περάσει μετά το αυτόματο (όπως φαίνεται και στον προηγούμενο πίνακα).

Για την περιγραφή της κατάστασης στην οποία μεταβαίνει το αυτόματο διαβάζοντας ολόκληρη τη συμβολοσειρά x χρησιμοποιούμε το σύμβολο $\delta^*(q, x)$.

Έστω $M = (Q, \Sigma, q0, \delta, F)$ ένα πεπερασμένο αυτόματο.

Η συνάρτηση $\delta^* : Q \times \Sigma^* \rightarrow Q$ ορίζεται αναδρομικά ως εξής :

1. Για κάθε $q \in Q$, $\delta^*(q, \varepsilon) = q$.

2. Για κάθε $q \in Q$, $x \in \Sigma^*$ και $\alpha \in \Sigma$, $\delta^*(q, x\alpha) = \delta(\delta^*(q, x), \alpha)$.

Μία συμβολοσειρά x ∈ Σ* *αναγνωρίζεται* ή *γίνεται δεκτή* από ένα αυτόματο M = (Q, Σ, q_0, δ, F), αν ξεκινώντας από την αρχική κατάσταση q_0 και διαβάζοντας τη x το M καταλήγει σε μια τελική κατάσταση, δηλαδή αν δ*(q_0, x) ∈ F (Δημητρίου, 2002). Στην αντίθετη περίπτωση θα λέμε ότι η x *απορρίπτεται* από το M. Η γλώσσα που γίνεται δεκτή ή αναγνωρίζεται από το M, συμβολίζεται με L(M) και είναι το σύνολο των συμβολοσειρών x που γίνονται δεκτές από το M. Σύμφωνα με τα προηγούμενα καταλήγουμε στο συμπέρασμα ότι μία γλώσσα είναι κανονική αν μπορούν οι συμβολοσειρές της να ταξινομηθούν σε ένα πεπερασμένο αριθμό περιπτώσεων, ώστε, αντί να θυμόμαστε ολόκληρη τη συμβολοσειρά ή μέρος αυτής, να θυμόμαστε μόνο σε ποια περίπτωση βρισκόμαστε. Στην αντίθετη περίπτωση, όταν δεν μπορούμε να ορίσουμε ένα πεπερασμένο αριθμό καταστάσεων, δηλαδή οι καταστάσεις είναι άπειρες, η γλώσσα δεν μπορεί να είναι κανονική.

2.2 ΜΗ ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΑ ΑΥΤΟΜΑΤΑ

Τα πεπερασμένα αυτόματα χαρακτηρίζονται από τη μοναδικότητα της επόμενης κατάστασης. Υπό αυτή την έννοια εκείνο που χαρακτηρίζει τα πεπερασμένα αυτόματα είναι ο ντετερμινισμός της επόμενης κίνησης.

Τα μη ντετερμινιστικά αυτόματα αποτελούνται και αυτά από ένα πεπερασμένο σύνολο καταστάσεων, ωστόσο για κάθε συνδυασμό κατάστασης και συμβόλου εισόδου υπάρχουν περισσότερες από μία επόμενες κινήσεις. Η επιλογή της επόμενης κίνησης είναι αυθαίρετη ή μη ντετερμινιστική. Η διαφορά λοιπόν των μη ντετερμινιστικών αυτομάτων με τα πεπερασμένα αυτόματα είναι πως, αντί να ρωτάμε αν το μονοπάτι που αντιστοιχεί στη συμβολοσειρά εισόδου οδηγεί σε τελική κατάσταση, ρωτάμε αν υπάρχει μονοπάτι που να οδηγεί σε μια τελική κατάσταση.

Οι γλώσσες που γίνονται δεκτές από τα μη ντετερμινιστικά αυτόματα είναι οι κανονικές γλώσσες· συνεπώς κάθε μη ντετερμινιστικό αυτόματο είναι ισοδύναμο με ένα ντετερμινιστικό και κάθε μη ντετερμινιστικό πεπερασμένο αυτόματο μετατρέπεται σε ντετερμινιστικό.

Ένα μη ντετερμινιστικό πεπερασμένο αυτόματο (Δημητρίου, 2002) είναι ένα μαθηματικό μοντέλο που ορίζεται ως μια 5-άδα (Q, Σ, q0, δ, F), όπου :

Q : πεπερασμένο σύνολο τα στοιχεία του οποίου αντιστοιχούν στις καταστάσεις της μηχανής

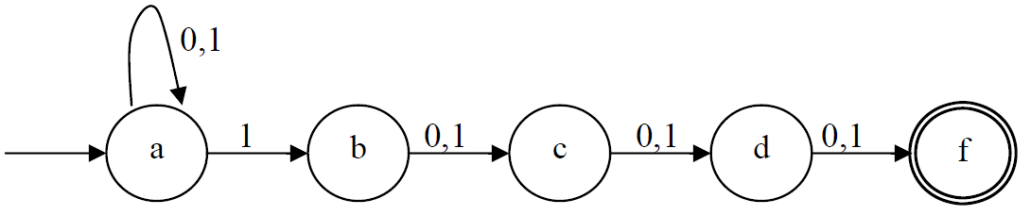
Σ : αλφάβητο των συμβόλων εισόδου

$q_0 \in Q$: αρχική κατάσταση

$\delta : Q \times \Sigma \rightarrow 2Q$ συνάρτηση μετάβασης

$F \subseteq Q$: σύνολο τελικών καταστάσεων.

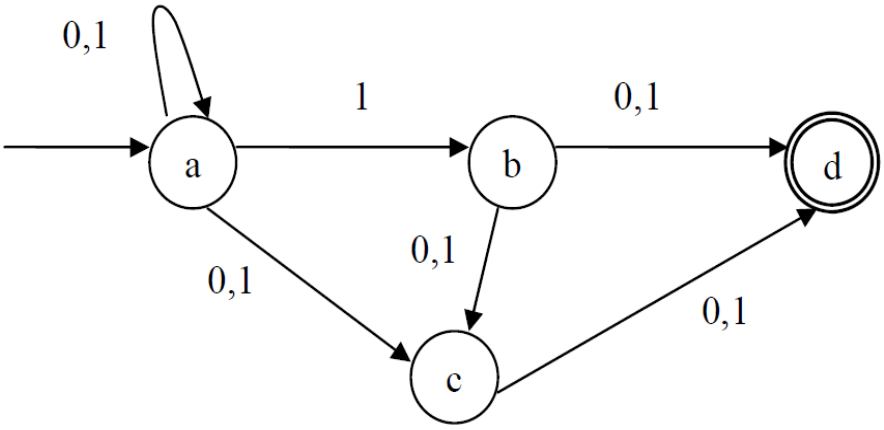
Στο σχήμα 2.2 που ακολουθεί βλέπουμε ένα μη ντετερμινιστικό αυτόματο.



ΣΧΗΜΑ 2.2 Μη ντετερμινιστικό αυτόματο

Το αυτόματο του σχήματος είναι ένα μη ντετερμινιστικό αυτόματο. Αυτό αποδεικνύεται από το γεγονός ότι όταν βρίσκεται το αυτόματο στην κατάσταση a, αν δεχθεί είσοδο 1, δεν είναι σαφές ποια θα είναι η επόμενη κατάσταση στην οποία θα εισέλθει. Μία περίπτωση είναι να παραμείνει στην κατάσταση a ή να μεταβεί στην κατάσταση b. Παρά τη διφορουμενολογία και την αυθαιρεσία ενός τέτοιου αυτομάτου αποτελεί ένα πολύ χρήσιμο εργαλείο, γιατί περιγράφει την ίδια γλώσσα που θα περιέγραφε ένα πεπερασμένο αυτόματο 32 καταστάσεων.

Στα μη ντετερμινιστικά αυτόματα για ένα συγκεκριμένο συνδυασμό κατάστασης συμβόλου εισόδου μπορεί να μην υπάρχει καμία επόμενη κατάσταση ή να υπάρχουν πολλές. Αν δ είναι η συνάρτηση μετάβασης, τότε η τιμή της δεν είναι απλά ένα μέλος του Q όπως στα πεπερασμένα αυτόματα, αλλά ένα υποσύνολο του Q, το οποίο μπορεί να είναι κενό ή να περιέχει πολλές καταστάσεις. Αυτό φαίνεται στο παρακάτω παράδειγμα ενός μη ντετερμινιστικού αυτόματου(Σχήμα 2.3) και στον πίνακα μετάβασης που ακολουθεί.



ΣΧΗΜΑ 2.3 Ένα μη ντετερμινιστικό αυτόματο με πολλές πιθανές καταστάσεις

ΕΙΣΟΔΟΣ	ΚΑΤΑΣΤΑΣΗ				
		a	b	c	d
	0	{a, c}	{c, d}	{d}	∅
	1	{a, b, c}	{c, d}	{d}	∅

Πίνακας Μετάβασης Αυτομάτου

Το παραπάνω μη ντετερμινιστικό αυτόματο όντας στην κατάσταση a, αν το επόμενο σύμβολο που θα διαβάσει είναι 1, τότε η επόμενη κατάσταση θα είναι κάποια από αυτές που ανήκουν στο σύνολο {a, b, c} και όχι κάποια συγκεκριμένη όπως στα πεπερασμένα αυτόματα. Αυτό συμβαίνει γιατί η συνάρτηση μετάβασης παίρνει τιμές στο 2^Q, το σύνολο όλων των δυνατών υποσυνόλων του Q, και όχι απλά στο Q. Συνεπώς από κάποιες καταστάσεις, όπως στην περίπτωσή μας τη d, δεν υπάρχουν δυνατές μεταβάσεις και συμβολίζονται με ∅.

Το σύνολο των καταστάσεων στις οποίες μπορεί να βρεθεί ένα αυτόματο M ξεκινώντας από μία κατάσταση q και διαβάζοντας μία συμβολοσειρά x συμβολίζεται με *δ*(q, x).*

Έστω $M = (Q, \Sigma, q_0, \delta, F)$ ένα ΜΠΑ. Η συνάρτηση $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ ορίζεται αναδρομικά ως εξής :

1. Για κάθε $q \in Q$, $\delta^*(q, \varepsilon) = \{q\}$.

2. Για κάθε $q \in Q$, $x \in \Sigma^*$ και $\alpha \in \Sigma$, $\delta^*(q, x\alpha) = (\delta^*(q, x), \alpha) =$

$$\bigcup_{p \in \delta^*(q,x)} \delta(p, \alpha)$$

Έτσι, μία συμβολοσειρά x ∈ Σ* αναγνωρίζεται ή γίνεται δεκτή από ένα μη ντετερμινιστικό αυτόματο M = (Q, Σ, q_0, δ, F), αν δ*(q_0, x) ∩ F ≠ ∅. Η γλώσσα που γίνεται δεκτή ή αναγνωρίζεται από το M, συμβολίζεται με L(M) και είναι το σύνολο των συμβολοσειρών x που γίνονται δεκτές από το M.

2.3 ΔΙΑΦΟΡΕΣ ΜΠΑ – ΠΑ

Τα αυτόματα είναι εργαλεία που αναγνωρίζουν γλώσσες. Ένα από τα βασικά αποτελέσματα της θεωρίας των αυτομάτων είναι πως η δυνατότητα αναγνώρισης γλωσσών στα ΜΠΑ δεν είναι μεγαλύτερη απ' ό,τι στα ΠΑ.

Οι διαφορές των δύο ειδών αυτομάτων, όπως σημειώθηκε και παραπάνω, είναι:

- Στα μη ντετερμινιστικά αυτόματα ο συνδυασμός κατάστασης-συμβόλου εισόδου οδηγεί σε καμία ή περισσότερες επόμενες καταστάσεις, ενώ ένα πεπερασμένο αυτόματο θα μπορούσε να θεωρηθεί σαν ένα μη ντετερμινιστικό, στο οποίο οι επόμενες καταστάσεις δε θα ήταν καμία ή περισσότερες από μία, αλλά πάντα μία. Συνεπώς, τα πεπερασμένα και τα μη ντετερμινιστικά αυτόματα αναγνωρίζουν τις ίδιες γλώσσες.
- Τα μη ντετερμινιστικά αυτόματα είναι πιο περιληπτικά. Γενικά, τα μη ντετερμινιστικά αυτόματα περιλαμβάνουν λιγότερες καταστάσεις και λιγότερες μεταβάσεις, γεγονός που κάνει τη χρήση τους ευκολότερη.
- Ο τρόπος με τον οποίο λειτουργεί ένα μη ντετερμινιστικό αυτόματο είναι πολύ κοντά στον ανθρώπινο τρόπο σκέψης. Από την άλλη μεριά, τα πεπερασμένα αυτόματα μπορούν να θεωρηθούν ως αλγεβρικές δομές και μπορούμε να τα χειριστούμε με πολύ αποδοτικά αλγεβρικά εργαλεία.

2.4 ΚΑΝΟΝΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

Υπάρχουν δύο τρόποι για να αναπαρασταθεί μια γλώσσα. Αν είναι πεπερασμένη, η αναπαράστασή της μπορεί να γίνει με την απλή παράθεση των συμβολοσειρών της. Αν όμως αποτελείται από άπειρο αριθμό συμβολοσειρών, τότε η αναπαράστασή της γίνεται με ένα συνδυασμό των βασικών πράξεων και η γλώσσα ονομάζεται *κανονική*. Η περιγραφή μιας κανονικής γλώσσας γίνεται με τη

χρήση των συμβόλων {, }, +, * και ονομάζεται *κανονική έκφραση (regular expression)* (Δημητρίου, 2002). Οι κανονικές εκφράσεις, δηλαδή, είναι εκφράσεις που αποτελούνται από χαρακτήρες και σύμβολα και μας επιτρέπουν να ορίσουμε τη σχέση ενός συγκεκριμένου συνόλου.

Μία κανονική έκφραση σ' ένα αλφάβητο Σ, αλλά και η αντίστοιχη γλώσσα, ορίζεται ως εξής:

1. Ø είναι η κανονική έκφραση που αντιστοιχεί στην κενή γλώσσα Ø.
2. ε είναι η κανονική έκφραση που αντιστοιχεί στη γλώσσα{ε}.
3. Για κάθε σύμβολο α ∈ Σ, α είναι η κανονική έκφραση που αντιστοιχεί στη γλώσσα {α}.
4. Αν r και s είναι εκφράσεις που συμβολίζουν τις γλώσσες Lr και Ls, τότε :
   - η (r + s) είναι η κανονική έκφραση που περιγράφει την Lr ∪ Ls
   - η (rs) είναι η κανονική έκφραση που περιγράφει την LrLs
   - η (r*) είναι η κανονική έκφραση που περιγράφει την Lr*
5. Τίποτα άλλο δεν είναι κανονική έκφραση, εκτός αν προκύπτει από τους παραπάνω κανόνες (Δημητρίου, 2002). Μία γλώσσα πάνω σε ένα αλφάβητο Σ θα ονομάζεται *κανονική*, αν υπάρχει κάποια κανονική έκφραση που αντιστοιχεί σε αυτή.

Οι *ιδιότητες* των κανονικών εκφράσεων είναι οι εξής : Αν r, s, t κανονικές εκφράσεις, ισχύουν:

$$r|s = s|r$$
$$r|(s|t) = (r|s)|t$$
$$r(st) = (rs)t$$
$$r(s|t) = rs|\ rt$$
$$(r|s)t = rt|\ st$$
$$\varepsilon r = r\varepsilon = r$$

Η ένωση συμπεριφέρεται ως πρόσθεση και η παράθεση ως πολλαπλασιασμός.

Σύμφωνα με τον ορισμό μπορεί να υπάρχουν άπειρες εκφράσεις που να αναπαριστούν την ίδια γλώσσα. Για παράδειγμα (0(01)) και ((00)1).

Εφόσον οι πράξεις της ένωσης και της συνένωσης είναι προσεταιριστικές, μπορούμε να παραλείψουμε τις παρενθέσεις. Ωστόσο δεν μπορούμε να τις εξαλείψουμε εντελώς, γιατί θα προέκυπτε πρόβλημα ερμηνείας των εκφράσεων

όπως στο παράδειγμα : η έκφραση 0 +10 θα μπορούσε να ερμηνευτεί σαν (0+1) 0 ή σαν 0 + (10).

Στο σημείο αυτό θα πρέπει να γίνει αναφορά στην προτεραιότητα των πράξεων. Η σειρά κατάταξης των πράξεων ανάλογα με την προτεραιότητά τους και ξεκινώντας από αυτή με τη μεγαλύτερη προτεραιότητα είναι :

1. *

2. συνένωση

3. ένωση

Οι εκφράσεις που περιλαμβάνουν πράξεις ίδιας προτεραιότητας παρά την εισαγωγή της έννοιας της προτεραιότητας, εξακολουθούν να μένουν διφορούμενες. Για την πλήρη άρση της διφορουμενολογίας εισάγεται η έννοια της προσεταιριστικότητας ενός ή περισσότερων πράξεων με την ίδια προτεραιότητα. Η προσεταιριστικότητα εκφράζεται με τους όρους :

2 από αριστερά προς δεξιά και

3 από δεξιά προς αριστερά

Στην προκειμένη περίπτωση οι πράξεις θα διαβάζονται από τα αριστερά προς τα δεξιά.

^	Αρχή της γραμμής
$	Τέλος της γραμμής
.	Οποιοδήποτε γράμμα
[abc]	Ένα από τα γράμματα a, b, ή c
[a-z]	Ένα από τα γράμματα a μέχρι z
[^abc]	Οποιοδήποτε γράμμα *εκτός* από τα a, b, και c.
*Έκφραση**	Η *έκφραση* μηδέν ή περισσότερες φορές
Έκφραση+	Η *έκφραση* μία ή περισσότερες φορές
Έκφραση?	Η *έκφραση* μία ή καμία φορά
Έκφραση1|*Έκφραση1*	Η *έκφραση1* ή η *έκφραση2*

2.5 ΚΑΝΟΝΙΚΕΣ ΠΡΑΞΕΙΣ

Στις προηγούμενες παραγράφους ορίσαμε τα πεπερασμένα αυτόματα και τις κανονικές γλώσσες σ'αυτήν την υποενότητα, θα μελετήσουμε τις ιδιότητες τους.

2.5.1 Η ΠΡΑΞΗ ΤΗΣ ΕΝΩΣΗΣ

Αν έχουμε τις γλώσσες L1 και L2 τότε η *ένωσή* τους θα συμβολίζεται ως L1 ∪ L2 (Κατωπόδης, 2000). Για τις συμβολοσειρές που θα προκύψουν από την ένωση των δύο αυτών γλωσσών θα ισχύει : L1 ∪ L2 = {x | x ∈ L1 ή x ∈ L2 } Έτσι θα μπορούν να περιέχουν στοιχεία από τη γλώσσα L1 ή από τη γλώσσα L2 ή στοιχεία που ανήκουν και στις δύο γλώσσες συγχρόνως. Για παράδειγμα αν L1 = {1, 2, 5, 6, 9} και L2 = {1, 3, 4, 5, 7} τότε: L1 ∪ L2 = {1, 2, 3, 4, 5, 6, 7, 9}. Για την πράξη της ένωσης ισχύουν οι παρακάτω βασικές ιδιότητες. Αν υποθέσουμε ότι L1, L2, L3 είναι τρεις γλώσσες τότε :

1. L1 ∪ L1 = L1
2. L1 ∪L2 = L2 ∪L1 (αντιμεταθετική ιδιότητα)
3. L1 ∪Ø = L1
4. L1 ∪ (L2 ∪L3) = (L1 ∪L2) ∪L3 (προσεταιριστική ιδιότητα)
5. Αν L1 ∪L2 τότε L1 ∪L2 = L2
6. Είναι L1 ⊆L1 ∪L2 και L2 ⊆L1 ∪L2

2.5.2 Η ΠΡΑΞΗ ΤΗΣ ΤΟΜΗΣ

Αν έχουμε τις γλώσσες L1 και L2, τότε η *τομή* τους θα συμβολίζεται ως L1 ∩ L2 (Κατωπόδης, 2000). Για τις συμβολοσειρές που θα προκύψουν από την τομή των δύο αυτών γλωσσών θα ισχύει :

L1 ∩ L2 = {x | x ∈ L1 και x ∈ L2 }

Έτσι θα μπορούν να περιέχουν στοιχεία που ανήκουν στη γλώσσα L1 και στη γλώσσα L2 συγχρόνως. Για παράδειγμα αν L1 = {1, 2, 5, 6, 9} και L2 = {1, 3, 4, 5, 7} τότε: L1 ∩ L2 = {1, 5}. Για την πράξη της τομής ισχύουν οι παρακάτω βασικές ιδιότητες. Αν υποθέσουμε ότι L1, L2, L3 είναι τρεις γλώσσες τότε :

1. L1 ∩ L1 = L1
2. L1 ∩ L2 = L2 ∩ L1 (αντιμεταθετική ιδιότητα)
3. L2 ∩ Ø = Ø

4. L1 ∩ (L2 ∩ L3) = (L1 ∩ L2) ∩ L3 (προσεταιριστική ιδιότητα)

5. Αν L1 ⊆ L2 τότε L1 ∩ L2 = L1

6. Είναι L1 ∩ L2 ⊆ L1 και L1 ∩ L2 ⊆ L2

2.5.3 Η ΠΡΑΞΗ ΤΗΣ ΔΙΑΦΟΡΑΣ

Αν έχουμε τις γλώσσες L1 και L2, τότε η *διαφορά* τους θα συμβολίζεται ως L1-L2 (Κατωπόδης, 2000). Για τις συμβολοσειρές που θα προκύψουν από τη διαφορά των δύο αυτών γλωσσών θα ισχύει : L1-L2 = {x | x ∈ L1 και x ∉ L2} Έτσι θα μπορούν να περιέχουν στοιχεία από τη γλώσσα L1 τα οποία δεν ανήκουν στη γλώσσα L2. Για παράδειγμα αν L1 = {1, 2, 5, 6, 9} και L2 = {1, 3, 4, 5, 7} τότε: L1-L2 = {2, 6, 9}. Για την πράξη της διαφοράς ισχύουν οι παρακάτω βασικές ιδιότητες.

1. L1 - L1 = Ø
2. L1 - Ø = L1
3. Ø - L1 = Ø
4. (L1 - L2) - L3 = L1 - (L2 ∪ L3) = (L1 - L3) - L2
5. L1 - L2 ⊂ L1

2.5.4 Η ΠΡΑΞΗ ΤΟΥ ΣΥΜΠΛΗΡΩΜΑΤΟΣ

Αν έχουμε τη γλώσσα L1 τότε το συμπλήρωμά της ως προς το Σ* θα συμβολίζεται ως $\bar{L}_1$ (Κατωπόδης, 2000). Για τις συμβολοσειρές που θα προκύψουν από το συμπλήρωμα θα ισχύει : $\bar{L}_1$ = Σ* - L = {x | x ∈ Σ* και x ∉ L1 } Έτσι θα μπορούν να περιέχουν στοιχεία που ανήκουν στο Σ* και δεν ανήκουν στη γλώσσα L . Για παράδειγμα αν L1 = {1, 2, 5, 6, 9} και Σ = {1, 2, 3, 4, 5, 6, 7, 8, 9} τότε : $\bar{L}_1$ = {3, 4, 7, 8, …}.

Για την πράξη του συμπληρώματος ισχύουν οι παρακάτω βασικές ιδιότητες.

1. $\bar{\bar{L}}_1 = L_1$
2. $\overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$
3. $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$
4. $L_1 \cup \overline{L_1} = \Sigma *$
5. $L_1 \cap \overline{L_1} = \emptyset$
6. $\overline{\phi} = \Sigma *$

7. $\overline{\Sigma *} = \emptyset$
8. $L_1 \subset L_2$ τότε $\overline{L_2} \subseteq \overline{L_1}$
9. $L_1 - L_2 = L_1 \cap \overline{L_2}$
10. Αν $L_1 = L_2$ τότε $\overline{L_1} = \overline{L_2}$

2.5.5 Η ΠΡΑΞΗ ΤΗΣ ΣΥΝΕΝΩΣΗΣ

Αν έχουμε τις γλώσσες L1 και L2, τότε η συνένωσή τους θα συμβολίζεται ως L1 ο L2 ή απλά L1L2 (Δημητρίου, 2002). Για τις συμβολοσειρές που θα προκύψουν από τη συνένωση των δύο αυτών γλωσσών θα ισχύει :

L1L2 = {xy | x ∈ L1 και y ∈ L2 }

Για παράδειγμα αν L1 = {0,1} και L2 = {a, b,c} τότε :

L1L2 = {0a, 0b, 0c, 1a, 1b, 1c}

Προφανώς κατά τη συνένωση της γλώσσας L3 = {x ∈ {0,1}* | |x| ≤ 5} και L4 = {ε} ισχύει : L3 L4 = x ε = x

Παρόμοια για κάθε γλώσσα L ⊆ Σ* και k ≥ 0 L k = 647k 48 LLL...LL όπου κάθε συμβολοσειρά της νέας γλώσσας προκύπτει από τη συνένωση οποιωνδήποτε k συμβολοσειρών της αρχικής L.

Προφανώς L_0 ={ε}.

2.5.6 ΑΣΤΕΡΙ KLEENE

Μια άλλη πράξη είναι το *αστέρι Kleene* μιας γλώσσας, που συμβολίζεται ως L* και ορίζεται :

$$L^* = \bigcup_{k=0}^{\infty} L^k = \{x \in \Sigma^* \mid x = x_1\, x_2\, x_3 \ldots x_k, \text{ για } k \geq 0 \text{ και } x_1, x_2, x_3, \ldots x_k \in L\}$$

δηλ. η συνένωση των 0 ή περισσότερων παραθέσεων της L (Δημητρίου, 2002).

Ως *L+* της γλώσσας L συμβολίζουμε το σύνολο όλων των συμβολοσειρών που προκύπτουν από τη συνένωση μιας ή περισσοτέρων συμβολοσειρών της L και ορίζεται ως :

$$L^+ = \bigcup_{k=1}^{\infty} L^k = \{x \in \Sigma^* \mid x = x_1\, x_2\, x_3 \ldots x_k,\ \text{για } k \geq 1 \text{ και } x_1,\ x_2,$$

$$x_3, \ldots x_k \in L\}$$

Μία γλώσσα είναι μη κανονική όταν δεν υπάρχει πεπερασμένο αυτόματο (ντετερμινιστικό ή μη) που να την αναγνωρίζει. Δεν υπάρχει γενική μέθοδος απόδειξης ότι μία γλώσσα δεν είναι κανονική. Για να αποδείξουμε ότι μια γλώσσα δεν είναι κανονική, χρησιμοποιούμε κάποιες βασικές ιδιότητες που ισχύουν για τις κανονικές γλώσσες.

Ιδιότητα 1 Καθώς διαβάζουμε μια λέξη (οσοδήποτε μεγάλου μήκους) από αριστερά προς τα δεξιά, το ποσό της μνήμης, που χρειάζεται για να αποφασιστεί στο τέλος αν η λέξη ανήκει ή όχι στη γλώσσα, πρέπει να είναι φραγμένο και εξαρτώμενο από τη γλώσσα (δηλαδή να μην εξαρτάται από τη συγκεκριμένη λέξη). π.χ. κάτι τέτοιο δεν ισχύει για τις λέξεις της γλώσσας $\{a^n b^n : n>0\}$ (Ρεφανίδης, 2002).

Ιδιότητα 2 Οι κανονικές γλώσσες με άπειρο αριθμό λέξεων (προφανώς μη φραγμένου μήκους) έχουν άπειρα υποσύνολα με κάποια απλή επαναληπτική δομή, που προκύπτει από την εφαρμογή του αστεριού Kleene (*), στην αντίστοιχη κανονική έκφραση ή από ένα κύκλο στο διάγραμμα του αντίστοιχου αυτόματου. π.χ. η γλώσσα $\{\alpha^n$: ο n είναι πρώτος αριθμός$\}$ δεν είναι κανονική (Ρεφανίδης, 2002).

Τέλος, ένα πολύ χρήσιμο εργαλείο για την απόδειξη της μη κανονικότητας μιας γλώσσας είναι το Λήμμα Άντλησης για κανονικές γλώσσες.

Το Λήμμα Άντλησης είναι το εξής : Έστω L μια άπειρη κανονική γλώσσα. Τότε υπάρχει ένας αριθμός n έτσι ώστε κάθε $x \in L$, με $|x| \geq n$, μπορεί να γραφεί στη μορφή x = wyz, όπου για τις συμβολοσειρές y, z και w ισχύει :

$|yz| \leq n$,
$y \neq \varepsilon$ και
$yz^m w \in L$, για κάθε $m \geq 0$.

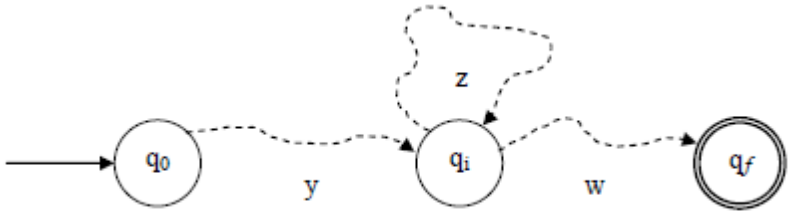
Απόδειξη :

Έστω M = (Q, Σ, δ, q_0, F) ένα αυτόματο n καταστάσεων, το οποίο αναγνωρίζει την L. Έστω τώρα μία οποιαδήποτε συμβολοσειρά x μήκους $\ell \geq n$, που αναγνωρίζεται από το M. Ας γράψουμε τώρα τη x στη μορφή $x = a_1 a_2 \ldots a_\ell$ και

ας είναι q_i η κατάσταση που πηγαίνει το αυτόματο διαβάζοντας τα πρώτα i σύμβολα της x, δηλαδή $q_i = \delta^*(q_0, a_1 a_2 \dots a_t)$.

Καθώς το αυτόματο αποτελείται από n καταστάσεις, η εφαρμογή της Αρχής των Περιστεροφωλιών[1] μας λέει ότι οι n + 1 καταστάσεις $q_0, q_1, \dots, q_n$ δεν μπορεί να είναι όλες διαφορετικές μεταξύ τους. Θα υπάρχουν λοιπόν δείκτες i, j, όπου $i \neq j$, έτσι ώστε $q_i = q_j$. Άρα η μη κενή συμβολοσειρά $a_{i+1} \dots a_j$ θα οδηγεί το αυτόματο από την κατάσταση q_i ξανά πίσω στην qi.

Για να απλοποιήσουμε τις εκφράσεις θέτουμε $y = a_1 \dots a_i$, $z = a_{i+1} \dots a_j$ και $w = a_{j+1} \dots a_l$. Εφόσον $\delta^* = (q_i, z) = q_i$, η z μπορεί να αφαιρεθεί από τη x ή να επαναληφθεί πολλές φορές και το αυτόματο να αναγνωρίζει τη νέα x. Αυτό φαίνεται καλύτερα στο σχήμα 2.4.



ΣΧΗΜΑ 2.4 Η κατάσταση q_i

Το αρχικό τμήμα y οδηγεί το M από την q_0 στην q_i, το z από την q_i ξανά πίσω στην q_i και το w από την q_i στην τελική κατάσταση. Εφόσον ο κύκλος z μπορεί να χρησιμοποιηθεί όσες φορές θέλουμε, συμπεραίνουμε ότι $\delta^* = (q_0, yz^m w) = q_f$, για κάθε $m \geq 0$ ή αλλιώς $yz^m w \in L$, για κάθε $m \geq 0$. Το θεώρημα προκύπτει αρκεί να παρατηρήσουμε ότι $|yz| \leq n$ και $|y| \neq 0$.

Με άλλα λόγια, υπάρχουν ορισμένα σημεία μέσα σε ορισμένες λέξεις (και σίγουρα στις λέξεις με μεγάλο μήκος) στα οποία μπορεί να εισαχθεί επανειλημμένα μια υπολέξη (η z με βάση τον ορισμό του θεωρήματος), χωρίς αυτό να επηρεάσει το γεγονός ότι η αρχική λέξη γίνεται δεκτή.

Π.χ. Η γλώσσα $L=\{\alpha^m b^m : m \geq 0\}$ δεν είναι κανονική.

[1] Αρχή της Περιστεροφωλιάς : Έστω δύο μη κενά σύνολα A, B τέτοια ώστε |A| > |B|. Τότε δεν μπορεί να υπάρχει ένα προς ένα συνάρτηση από το A στο B. Με άλλα λόγια, αν προσπαθήσουμε να ταιριάξουμε τα στοιχεία του A με αυτά του B αναγκαστικά δύο στοιχεία από το A θα απεικονιστούν στο ίσιο στοιχείο. Η αρχή αυτή λέγεται *Αρχή της Περιστεροφωλιάς* και χρησιμοποιείται συχνά στην πράξη όταν θέλουμε να καταλήξουμε σε κάποια αντίφαση : αν προσπαθήσουμε να βάλουμε n «περιστέρια» σε m < n «φωλιές», κάποια στιγμή θα αναγκαστούμε να βάλουμε περισσότερα από ένα σε μία «φωλιά».

Προσπαθούμε να φτιάξουμε λέξεις της μορφής yz^mw που να ανήκουν στη γλώσσα για κάθε m. Υπάρχουν τρεις περιπτώσεις :

1. Το z να αποτελείται μόνο από α.

2. Το z να αποτελείται μόνο από b.

3. Το z να αποτελείται από μερικά α και στη συνέχεια από μερικά b. Και στις τρεις περιπτώσεις μπορεί εύκολα να φανεί ότι δεν είναι δυνατόν λέξεις αυτής της μορφής να ανήκουν στη γλώσσα L για κάθε m, είτε γιατί δε θα είναι ίσος ο αριθμός των α και των b(πρώτη και δεύτερη περίπτωση) είτε γιατί θα υπάρχουν ανακατωμένα α και b (τρίτη περίπτωση) (Δημητρίου, 2002).

ΚΕΦΑΛΑΙΟ 3

ΑΥΤΟΜΑΤΑ ΣΤΟΙΒΑΣ – ΓΛΩΣΣΕΣ ΑΝΕΞΑΡΤΗΤΕΣ ΣΥΜΦΡΑΖΟΜΕΝΩΝ

Εκτός από τις κανονικές εκφράσεις που είναι μέθοδοι παραγωγής κανονικών γλωσσών, υπάρχουν κι άλλες μέθοδοι παραγωγής γλωσσών, όχι πάντα κανονικών, όπως οι γραμματικές ανεξάρτητες συμφραζομένων. Κατά συνέπεια τα πεπερασμένα αυτόματα αλλά και τα μη ντετερμινιστικά δεν επαρκούν για να αναγνωρίσουν αυτές τις γλώσσες, μιας και διαθέτουν περιορισμένη μνήμη, γεγονός που αποδεικνύεται από τον πεπερασμένο αριθμό καταστάσεων. Τη λύση στο πρόβλημα της αναγνώρισης των μη κανονικών γλωσσών έρχονται να δώσουν τα αυτόματα στοίβας, τα οποία διαθέτουν μνήμη με τη μορφή της στοίβας.

Στοίβα ή σωρός είναι μία λίστα στην οποία νέα στοιχεία μπορούν να προστεθούν και να αφαιρεθούν μόνο από τη μία άκρη της, που ονομάζεται κορυφή της στοίβας (Σταμάτης, 2003). Συχνά μία στοίβα αναφέρεται και σαν μία λίστα τύπου *LIFO* (Last -In -First -Out), για να δηλώνεται έτσι ρητά η βασική της ιδιότητα, ότι το στοιχείο που θα προστεθεί τελευταίο στη στοίβα θα αφαιρεθεί πρώτο ή ισοδύναμα το πρώτο στοιχείο της στοίβας αναγκαστικά πρέπει να αφαιρεθεί τελευταίο.

3.1 ΓΛΩΣΣΕΣ ΑΝΕΞΑΡΤΗΤΕΣ ΣΥΜΦΡΑΖΟΜΕΝΩΝ

Μία γλώσσα L θα λέγεται **ανεξάρτητη συμφραζομένων**, αν υπάρχει γραμματική ανεξάρτητη συμφραζομένων G έτσι ώστε L = L(G). Οι γραμματικές αυτού του είδους ονομάζονται ανεξάρτητες συμφραζομένων για τον εξής λόγο : έστω η συμβολοσειρά 01 A 01, η οποία είναι ένα ενδιάμεσο βήμα στην παραγωγή του 01 101 01. Οι συμβολοσειρές 01 και 01 που περιβάλλουν το A = 101 είναι τα συμφραζόμενα του A στη συμβολοσειρά. Ο κανόνας A → 1A1 επισημαίνει την αντικατάσταση του A με τη συμβολοσειρά 1A1 ανεξάρτητα από τις περιβάλλουσες συμβολοσειρές, ανεξάρτητα δηλαδή από τα συμφραζόμενα.

3.1.1 ΙΔΙΟΤΗΤΕΣ ΓΡΑΜΜΑΤΙΚΩΝ ΑΝΕΞΑΡΤΗΤΕΣ ΣΥΜΦΡΑΖΟΜΕΝΩΝ

Σε ότι αφορά τις γραμματικές ανεξάρτητες συμφραζομένων ισχύουν κάποιες ιδιότητες :

Η ιδιότητα της κλειστότητας ως προς τις πράξεις της ένωσης, της συνένωσης και του αστεριού Kleene (δεν ισχύει αυτή η ιδιότητα για τις πράξεις της τομής και του συμπληρώματος)

Ένωση

Έστω G1 = $(V_1, \Sigma_1, S_1, R_1)$ και $G_2 = (V_2, \Sigma_2, S_2, R_2)$. Η γραμματική G για την οποία ισχύει $L(G)=L(G_1) \cup L(G_2)$ ορίζεται ως εξής :

$V = V_1 \cup V_2 \cup \{S\}$,
$R = R_1 \cup R_2 \cup \{S \rightarrow S_1 \mid S_2\}$

Συνένωση

Έστω $G_1 = (V_1, \Sigma_1, S_1, R_1)$ και $G_2 = (V_2, \Sigma_2, S_2, R_2)$. Η γραμματική G για την οποία ισχύει $L(G)=L(G_1)\ L(G_2)$ ορίζεται ως εξής :

$V = V_1 \cup V_2 \cup \{S\}$,
$R = R_1 \cup R_2 \cup \{S \rightarrow S_1\ S_2\}$

Αστέρι Kleene

Έστω $G_1 = (V_1, \Sigma_1, S_1, R_1)$. Η γραμματική G για την οποία ισχύει

$L(G)=L(G_1)^*$ ορίζεται ως εξής:
$R = R_1 \cup \{S \rightarrow S_1\ S \mid \varepsilon\}$

Θεώρημα Λήμματος Άντλησης για γραμματικές

Για τις γλώσσες ανεξάρτητες συμφραζομένων υπάρχει ένα λήμμα παρόμοιο με εκείνο του λήμματος άντλησης, στο οποίο θα γίνει εκτενής αναφορά στο επόμενο κεφάλαιο, το οποίο χρησιμεύει στην απόδειξη της ιδιότητας της περιοδικότητας αυτών των γλωσσών. Στο συγκεκριμένο λήμμα που χρησιμοποιείται στις γραμματικές, υπάρχουν δύο συμβολο- σειρές οι οποίες μπορούν να επαναληφθούν όσες φορές επιθυμούμε και το αποτέλεσμα να ανήκει και πάλι στη γλώσσα.

Έστω G μια γραμματική ανεξάρτητη συμφραζομένων τότε υπάρχει ένας αριθμός K (που εξαρτάται από την G) τέτοιος ώστε κάθε συμβολοσειρά $w \in L(G)$ με $|w| > K$ να μπορεί να γραφεί ως $w = uvxyz$, $|v|+|y|>0$ και $uv^nxy^nz \in L(G)$ για κάθε $n \geq 0$.

3.2 ΑΥΤΟΜΑΤΑ ΣΤΟΙΒΑΣ

Τα αυτόματα στοίβας διαθέτουν ένα πεπερασμένο σύνολο καταστάσεων και ένα άπειρο σωρό που λειτουργεί ως μνήμη. Για να γίνει πιο σαφής η λειτουργία ενός τέτοιου αυτομάτου παρατίθεται το παρακάτω παράδειγμα.

Έστω η γλώσσα $L = \{0^n1^n \mid n \geq 0\}$ που αναγνωρίζεται από το αυτόματο στοίβας και εισάγεται η συμβολοσειρά x = 000111. Η λειτουργία του αυτομάτου θα είναι η εξής : το αυτόματο θα διαβάσει αρχικά το πρώτο 0 και θα το τοποθετήσει στη στοίβα. Κατόπιν θα διαβάσει και τα επόμενα 0 τα οποία ομοίως θα τοποθετήσει στη στοίβα. Όταν θα φτάσει στην εισαγωγή του 1, θα το διαβάσει και θα αφαιρέσει από τη στοίβα το τελευταίο εισαχθέν 0, δηλαδή διαβάζοντας τον τέταρτο χαρακτήρα θα αφαιρέσει τον τρίτο χαρακτήρα που είναι 0. Έπειτα θα διαβάσει τον πέμπτο χαρακτήρα που είναι 1 και εφόσον αυτός δεν είναι 0, θα αφαιρέσει από τη στοίβα το δεύτερο εισαχθέν 0. Ομοίως θα συμβεί και με το τελευταίο 1. Η έξοδος του αυτομάτου μετά και την εισαγωγή του τελευταίου χαρακτήρα θα είναι «ναι», δηλαδή η συμβολοσειρά αναγνωρίζεται από το αυτόματο και συνεπώς ανήκει στη γλώσσα. Οι μεταβάσεις που συμβαίνουν φαίνονται παρακάτω :

- $\delta(q0, 0, Z) = \{(q1, 0Z)\}$
- $\delta(q1, 0, 0) = \{(q1, 00)\}$
- $\delta(q1, 1, 0) = \{(q2, \varepsilon)\}$
- $\delta(q2, 1, 0) = \{(q2, \varepsilon)\}$
- $\delta(q2, \varepsilon, Z) = \{(q0, \varepsilon)\}$

Σύμφωνα με το παράδειγμα φαίνεται ότι κάθε κίνηση του αυτομάτου εξαρτάται από τρεις παραμέτρους : την τρέχουσα κατάσταση, το σύμβολο εισόδου και το σύμβολο που βρίσκεται στην κορυφή της στοίβας. Επιπλέον παρατηρούμε ότι τα αυτόματα στοίβας συνήθως δρουν μη ντετερμινιστικά εφόσον «αποφασίζουν» για το πότε έχουν διαβάσει το πρώτο μισό της εισαγόμενης συμβολοσειράς αυθαίρετα. Αυτή η αυθαιρεσία μας οδηγεί στο συμπέρασμα πως η συνάρτηση μετάβασης θα είναι ένα σύνολο από πιθανές επόμενες καταστάσεις και θα είναι της μορφής $\delta(q, \alpha, Z) = \{(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_n, \alpha_n)\}$ όπου p_i : νέα κατάσταση και α_i η συμβολοσειρά που αντικαθιστά το σύμβολο Z στην κορυφή της στοίβας.

Ένα *αυτόματο στοίβας ή σωρού* (Δημητρίου, 2002) είναι μία 7-άδα

$(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ όπου :

Q : ένα πεπερασμένο σύνολο καταστάσεων,

Σ και Γ : τα αλφάβητα των συμβόλων εισόδου και σωρού αντίστοιχα,

$q_0 \in Q$: η αρχική κατάσταση

$Z_0 \in \Gamma$: το αρχικό σύμβολο του σωρού

δ : $Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow$ πεπερασμένα υποσύνολα του $Q \times \Gamma^*$: η συνάρτηση μετάβασης

$F \subseteq Q$: το σύνολο τελικών καταστάσεων.

Μια *στιγμιαία περιγραφή* ενός αυτόματου στοίβας M είναι μία 3-άδα (q, x, α), όπου:

q : η τρέχουσα κατάσταση του M,

$x \in \Sigma^*$: το τμήμα της συμβολοσειράς που απομένει να διαβαστεί και

$\alpha \in \Gamma^*$: τα περιεχόμενα του σωρού με το αριστερότερο σύμβολο του α να αντιστοιχεί στην κορυφή του σωρού.

Αν $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ είναι ένα αυτόματο στοίβας, θα γράφουμε :

$$(q, \alpha x, Z\beta) \vdash_M (p, x, \alpha\beta)$$

και θα λέμε ότι η περιγραφή (q, αx, Zβ) παράγει σ' ένα βήμα την περιγραφή (p, x, αβ), αν και μόνο αν το αυτόματο από την κατάσταση q, διαβάζοντας το α και έχοντας το σύμβολο Z στην κορυφή του σωρού, μεταβαίνει στην κατάσταση p και αντικαθιστά το Z με α ή αλλιώς αν η δ(q, α, Z) περιέχει την κίνηση (p, α). Αν το $\alpha = \varepsilon$ τότε η περιγραφή αυτή αντιστοιχεί σε μία ε-κίνηση. Αν η (q, x, α) παράγει την (p, y, β) σε μηδέν ή περισσότερα βήματα τότε γράφουμε :

$$(q, x, \alpha) \vdash_M^* (p, y, \beta).$$

Τέλος, για να οριστούν οι γλώσσες που αναγνωρίζονται από τα αυτόματα στοίβας χρειάζεται να είναι γνωστή όχι μόνο η κατάσταση στην οποία μεταβαίνει το αυτόματο, αλλά και τα περιεχόμενα της στοίβας. Αυτό συμβαίνει γιατί, όπως προαναφέρθηκε, η μετάβαση εξαρτάται από το τρέχον σύμβολο και από το σύμβολο στην κορυφή του σωρού.

Έστω $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ ένα αυτόματο στοίβας. Μία συμβολοσειρά $x \in \Sigma^*$ γίνεται δεκτή από το M με τελική κατάσταση, αν

$$(q_0, x, Z_0) \vdash_M^* (f, \varepsilon, \alpha)$$

για κάποιο $\alpha \in \Gamma^*$ και $f \in F$. Η x γίνεται δεκτή από το M με άδειο σωρό αν

$$(q_0, x, Z_0) \models^*_M (q, \varepsilon, \varepsilon)$$

για κάποια q ∈ Q. Αντίστοιχα, οι γλώσσες $L_f(M)$ και $L_\varepsilon(M)$ είναι τα σύνολα των συμβολοσειρών που γίνονται δεκτές από το M με τελική κατάσταση και άδειο σωρό.

Γενικά, τα αυτόματα στοίβας είναι μη ντετερμινιστικά. Ωστόσο, υπάρχουν και ντετερμινιστικά αυτόματα στοίβας, τα οποία μπορεί να αναγνωρίζουν μόνο ένα υποσύνολο των γραμματικών ανεξάρτητων συμφραζομένων, αλλά παρέχουν πολλά πλεονεκτήματα στην ανάπτυξη μεταγλωττιστών.

Ένα αυτόματο στοίβας $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ είναι ντετερμινιστικό αν ισχύουν τα παρακάτω :

1. Για κάθε q ∈ Q, α ∈ Σ ∪ {ε} και Z ∈ Γ, η δ(q, α, Z) περιέχει το πολύ μία επόμενη κίνηση.
2. Για κάθε q ∈ Q και Z ∈ Γ, αν δ(q, α, Z) ≠ ∅ , τότε για οποιοδήποτε α ∈ Σ, δ(q, α, Z) =Ø.

Για τις γνωστές γλώσσες προγραμματισμού μπορούν να κατασκευαστούν ντετερμινιστικά αυτόματα στοίβας που να δέχονται συντακτικώς σωστά προγράμματα αυτών. Τα προγράμματα αυτά ονομάζονται συντακτικοί αναλυτές (parsers).

ΚΕΦΑΛΑΙΟ 4

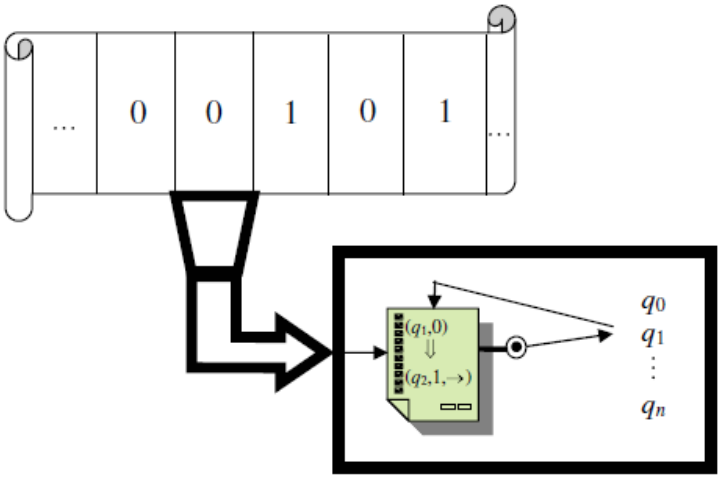
ΜΕΤΡΗΣΗ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ - ΜΗΧΑΝΕΣ TURING

Στο τμήμα της θεωρίας του υπολογισμού υπάρχουν διάφορα μοντέλα υπολογιστικών συσκευών. Ένα είδος είναι τα πεπερασμένα αυτόματα, τα οποία αναπαριστούν ικανοποιητικά συσκευές με μικρή μνήμη. Ένα άλλο είδος είναι τα αυτόματα στοίβας, που αποτελούν μια ικανοποιητική αναπαράσταση συσκευών με απεριόριστη μνήμη, η οποία είναι όμως προσπελάσιμη μόνο ως στοίβα: το στοιχείο που μπορεί να αναληφθεί ανά πάσα στιγμή είναι αυτό που ανατέθηκε τελευταίο. Τα συγκεκριμένα μοντέλα, που παρουσιάστηκαν και στα προηγούμενα κεφάλαια, όμως δεν μπορούν να χρησιμεύσουν ως μοντέλα γενικής χρήσης.

Στο κεφάλαιο αυτό θα παρουσιάσω ένα ισχυρό υπολογιστικό μοντέλο, που προτάθηκε το 1936 από τον Alan Turing, και λέγεται μηχανή Turing (για συντομία ΤΜ). Η συγκεκριμένη μηχανή μοιάζει πολύ με πεπερασμένο αυτόματο, αλλά διαθέτει άπειρη μνήμη την όποια μπορεί να προσπελάσει χωρίς περιορισμούς. Αποτελεί επομένως ένα πολύ πιο πιστό μοντέλο υπολογιστή γενικής χρήσης. Για την ακρίβεια, μια μηχανή Turing μπορεί να κάνει ότι και ένας πραγματικός υπολογιστής. Παρ' όλα αυτά υπάρχουν προβλήματα που ούτε αυτή η μηχανή μπορεί να επιλύσει. Υπό μια πολύ ρεαλιστική έννοια, τα προβλήματα αυτά βρίσκονται πέρα από τα όρια των υπολογιστικών δυνατοτήτων.

Η Μηχανή Turing χρησιμοποιεί ως απεριόριστη μνήμη μια άπειρη ταινία. Διαθέτει μια κεφαλή η οποία έχει την δυνατότητα, μετακινούμενη επάνω στην ταινία, να διαβάζει και να γράφει σύμβολα. Αρχικά, η ταινία περιέχει μόνο τη λέξη εισόδου, και σύμβολα διαστήματος σε όλες τις υπόλοιπες θέσεις. Για να αποθηκεύσει κάποιες πληροφορίες, η μηχανή γράφει απλώς αυτές τις πληροφορίες σε κάποιο τμήμα της ταινίας. Για να διαβάσει κάποιες εγγεγραμμένες πληροφορίες, μετακινεί την κεφαλή στο αντίστοιχο τμήμα της ταινίας. Η μηχανή συνεχίζει τον υπολογισμό μέχρι ότου αποφασίσει να παραγάγει μια έξοδο. Οι μοναδικές πιθανές έξοδοι είναι αποδοχή και απόρριψη, τις οποίες η μηχανή παράγει μεταβαίνοντας σε αντίστοιχες προκαθορισμένες καταστάσεις αποδοχής και απόρριψης. Εάν δεν μεταβεί ποτέ σε κάπου από αυτές τις καταστάσεις, συνεχίζει την λειτουργία της επ' άπειρον, χωρίς να τερματίζει ποτέ. Στο σχήμα 4.1

απεικονίζεται μια μηχανή Turing n καταστάσεων, που ετοιμάζεται να μεταβεί στην κατάσταση q_2 γράφοντας στη θέση του 0 το 1 και μεταφέροντας την κεφαλή μια θέση δεξιά.



ΣΧΗΜΑ 4.1 Μια μηχανή Turing n καταστάσεων

4.1 ΤΥΠΙΚΟΣ ΟΡΙΣΜΟΣ ΤΗΣ ΜΗΧΑΝΗΣ TURING

Μηχανή Turing είναι μια επτάδα, $(Q, \Sigma, \Gamma, \delta, q_0, q_{αποδοχής}, q_{απόρριψης})$ με τα Q, Σ και Γ να είναι όλα πεπερασμένα σύνολα.

1. Q είναι το σύνολο των καταστάσεων.
2. Σ είναι το αλφάβητο των εισαγομένων (input), το οποίο δεν περιέχει τον ειδικό "κενό" χαρακτήρα #.
3. Γ είναι το αλφάβητο της ταινίας (ή αλφάβητο εργασίας) όπου $\# \in \Gamma$ και $\Sigma \subseteq \Gamma$
4. δ είναι η συνάρτηση μετάβασης. $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$.
5. q_0 ε Q είναι η αρχική κατάσταση της ΜΤ.
6. $q_{αποδοχής}$ είναι η κατάσταση αποδοχής ($q_{αποδοχής} \in Q$) και
7. $q_{απόρριψης}$ είναι η κατάσταση απόρριψης($q_{απόρριψης} \in Q$), ισχύει $q_{αποδοχή} \neq q_{απόρριψης}$

Συνοπτικά να αναφέρουμε ότι:

Η κεφαλή της ΜΤ βρίσκεται αρχικά στο "πρώτο" (αριστερότερο) σύμβολο της ταινίας ενώ η μονάδα ελέγχου βρίσκεται στην αρχική κατάσταση q_0. Ανάλογα με το τι σύμβολο διαβάζει η κεφαλή κάθε φορά και ανάλογα με την κατάσταση που βρίσκεται η μονάδα ελέγχου, η κεφαλή γράφει στην ταινία (συγκεκριμένα στην τρέχουσα θέση της) ένα κατάλληλο νέο σύμβολο (χωρίς να αποκλείεται να γράψει το ήδη υπάρχον), η μονάδα ελέγχου περνά σε μια νέα κατάσταση (χωρίς να

αποκλείεται να μείνει στην ίδια) ενώ η κεφαλή μετακινείται πάνω στην ταινία κατά μια θέση αριστερά ή δεξιά. Τις μεταβάσεις αυτές ορίζει η συνάρτηση μετάβασης δ. Για τις συμβολοσειρές εισαγομένων (input) w ισχύει w ε Σ*. Τα στοιχεία που γράφει η κεφαλή στην ταινία ανήκουν στο αλφάβητο εργασίας Γ. Οι συμβολοσειρά εισαγομένων βρίσκεται στις ‘’πρώτες’’ (αριστερότερες) θέσεις της ταινίας και την ακολουθούν σύμβολα κενού (#). Το πρώτο κενό που συναντά η κεφαλή μετακινούμενη προς τα δεξιά, υποδεικνύει και το τέλος του input. Όταν η κεφαλή βρίσκεται στο αριστερότερο στοιχείο και η συνάρτηση μετάβασης δίνει εντολή για μετακίνηση της ταινίας προς τα αριστερά η κεφαλή θα μείνει στη θέση της. Οι καταστάσεις $q_{αποδοχής}$ και $q_{απόρριψης}$ ονομάζονται και τερματικές καταστάσεις (halting states). Ο υπολογισμός της ΜΤ συνεχίζεται έως ότου η Μηχανή φτάσει σε τερματική κατάσταση (αποδοχής ή απόρριψης).

Όπως είπαμε και προηγουμένως καθώς μια ΜΤ κάνει τον υπολογισμό της, συμβαίνουν αλλαγές: Ένας συνδυασμός των τριών παραπάνω στοιχείων καλείται διαμόρφωση. Ένας συνήθης τρόπος αναπαράστασης μιας διαμόρφωσης όπου η τρέχουσα κατάσταση είναι η q και η κεφαλή διαβάζει το πρώτο σύμβολο μιας συμβολοσειράς υ είναι η uqυ. Προφανώς εδώ τα περιεχόμενα της ταινίας είναι η συνένωση (concatenation) των συμβολοσειρών u και υ (οι δυο συμβολοσειρές διαδοχικά). Να σημειώσουμε ότι u, υ ∈ Γ*.

Για παράδειγμα αν u = 110001 και υ = 10001 στη διαμόρφωση 110001q210001, η τρέχουσα κατάσταση είναι η q_2 το περιεχόμενο της ταινίας είναι το 11000110001 και η κεφαλή διαβάζει τον τέταρτο άσσο μετρώντας από αριστερά. Θα λέμε ότι μια διαμόρφωση C_1 παράγει μια άλλη διαμόρφωση C_2 όταν μπορεί να μεταβεί νόμιμα από τη μια διαμόρφωση στην άλλη. Το "νόμιμα" σημαίνει ότι η μετάβαση αυτή ορίζεται από τη συνάρτηση μετάβασης δ. Συγκεκριμένα για a,b, c ∈ Γ και u,υ ∈ Γ* και qi, qj ∈ Q η διαμόρφωση uaqibu παράγει τη uacqju αν ισχύει: δ(qi, b) = (qj, c, R)

Όπως φαντάζεστε η μετάβαση μπορεί να προβλέπει και μετακίνηση της κεφαλής προς τα αριστερά. Έτσι η διαμόρφωση uaqibu παράγει τη uqjacu εφόσον ισχύει: δ(qi, b) = (qj,c,L) Αναγνωρίζουμε τις εξής περιπτώσεις ειδικών διαμορφώσεων:

Η αρχική διαμόρφωση για είσοδο w είναι η q0w: η κεφαλή της ΜΤ διαβάζει το πρώτο σύμβολο του w.

Τόσο η διαμόρφωση αποδοχής όσο και αυτή της απόρριψης ονομάζονται διαμορφώσεις τερματισμού (halting configurations) καθώς είναι οι διαμορφώσεις στις οποίες τερματίζει τη λειτουργία της η ΜΤ. Αυτές οι διαμορφώσεις δεν παράγουν άλλες.

Θα λέμε ότι μια μηχανή Turing M δέχεται μια εισερχόμενη συμβολοσειρά w εάν υπάρχει μια ακολουθία διαμορφώσεων Ci,... ,Ck τέτοια ώστε:

- η διαμόρφωση Ci είναι η αρχική διαμόρφωση της M στο w.
- κάθε Ci παράγει την Ci+1, και
- Η Ck είναι αποδεκτή.

Η συλλογή των λέξεων που αποδέχεται η M είναι η γλώσσα της M, ή η γλώσσα που αναγνωρίζεται από την M, και συμβολίζεται με L(M).

Μια γλώσσα λέγεται αναγνωρίσιμη εάν υπάρχει μηχανή Turing που να την αναγνωρίζει. Όταν θέτουμε σε λειτουργία μια μηχανή Turing επί κάποιας εισόδου, υπάρχουν τρία δυνατά αποτελέσματα: η μηχανή μπορεί να αποδεχτεί, να απορρίψει η' να εγκλωβιστεί. Με τον όρο εγκλωβισμός, εννοούμε ότι η μηχανή απλώς δεν τερματίζει ποτέ, αλλά πλανάται επ' άπειρον σε αποκλειστικά μη τερματικές καταστάσεις. Ο εγκλωβισμός μπορεί να περιλαμβάνει οποιαδήποτε απλή ή περίπλοκη συμπεριφορά της μηχανής η οποία δεν καταλήγει σε κάποια τερματική κατάσταση.

Μια μηχανή Turing μπορεί να μην αποδεχτεί την είσοδο της είτε μεταβαίνοντας στην κατάσταση $q_{\text{απόρριψης}}$ και απορρίπτοντας, είτε εγκλωβιζόμενη. Μερικές φορές είναι δύσκολο να διακρίνουμε εάν μια μηχανή έχει εγκλωβιστεί ή εάν απλώς καθυστερεί υπερβολικά. Για τον λόγο αυτό, προτιμούμε μηχανές που τερματίζουν για όλες τις εισόδους, δηλαδή που δεν εγκλωβίζονται ποτέ.

Προκειμένου να προσδιορίσουμε πόσο χρόνο απαιτεί μια μηχανή Turing για να διαγνώσει την A, θα πρέπει να αναλύσουμε τον αλγόριθμο της. Το πλήθος των βημάτων που εκτελεί κάποιος αλγόριθμος για μια συγκεκριμένη είσοδο μπορεί να εξαρτάται από πολλές παραμέτρους. Εάν π.χ. η είσοδος είναι κάποιο γράφημα, το πλήθος των βημάτων μπορεί να εξαρτάται από το πλήθος των κόμβων, το πλήθος των ακμών, ή τον μέγιστο βαθμό του γραφήματος, ή ακόμη και από κάποιο συνδυασμό αυτών των παραγόντων και/ή άλλων. Ωστόσο, για λόγους απλότητας, υπολογίζουμε τον χρόνο εκτέλεσης ενός αλγορίθμου ως συνάρτηση μόνο του

μήκους της λέξης που αναπαριστά την είσοδο, αδιαφορώντας για οποιεσδήποτε άλλες παραμέτρους. Στην ανάλυση χειρότερης περίπτωσης, την οποία θα ακολουθήσουμε, μας ενδιαφέρει, για κάθε μήκος, μόνο ο μέγιστος από τους χρόνους εκτέλεσης για όλες τις εισόδους του συγκεκριμένου μήκους. Μια άλλη επιλογή, με την οποία όμως δεν θα ασχοληθούμε, είναι η ανάλυση μέσης περίπτωσης, όπου, για κάθε μήκος, μας ενδιαφέρει ο μέσος όρος των χρόνων εκτέλεσης για όλες τις εισόδους του μήκους αυτού.

Έστω M μια αιτιοκρατική μηχανή Turing που τερματίζει για κάθε είσοδο. Ο χρόνος εκτέλεσης ή η χρονική πολυπλοκότητα της M είναι η συνάρτηση f : N→N που για κάθε n επιστρέψει ως f(n) το μέγιστο πλήθος βημάτων που είναι δυνατόν να πραγματοποιήσει η M όταν το μήκος της εισόδου της είναι n. Εάν ο χρόνος εκτέλεσης της M είναι f(n), λέμε ότι η M εκτελείται σε χρόνο f(n) ή ότι είναι μια μηχανή Turing χρόνου f(n). Κατά σύμβαση, το μήκος της εισόδου αναπαρίσταται πάντοτε από τη μεταβλητή n.

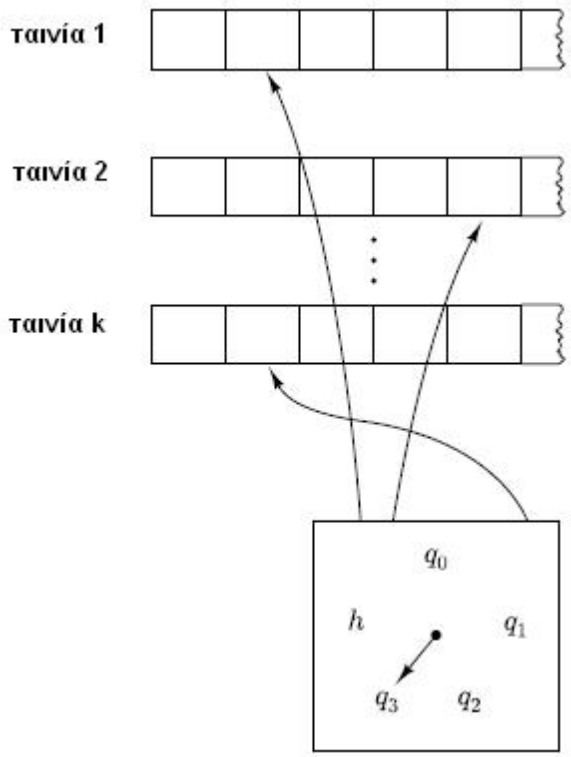
4.2 ΠΑΡΑΛΛΑΓΕΣ ΜΗΧΑΝΩΝ TURING

Υπάρχουν πολλοί εναλλακτικοί ορισμοί της μηχανής Turing στους οποίους συγκαταλέγονται και οι πολυταινιακές ή μη αιτιοκρατικές εκδοχές. Όλες αυτές οι εναλλακτικές εκδοχές ονομάζονται παραλλαγές του βασικού μοντέλου. Το αυθεντικό μοντέλο και οι εύλογες παραλλαγές του έχουν την ίδια ισχύ. Με άλλα λόγια, αναγνωρίζουν την ίδια κλάση γλωσσών. Γενικότερα, όταν κάποιες τροποποιήσεις στον ορισμό ενός υπολογιστικού μοντέλου δεν επηρεάζουν την ισχύ του, λέμε ότι το μοντέλο είναι ευσταθές(Rus, 2005).

4.2.1 ΠΟΛΥΤΑΙΝΙΑΚΕΣ ΜΗΧΑΝΕΣ TURING

Μια πολυταινιακή μηχανή Turing δεν είναι παρά μια συνήθης μηχανή Turing με περισσότερες από μια ταινίες. Κάθε ταινία έχει την δική της κεφαλή εγγραφής και ανάγνωσης. Κατά την εκκίνηση της μηχανής, η είσοδος είναι καταχωρισμένη στην ταινία με αριθμό 1, ενώ οι υπόλοιπες ταινίες είναι κενές. Η μόνη αλλαγή στον τυπικό ορισμό αφόρα τη συνάρτηση μεταβάσεων, που τώρα επιτρέπει την εγγραφή, την ανάγνωση, και τη μετακίνηση της κεφαλής ταυτόχρονα σε όλες τις ταινίες ή μόνο σε ορισμένες από αυτές. Συγκεκριμένα, η τυπική μορφή της

συνάρτησης μεταβάσεων είναι : Q x Γ k—> Q x Γk x {L, R,S}k , όπου k το πλήθος των ταινιών και το S αντιπροσωπεύει τη νέα δυνατότητα της στασιμότητας. Στο σχήμα 4.2 βλέπουμε μια πολυταινιακή μηχανή Turing k ταινιών.



ΣΧΗΜΑ 4.2 Μια πολυταινιακή μηχανή Turing k ταινιών

Αν και οι πολυταινιακές μηχανές Turing φαίνονται εκ πρώτης όψεως ισχυρότερες από τις συνήθεις μηχανές Turing, στην πραγματικότητα, τα δυο μοντέλα είναι ισοδύναμα (Καββαδίας,1999).

4.2.2 ΜΗ ΑΙΤΙΟΚΡΑΤΙΚΕΣ ΜΗΧΑΝΕΣ TURING

Μια μη αιτιοκρατική μηχανή Turing (nondeterministic Turing machine, NTM) ορίζεται κατά τον αναμενόμενο τρόπο : σε οποιοδήποτε σημείο ενός υπολογισμού, η μηχανή ενδέχεται να έχει περισσότερες από μια δυνατότητες για τη συνέχεια. Επομένως, η συνάρτηση μεταβάσεων της έχει τη μορφή :

δ :Q x Γ —> P(Q x Γ x {L, R}).

Ως συνήθως ο υπολογισμός μιας τέτοιας μηχανής σχηματίζει ένα δένδρο του οποίου οι κλάδοι αντιστοιχούν στις διαφορετικές δυνατότητες. Η μηχανή αποδέχεται την είσοδο της εάν υπάρχει κλάδος που να την οδηγεί στην κατάσταση αποδοχής. Όπως έχει αποδειχτεί η μη αιτιοκρατία δεν επηρεάζει την ισχύ του μοντέλου της μηχανής Turing . με αλλά λόγια, κάθε μη αιτιοκρατική μηχανή μπορεί να προσομοιωθεί από κάποια αιτιοκρατική (deterministic Turing machine, DTM) (Καββαδίας,1999).

4.3 ΤΟ ΔΟΓΜΑ CHURCH-TURING

Παρά τη μακρά προϊστορία των αλγορίθμων στα μαθηματικά, η έννοια του αλγορίθμου καθεαυτή ορίστηκε με ακρίβεια στον εικοστό αιώνα. Μέχρι τότε, η μαθηματικοί είχαν μόνο μια διαισθητική αντίληψη της, βάση της οποίας χρησιμοποιούσαν ή περιέγραφαν αλγόριθμους. Ωστόσο, η διαισθητική αυτή ανάλυση δεν επαρκούσε για την βαθύτερη κατανόηση των αλγορίθμων.

Το 1900 ο μαθηματικός David Hilbert παρέδωσε μια διάλεξη στο διεθνές συνέδριο μαθηματικών των Παρισίων. Στο πλαίσιο αυτής της διάλεξης, απαρίθμησε είκοσι τρία μαθηματικά προβλήματα, τα οποία και πρότεινε ως στόχο για τον επόμενο αιώνα. Το δέκατο πρόβλημα του Hilbert ήταν να επινοηθεί ένας αλγόριθμος που να ελέγχει αν κάποιο πολυώνυμο έχει ακέραια ρίζα. Για την ακρίβεια, η διατύπωση δεν αναφερόταν σε αλγόριθμο αλλά σε «μια διαδικασία σύμφωνα με την οποία να μπορεί αυτό να διαπιστωθεί με πεπερασμένο πλήθος πράξεων».

Σήμερα γνωρίζουμε ότι δεν υπάρχει αλγόριθμος για την συγκεκριμένη εργασία, με άλλα λόγια το πρόβλημα αυτό είναι αλγοριθμικά ανεπίλυτο. Αν και η διαισθητική αντίληψη τότε επαρκούσε για την σύνταξη αλγορίθμων που εκτελούσαν ορισμένες εργασίες, ήταν εντελώς ανεπαρκής για να αποδειχτεί ότι δεν υπάρχει αλγόριθμος για κάποια συγκεκριμένη εργασία. Για να αποδεχτεί ότι κάποιος συγκεκριμένος αλγόριθμος δεν υπάρχει, απαιτείτε ένας σαφής ορισμός του αλγορίθμου. Για να υπάρξει κάποια πρόοδος στο δέκατο πρόβλημα, θα έπρεπε λοιπόν πρώτα να διατυπωθεί ένας τέτοιος ορισμός.

Ο ορισμός αυτός διατυπώθηκε το 1936 σε άρθρα των Alonso church και Alan Turing.o church όρισε την έννοια του αλγορίθμου χρησιμοποιώντας ένα συμβολικό

σύστημα των λεγόμενο λογισμό λ, ενώ ο Turing μέσω των «μηχανών» του. Η δυο ορισμοί αυτοί αποδείχτηκαν ισοδύναμοι, και έτσι αυτή η συσχέτιση ανάμεσα στην άτυπη έννοια του αλγορίθμου και στον ακριβή ορισμό του ονομάστηκε τελικά δόγμα Church Turing.

Διαισθητική έννοια του αλγορίθμου	ίσον	πρόγραμμα μηχανής Turing

ΣΧΗΜΑ 4.3 Το δόγμα Church-Turing

ΚΕΦΑΛΑΙΟ 5

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΧΡΟΝΟΥ

Στόχος μας σε αυτό το κεφάλαιο είναι να παρουσιάσουμε τα βασικά στοιχειά της θεωρίας χρονικής πολυπλοκότητας. Θα δείξουμε πως ταξινομούνται τα διάφορα προβλήματα ανάλογα με το χρόνο που απαιτούν. Επίσης, θα μελετήσουμε το ενδεχόμενο κάποια επιλύσιμα προβλήματα να απαιτούν εξωπραγματικό χρόνο, και θα εξηγήσουμε πως μπορούμε να αντιληφθούμε ότι ένα δεδομένο πρόβλημα είναι αυτού του τύπου.

5.1 ΣΧΕΣΕΙΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ ΜΕΤΑΞΥ ΜΟΝΤΕΛΩΝ

Στην ενότητα αυτή, θα εξετάσουμε πώς μπορεί να επηρεάσει τη χρονική πολυπλοκότητα μιας γλώσσας η επιλογή υπολογιστικού μοντέλου. Θα εστιάσουμε την προσοχή μας σε τρία διαφορετικά μοντέλα: τη μονοταινιακή μηχανή Turing, την πολυταινιακή μηχανή Turing, και την μη αιτιοκρατική μηχανή Turing.

Έστω t(n) κάποια συνάρτηση τέτοια ώστε t(n) ≥ n. Για κάθε πολυταινιακή μηχανή Turing χρόνου t(n), υπάρχει ισοδύναμη μονοταινιακή μηχανή Turing χρόνου $O(t^2(n))$.

Η ιδέα στην οποία βασίζεται η απόδειξη αυτού του θεωρήματος είναι πολύ απλή. Υπενθυμίζουμε ότι αναφέραμε πώς μπορεί να προσομοιωθεί μια πολυταινιακή μηχανή Turing από μια μονοταινιακή. Θα αναλύσουμε, λοιπόν, εκείνη την προσομοίωση, ώστε να προσδιορίσουμε πόσο επιπλέον χρόνο αναλώνει. Όπως θα δούμε, η προσομοίωση καθενός από τα βήματα της πολυταινιακής μηχανής απαιτεί το πολύ O(t(n)) βήματα στη μονοταινιακή μηχανή. Επομένως, το συνολικό πλήθος βημάτων είναι $O(t^2(n))$.

Θα πρέπει να επισημάνουμε ότι ο χρόνος εκτέλεσης μιας μη αιτιοκρατικής μηχανής Turing, όπως ορίζεται, δεν επιδιώκεται να αντιστοιχεί σε κάποια πραγματική υπολογιστική συσκευή. Ο ορισμός αυτός αποτελεί περισσότερο ένα χρήσιμο μαθηματικό εργαλείο που, όπως θα δείξουμε σε λίγο, μας διευκολύνει να χαρακτηρίσουμε την πολυπλοκότητα μιας σημαντικής κλάσης υπολογιστικών προβλημάτων.

Έστω t(n) οποιαδήποτε συνάρτηση τέτοια ώστε t(n) > n. Ώα κάθε μη αιτιοκρατική μονοταινιακή μηχανή Turing χρόνου t(n) υπάρχει ισοδύναμη αιτιοκρατική μονοταινιακή μηχανή Turing χρόνου $2^{O(t(n))}$ (Sipser, 2006)

5.2 Η ΚΛΑΣΗ P – ΠΟΛΥΩΝΥΜΙΚΟΣ ΧΡΟΝΟΣ

Για τους σκοπούς της μελέτης μας, οποιαδήποτε πολυωνυμική διαφορά μεταξύ δύο χρόνων εκτέλεσης θεωρείται μικρή, ενώ οποιαδήποτε εκθετική διαφορά θεωρείται μεγάλη. Ας δούμε για ποιους λόγους επιλέγουμε να κάνουμε αυτή τη διάκριση ανάμεσα στις πολυωνυμικές και τις εκθετικές συναρτήσεις, και όχι ανάμεσα σε κάποιες άλλες κλάσεις συναρτήσεων.

Κατ' αρχάς, παρατηρήστε την τρομακτική διαφορά στον ρυθμό αύξησης ενός τυπικού πολυωνύμου, όπως το n^3, και μιας τυπικής εκθετικής συνάρτησης, όπως η 2^n.Έστω, π.χ., ότι το n ισούται με 1000, ένα εύλογο μήκος εισόδου για κάποιον αλγόριθμο. Στην περίπτωση αυτή, το n^3 ισούται με 1 δισεκατομμύριο, ένας μεγάλος αριθμός, τον οποίο όμως μπορούμε να διαχειριστούμε. Αντιθέτως, η τιμή 2^n είναι ένας αριθμός πολύ μεγαλύτερος ακόμη και από το πλήθος των ατόμων του σύμπαντος. Με λίγα λόγια, οι αλγόριθμοι πολυωνυμικού χρόνου είναι αρκετά ταχείς ώστε να είναι χρήσιμοι σε πολλές περιπτώσεις, ενώ οι αλγόριθμοι εκθετικού χρόνου είναι τόσοι βραδείς που σπανίως είναι χρήσιμοι.

Συνήθως, οι αλγόριθμοι εκθετικού χρόνου προκύπτουν όταν επιχειρούμε να βρούμε τη λύση ενός προβλήματος αναζητώντας την εξαντλητικά μέσα στον χώρο όλων των υποψήφιων λύσεων. Η μέθοδος αυτή λέγεται εξαντλητική αναζήτηση. Παραδείγματος χάριν, ένας τρόπος να παραγοντοποιήσουμε έναν αριθμό είναι να αναζητήσουμε τους πρώτους αριθμούς στους οποίους αναλύεται ανάμεσα σε όλους τους δυνητικούς διαιρέτες. Δεδομένου ότι το μέγεθος αυτού του χώρου αναζήτησης είναι εκθετικό ως προς το μέγεθος της εισόδου, το ίδιο ισχύει και για τον χρόνο που απαιτεί η συγκεκριμένη αναζήτηση. Σε ορισμένες περιπτώσεις, η βαθύτερη κατανόηση του προβλήματος μας βοηθάει να αποφύγουμε την εξαντλητική αναζήτηση, αποκαλύπτοντας έναν αλγόριθμο πολυωνυμικού χρόνου και επομένως μεγαλύτερης χρηστικότητας.

Όλα τα εύλογα αιτιοκρατικά υπολογιστικά μοντέλα είναι πολυωνυμικώς ισοδύναμα. Αυτό σημαίνει ότι οποιοδήποτε από αυτά μπορεί να προσομοιώσει

οποιοδήποτε άλλο προκαλώντας πολυωνυμική μόνο αύξηση του χρόνου εκτέλεσης (Sipser, 2006).

Εφεξής, θα επικεντρώσουμε την προσοχή μας σε πλευρές της θεωρίας χρονικής πολυπλοκότητας που δεν επηρεάζονται από πολυωνυμικές διαφορές στον χρόνο εκτέλεσης. Τέτοιες διαφορές θα θεωρούνται ασήμαντες, και επομένως θα αγνοούνται. Αυτό θα μας επιτρέψει να αναπτύξουμε τη θεωρία με τέτοιο τρόπο που να μην εξαρτάται από την επιλογή κάποιου συγκεκριμένου υπολογιστικού μοντέλου. Ενώ έχουμε ήδη αδιαφορήσει για τους σταθερούς συντελεστές υιοθετώντας τον ασυμπτωτικό συμβολισμό, τώρα είμαστε έτοιμοι να παραβλέψουμε ακόμη και τις πολύ μεγαλύτερες πολυωνυμικές διαφορές, όπως είναι παραδείγματος χάριν η διαφορά ανάμεσα στον χρόνο n και τον χρόνο n^3.

Στην πραγματικότητα, δεν υπάρχει καμία αντίφαση. Επιλέγουμε να αγνοήσουμε τις πολυωνυμικές διαφορές απλώς επειδή είναι ασήμαντες για τους σκοπούς της μελέτης μας, και όχι επειδή τις θεωρούμε ασήμαντες γενικότερα. Αντιθέτως, ασφαλώς και θεωρούμε σημαντική τη διαφορά μεταξύ των χρόνων n και n^3. Ωστόσο, η απάντηση σε ορισμένα εξίσου σημαντικά ερωτήματα, όπως η πολυωνυμικότητα ή μη του προβλήματος της παραγοντοποίησης, είναι ανεξάρτητη από τις πολυωνυμικές διαφορές. Και αυτό ακριβώς είναι το είδος των ερωτημάτων στα οποία θα εστιάσουμε την προσοχή μας.

Η κλάση γλωσσών P αποτελείται από τις γλώσσες που μπορούν να διαγνωστούν σε πολυωνυμικό χρόνο από κάποια αιτιοκρατική μονοταινιακή μηχανή Turing. Με άλλα λόγια,

$$P = \bigcup_k \text{TIME}(n^k)$$

Η σημασία της οφείλεται στα εξής δύο δεδομένα:

1. Η P είναι αναλλοίωτη για όλα τα υπολογιστικά μοντέλα που είναι πολυωνυμικώς ισοδύναμα με την αιτιοκρατική μονοταινιακή μηχανή Turing.
2. Η P αντιστοιχεί χονδρικά στην κλάση των προβλημάτων που είναι πρακτικώς επιλύσιμα με υπολογιστές.

Το πρώτο δεδομένο υποδεικνύει ότι η P είναι μια μαθηματικώς ευσταθής κλάση. Είναι δηλαδή ανεξάρτητη από τα ιδιαίτερα χαρακτηριστικά του συγκεκριμένου υπολογιστικού μοντέλου που χρησιμοποιούμε.

Το δεύτερο δεδομένο υποδεικνύει ότι η P παρουσιάζει ενδιαφέρον και από πρακτική άποψη. Όταν ένα πρόβλημα ανήκει στη συγκεκριμένη κλάση,

γνωρίζουμε ότι μπορεί να επιλυθεί με κάποια μέθοδο που έχει χρόνο εκτέλεσης n^k, για κάποια σταθερά k. Το κατά πόσο αυτή η μέθοδος είναι και πρακτικά λειτουργική εξαρτάται από τη σταθερά k και από την εκάστοτε εφαρμογή. Προφανώς, μια μέθοδος με χρόνο εκτέλεσης n^{100} είναι πιθανότατα πρακτικά ανεφάρμοστη. Παρ' όλα αυτά, η «αναγόρευση» του πολυωνυμικού χρόνου σε απώτατο όριο της πρακτικής επιλυσιμότητας έχει αποδειχθεί χρήσιμη. Η εύρεση κάποιου αλγορίθμου πολυωνυμικού χρόνου για ένα πρόβλημα που πριν φαινόταν να απαιτεί εκθετικό χρόνο είναι πάντοτε αποτέλεσμα μιας νέας, βαθύτερης κατανόησης του προβλήματος, και συνήθως ακολουθείται από επιπλέον μείωση της πολυπλοκότητας του, συχνά μέχρι του σημείου της πρακτικής εφαρμοσιμότητας (Sipser, 2006).

5.3 ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΡΟΒΛΗΜΑΤΩΝ ΣΤΗΝ ΚΛΑΣΗ P

Η παρουσίαση ενός αλγορίθμου πολυωνυμικού χρόνου γίνεται μέσω κάποιας αφ' υψηλού περιγραφής του, χωρίς καμία αναφορά στις ιδιαιτερότητες κάποιου συγκεκριμένου υπολογιστικού μοντέλου. Με τον τρόπο αυτό, αποφεύγονται οι ανιαρές λεπτομέρειες σχετικά με τις ταινίες και τις κινήσεις τις κεφαλής. Ωστόσο, για να μπορούμε να ελέγχουμε την πολυωνυμικότητα του αλγορίθμου, θα πρέπει η περιγραφή του να ακολουθεί ορισμένες συμβάσεις.

Η περιγραφή των αλγορίθμων γίνεται μέσω αριθμημένων σταδίων. Η έννοια του σταδίου ενός αλγορίθμου είναι ανάλογη με αυτή του βήματος μιας μηχανής Turing, αν και βέβαια η υλοποίηση ενός σταδίου κάποιου αλγορίθμου σε μια μηχανή Turing θα απαιτεί, εν γένει, πολλά βήματα της μηχανής.

Όταν αναλύουμε κάποιον αλγόριθμο για να δείξουμε ότι εκτελείται σε πολυωνυμικό χρόνο, θα πρέπει να κάνουμε δύο πράγματα. Πρώτον, θα πρέπει να προσδιορίσουμε ένα πολυωνυμικό άνω φράγμα (συνήθως σε συμβολισμό κεφαλαίου όμικρον) για το πλήθος των σταδίων από τα οποία διέρχεται ο αλγόριθμος όταν δέχεται μια είσοδο μήκους n. Δεύτερον, θα πρέπει να εξετάσουμε το κάθε μεμονωμένο στάδιο της περιγραφής του αλγορίθμου ώστε να βεβαιωθούμε ότι μπορεί να υλοποιηθεί σε πολυωνυμικό χρόνο με κάποιο εύλογο αιτιοκρατικό υπολογιστικό μοντέλο. Αφού ολοκληρωθούν και τα δύο αυτά σκέλη της ανάλυσης, μπορούμε να συμπεράνουμε ότι ο αλγόριθμος έχει πολυωνυμικό χρόνο εκτέλεσης, διότι εκτελεί ένα πολυωνυμικό πλήθος σταδίων, καθένα από τα

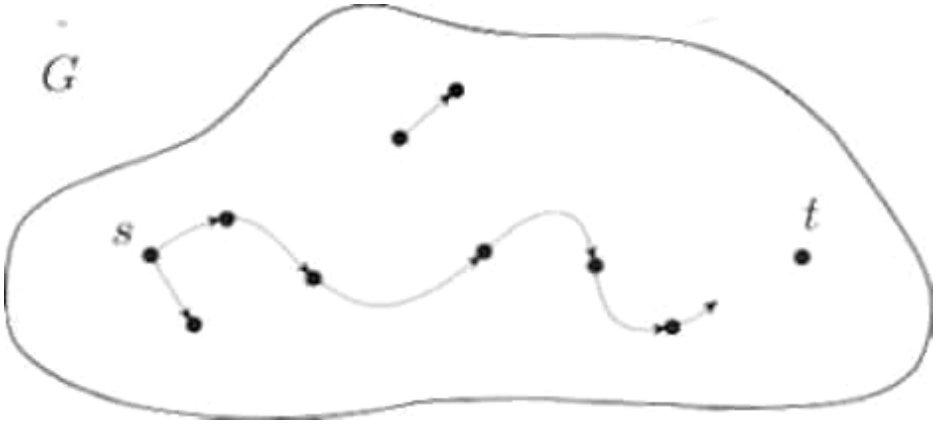
οποία μπορεί να εκτελεστεί σε πολυωνυμικό χρόνο, και διότι η σύνθεση πολυωνύμων δίνει επίσης πολυώνυμο. Θα πρέπει ακόμη να σημειωθεί ότι όταν περιγράφουμε τον αλγόριθμο, επιλέγουμε τα διάφορα στάδια της περιγραφής με τέτοιο τρόπο ώστε να διευκολύνουμε το δεύτερο σκέλος της ανάλυσης.

Ένα σημείο που απαιτεί προσοχή είναι η μέθοδος κωδικοποίησης που επιλέγουμε για τον ορισμό των προβλημάτων. Θα εξακολουθήσουμε να χρησιμοποιούμε τον συμβολισμό των γωνιωδών αγκυλών $\langle\ \rangle$, ο οποίος αναπαριστά κάποια εύλογη κωδικοποίηση ενός ή περισσότερων αντικειμένων σε μια λέξη, χωρίς να προσδιορίζουμε κάποια συγκεκριμένη μέθοδο κωδικοποίησης. Στην προκειμένη περίπτωση, εύλογη μέθοδος είναι κάποια μέθοδος που επιτρέπει πολυωνυμικού χρόνου κωδικοποίηση και αποκωδικοποίηση αντικειμένων προς και από φυσιολογικές εσωτερικές αναπαραστάσεις, ή προς άλλες εύλογες κωδικοποιήσεις. Ευτυχώς, οι συνήθεις μέθοδοι κωδικοποίησης για γραφήματα, αυτόματα, και άλλα τέτοια αντικείμενα είναι όλες εύλογες. Θα πρέπει όμως να σημειωθεί ότι η κωδικοποίηση αριθμών στο μοναδιαίο σύστημα (στο οποίο π.χ. ο αριθμός 17 αναπαρίσταται από τη μονοσυμβολική λέξη 11111111111111111) δεν αποτελεί εύλογη μέθοδο, αφού παράγει αναπαραστάσεις που είναι εκθετικά μεγαλύτερες από αυτές των πραγματικά εύλογων κωδικοποιήσεων, όπως είναι η κωδικοποίηση στο k-αδικό σύστημα για οποιοδήποτε $k \geq 2$.

Πολλά από τα υπολογιστικά προβλήματα που θα μελετήσουμε σε αυτό το κεφάλαιο περιλαμβάνουν κωδικοποιήσεις γραφημάτων. Μια εύλογη κωδικοποίηση ενός γραφήματος είναι ο κατάλογος των κόμβων και ο κατάλογος των ακμών του. Μια άλλη είναι ο πίνακας γειτνίασης, του οποίου το στοιχείο (i.j) έχει την τιμή 1 εάν υπάρχει ακμή από τον κόμβο i προς τον κόμβο j και την τιμή 0 εάν δεν υπάρχει τέτοια ακμή. Στην ανάλυση αλγορίθμων για γραφήματα, ο χρόνος εκτέλεσης ενδέχεται να υπολογίζεται συναρτήσει του πλήθους των κόμβων, και όχι του μεγέθους της αναπαράστασης του γραφήματος. Σε οποιαδήποτε εύλογη αναπαράσταση γραφήματος, όμως, το μήκος της αναπαράστασης είναι ένα πολυώνυμο του πλήθους των κόμβων. Επομένως, εάν από την ανάλυση κάποιου τέτοιου αλγορίθμου προκύψει ότι ο χρόνος εκτέλεσης του είναι πολυωνυμικός (ή εκθετικός) ως προς το πλήθος των κόμβων, τότε έπεται αυτομάτως ότι είναι επίσης πολυωνυμικός (ή εκθετικός) ως προς το μήκος της εισόδου.

5.3.1 Η ΔΙΑΔΡΟΜΗ $\in$ P.

Το πρώτο πρόβλημα που θα εξετάσουμε αφορά κατευθυνόμενα γραφήματα. Έστω ότι μας δίνεται ένα κατευθυνόμενο γράφημα G και δύο συγκεκριμένοι κόμβοι του, s και t, όπως στο παρακάτω σχήμα, και μας ζητείται να προσδιοριστεί εάν υπάρχει κατευθυνόμενη διαδρομή από τον s μέχρι τον t. Το πρόβλημα αυτό μπορεί να οριστεί μέσω της γλώσσας ΔΙΑΔΡΟΜΗ = $\{\langle G, s, t\rangle \mid$ το G είναι ένα κατευθυνόμενο γράφημα που περιλαμβάνει κατευθυνόμενη διαδρομή από τον κόμβο s μέχρι τον t$\}$.



ΣΧΗΜΑ 5.1 Μια κατευθυνόμενη διαδρομή από τον κόμβο *s* μέχρι τον *t*

Για να αποδείξουμε το θεώρημα, θα παραθέσουμε έναν αλγόριθμο ο οποίος διαγιγνώσκει τη γλώσσα ΔΙΑΔΡΟΜΗ σε πολυωνυμικό χρόνο. Προτού περάσουμε στην περιγραφή του αλγορίθμου, θα δείξουμε ότι οποιοσδήποτε εξαντλητικός αλγόριθμος για το συγκεκριμένο πρόβλημα δεν είναι αρκετά ταχύς.

Ένας εξαντλητικός αλγόριθμος εξετάζει όλες τις πιθανές διαδρομές στο G και ελέγχει για καθεμία από αυτές εάν αποτελεί κατευθυνόμενη διαδρομή από τον s μέχρι τον t. Λέγοντας «πιθανή διαδρομή» εννοούμε οποιαδήποτε ακολουθία κόμβων του G που έχει μήκος μικρότερο ή ίσο του m, όπου m το πλήθος των κόμβων. (Εάν υπάρχει κατευθυνόμενη διαδρομή από τον s μέχρι τον t , τότε υπάρχει επίσης τέτοια διαδρομή με μήκος μικρότερο ή ίσο του πλήθους m, αφού καμία επανάληψη κόμβου δεν είναι απαραίτητη.) Το πλήθος όλων αυτών των πιθανών διαδρομών, όμως, είναι της τάξης του m^m, δηλαδή αυξάνεται εκθετικά ως προς το πλήθος των κόμβων. Κατά συνέπεια, ο εξαντλητικός αυτός αλγόριθμος απαιτεί εκθετικό χρόνο(Sipser, 2006).

Για να καταστρώσουμε συνεπώς έναν αλγόριθμο πολυωνυμικού χρόνου, θα πρέπει να αποφύγουμε την εξαντλητική μέθοδο. Μια λύση είναι να χρησιμοποιήσουμε κάποια από τις συνήθεις μεθόδους διερεύνησης γραφημάτων, π.χ. την οριζόντια διερεύνηση. Στο πλαίσιο της διερεύνησης αυτής, καταγράφουμε διαδοχικά όλους τους κόμβους του G που είναι προσπελάσιμοι από τον s μέσω κατευθυνόμενων διαδρομών μήκους 1, κατόπιν μήκους 2, κατόπιν μήκους 3, έως m. Ο χρόνος εκτέλεσης αυτής της διαδικασίας μπορεί να φραγεί εύκολα με κάποιο πολυώνυμο.

ΑΠΟΔΕΙΞΗ :

Ένας αλγόριθμος πολυωνυμικού χρόνου για το πρόβλημα της διαδρομής είναι ο εξής.

Μ = “Για είσοδο $\langle G, s, t \rangle$, όπου G ένα κατευθυνόμενο γράφημα και s, t δύο κόμβοι του:

1. Καταγράφουμε τον κόμβο s.
2. Επαναλαμβάνουμε το εξής, μέχρις ότου να πάψουν να καταγράφονται επιπλέον κόμβοι:
3. Διατρέχουμε όλες τις ακμές του G. Εάν βρούμε κάποια ακμή (α, b) που να ξεκινά από κάποιον κόμβο α και να καταλήγει σε κάποιον μη καταγεγραμμένο κόμβο b, καταγράφουμε τον κόμβο b.
4. Εάν ο κόμβος t είναι καταγεγραμμένος, αποδεχόμαστε. Διαφορετικά, Απορρίπτουμε”

Θα αναλύσουμε τον παραπάνω αλγόριθμο, για να δείξουμε ότι εκτελείται σε πολυωνυμικό χρόνο. Προφανώς, καθένα από τα στάδια 1 και 4 εκτελείται μόνο μία φορά. Το στάδιο 3 εκτελείται το πολύ m φορές, αφού σε κάθε επανάληψη εκτός από την τελευταία σημαίνεται κάποιος επιπλέον κόμβος του G. Επομένως, το συνολικό πλήθος των εκτελούμενων σταδίων είναι το πολύ 1+ 1+m, και άρα πολυωνυμικό ως προς το μέγεθος του G.

Για να φράξουμε τον χρόνο που απαιτεί το κάθε μεμονωμένο στάδιο, παρατηρούμε ότι καθένα από τα στάδια 1 και 4 μπορεί να υλοποιηθεί εύκολα σε πολυωνυμικό χρόνο με οποιοδήποτε εύλογο αιτιοκρατικό μοντέλο. Το στάδιο 3 απαιτεί απλώς σάρωση της εισόδου και έλεγχο της κατάστασης ορισμένων κόμβων, και άρα μπορεί επίσης να υλοποιηθεί σε πολυωνυμικό χρόνο. Επομένως, ο Μ είναι ένας αλγόριθμος πολυωνυμικού χρόνου για το πρόβλημα της διαδρομής.

5.3.2 ΟΙ ΑΜΟΙΒΑΙΑ ΠΡΩΤΟΙ $\in$ P.

Ας περάσουμε τώρα σε ένα άλλο παράδειγμα αλγορίθμου πολυωνυμικού χρόνου. Δύο αριθμοί λέγονται **αμοιβαία πρώτοι** εάν ο μεγαλύτερος ακέραιος που διαιρεί και τους δύο ακριβώς είναι ο 1. Παραδείγματος χάριν, οι αριθμοί 10 και 21 είναι αμοιβαία πρώτοι, παρ' όλο που κανένας τους δεν είναι αφ' εαυτού πρώτος. Αντιθέτως, οι αριθμοί 10 και 22 δεν είναι αμοιβαία πρώτοι, αφού αμφότεροι διαιρούνται ακριβώς από το 2. Το πρόβλημα του ελέγχου εάν δύο αριθμοί είναι αμοιβαία πρώτοι ορίζεται τυπικά μέσω της γλώσσας

ΑΜΟΙΒΑΙΑ ΠΡΩΤΟΙ = {$\langle x, y \rangle$ | οι x και y είναι αμοιβαία πρώτοι αριθμοί}.

Ένας τρόπος επίλυσης αυτού του προβλήματος είναι να εξετάσουμε όλους τους πιθανούς κοινούς διαιρέτες των δύο αριθμών, και να αποδεχθούμε εάν δεν βρεθεί μεταξύ αυτών κανένας κοινός διαιρέτης μεγαλύτερος του 1. Λέγοντας «πιθανός κοινός διαιρέτης» εννοούμε οποιονδήποτε φυσικό αριθμό που είναι μικρότερος ή ίσος του μικρότερου από τους δύο αριθμούς της εισόδου. Ωστόσο, το μέγεθος ενός αριθμού που αναπαρίσταται στο δυαδικό σύστημα, ή σε οποιοδήποτε άλλο k-αδικό σύστημα για k ≥ 2, είναι εκθετικό ως προς το μήκος της αναπαράστασης του. Συνεπώς, ο συγκεκριμένος εξαντλητικός αλγόριθμος θα πρέπει να εξετάσει εκθετικό πλήθος πιθανών κοινών διαιρετών, και άρα απαιτεί εκθετικό χρόνο(Sipser, 2006).

Αντί αυτού του αλγορίθμου, θα χρησιμοποιήσουμε μια αριθμητική διαδικασία για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο αριθμών, που είναι γνωστή ως **ευκλείδειος αλγόριθμος**. Ο μέγιστος κοινός διαιρέτης δύο φυσικών αριθμών χ και y είναι ο μεγαλύτερος ακέραιος που διαιρεί ακριβώς αμφότερους τους x και y, και συμβολίζεται με ΜΚΔ(x, y). Παραδείγματος χάριν, ΜΚΔ(18,24) = 6. Προφανώς, οι x και y είναι αμοιβαία πρώτοι εάν και μόνο εάν ΜΚΔ(x, y) = 1. Ο ευκλείδειος αλγόριθμος Ε, που παρατίθεται αμέσως παρακάτω, χρησιμοποιεί τη συνάρτηση mod, όπου x mod y είναι το υπόλοιπο της διαίρεσης του x δια y .

ΑΠΟΔΕΙΞΗ :

Ο ευκλείδειος αλγόριθμος Ε έχει ως εξής.

Ε = “Για είσοδο $\langle x, y \rangle$, όπου x, y δύο φυσικοί αριθμοί στο δυαδικό σύστημα:

1. Επαναλαμβάνουμε μέχρις ότου y = 0:

2. Αναθέτουμε στον χ την τιμή x mod y.

3. Εναλλάσσουμε τις τιμές των x και y.

4. Αποδίδουμε ως έξοδο το x.”

Ο αλγόριθμος R που ακολουθεί λύνει το πρόβλημα των αμοιβαία πρώτων αριθμών καλώντας τον αλγόριθμο Ε ως υποπρόγραμμα.

R = “Για είσοδο $\langle x, y \rangle$,όπου x, y δύο φυσικοί αριθμοί στο δυαδικό σύστημα:

1. Εκτελούμε τον Ε για είσοδο $\langle x, y \rangle$.

2. Εάν η έξοδος του Ε είναι 1, αποδεχόμαστε. Διαφορετικά, απορρίπτουμε.”

Προφανώς, εάν ο Ε επιστρέφει πάντοτε το σωστό αποτέλεσμα και εκτελείται σε πολυωνυμικό χρόνο, τότε το ίδιο ισχύει και για τον R. Αρκεί λοιπόν να αναλύσουμε τον Ε ως προς τον χρόνο εκτέλεσης και την ορθότητα του. Η ανάλυση της ορθότητας είναι ευρύτατα γνωστή, και επομένως θα την παραλείψουμε.

Για να αναλύσουμε τη χρονική πολυπλοκότητα του Ε, θα δείξουμε κατ' αρχάς ότι κάθε εκτέλεση του σταδίου 2 (εκτός ίσως από την πρώτη) μειώνει την τιμή του x τουλάχιστον κατά το ήμισυ. Μετά την εκτέλεση του σταδίου 2, έχουμε $x < y$, λόγω της λειτουργίας της συνάρτησης mod. Στη συνέχεια, μετά την εκτέλεση του σταδίου 3, έχουμε $x > y$, αφού οι δύο αριθμοί έχουν εναλλαγεί. Συνεπώς, στην επόμενη εκτέλεση του σταδίου 2 έχουμε και πάλι $x > y$. Διακρίνουμε δύο περιπτώσεις. Εάν $x/2 > y$, τότε $x \bmod y < y \leq x/2$ και ο x ελαττώνεται τουλάχιστον κατά το ήμισυ. Εάν $x/2 < y$, τότε $x \bmod y = x - y < x/2$ και ο x ελαττώνεται και πάλι τουλάχιστον κατά το ήμισυ.

Δεδομένου ότι οι τιμές των x και y εναλλάσσονται σε κάθε εκτέλεση του σταδίου 3, οι αρχικές τιμές των x και y ελαττώνονται αμφότερες τουλάχιστον κατά το ήμισυ ανά δύο εκτελέσεις του βρόχου. Συνεπώς, το μέγιστο πλήθος εκτελέσεων των σταδίων 2 και 3 ισούται με τον μικρότερο από τους αριθμούς $2 \log_2 x$ και $2 \log_2 y$. Δεδομένου ότι οι λογάριθμοι αυτοί είναι ανάλογοι προς τα μήκη των αναπαραστάσεων, έπεται ότι το πλήθος των εκτελούμενων σταδίων είναι O(n).

Επιπλέον, είναι προφανές ότι ο χρόνος που αναλώνεται σε κάθε στάδιο του Ε είναι πολυωνυμικός. Άρα ο συνολικός χρόνος εκτέλεσης είναι επίσης πολυωνυμικός.

5.3.3 ΟΛΕΣ ΟΙ ΓΛΩΣΣΕΣ ΑΝΕΞΑΡΤΗΤΕΣ ΣΥΜΦΡΑΖΟΜΕΝΩΝ ΑΝΗΚΟΥΝ ΣΤΗΝ ΚΛΑΣΗ P.

Θα ολοκληρώσουμε αυτήν την υποενότητα δείχνοντας ότι οποιαδήποτε γλώσσα ανεξάρτητη συμφραζομένων είναι διαγνώσιμη σε πολυωνυμικό χρόνο.

Δεδομένου ότι υπάρχει η απόδειξη ότι όλες οι γλώσσες ανεξάρτητες συμφραζομένων είναι διαγνώσιμες και συγκεκριμένα, για κάθε τέτοια γλώσσα υπάρχει και ένας αλγόριθμος που την διαγιγνώσκει, εάν ο συγκεκριμένος αλγόριθμος εκτελείται σε πολυωνυμικό χρόνο, τότε το παραπάνω θεώρημα έπεται άμεσα ως πόρισμα. Ας δούμε λοιπόν τον αλγόριθμο εκείνο, και ας υπολογίσουμε τον χρόνο που απαιτεί.

Έστω L μια γλώσσα ανεξάρτητη συμφραζομένων η οποία παράγεται από κάποια γραμματική ανεξάρτητη συμφραζομένων G. Γνωρίζουμε ότι οποιαδήποτε παραγωγή κάποιας λέξης w αποτελείται από 2n -1 βήματα, όπου n το μήκος της w. Με βάση αυτήν την παρατήρηση, ο διαγνώστης της L εξετάζει, για είσοδο w, όλες τις δυνατές παραγωγές που αποτελούνται από 2n - 1 βήματα. Εάν κάποια από αυτές αποτελεί παραγωγή της w αποδέχεται, διαφορετικά, απορρίπτει.

Μια γρήγορη ανάλυση αυτού του αλγορίθμου δείχνει ότι ο χρόνος εκτέλεσης του είναι μη πολυωνυμικός. Στη γενική περίπτωση, το πλήθος όλων των παραγωγών με k βήματα αυξάνεται εκθετικά ως προς k, και συνεπώς ο συγκεκριμένος αλγόριθμος ενδέχεται να απαιτεί εκθετικό χρόνο.

Για να καταστρώσουμε έναν αλγόριθμο πολυωνυμικού χρόνου, θα χρησιμοποιήσουμε μια ισχυρή τεχνική που λέγεται δυναμικός προγραμματισμός. Η τεχνική αυτή βασίζεται στη χρήση πληροφοριών που έχουν προκύψει από την επίλυση υποπροβλημάτων μικρού μεγέθους για την επίλυση προβλημάτων μεγαλύτερου μεγέθους. Στο πλαίσιο της συγκεκριμένης μεθόδου, η λύση του κάθε υποπροβλήματος θα πρέπει να καταγράφεται, ώστε να αποφεύγεται η εκ νέου επίλυση των ίδιων υποπροβλημάτων. Για τον σκοπό αυτό, κατασκευάζουμε έναν πίνακα όλων των υποπροβλημάτων, στον οποίο καταχωρίζουμε μεθοδικά τη λύση κάθε υποπροβλήματος που επιλύουμε.

Στην προκειμένη περίπτωση, τα υποπροβλήματα συνίστανται στο να προσδιοριστεί, για κάθε μεταβλητή της G και κάθε υπόλεξη της w, εάν η συγκεκριμένη μεταβλητή μπορεί να παραγάγει τη συγκεκριμένη υπόλεξη. Οι

λύσεις όλων αυτών των υποπροβλημάτων αποθηκεύονται σε έναν πίνακα διαστάσεων n * n. Συγκεκριμένα, για i ≤ j, το (i, j)-οστό στοιχείο του πίνακα περιέχει τη συλλογή όλων των μεταβλητών που μπορούν να παραγάγουν την υπόλεξη w_i w_{i+1}… w_j. Τα στοιχεία του πίνακα με i > j δεν χρησιμοποιούνται.

Ο αλγόριθμος συμπληρώνει όλα τα στοιχεία του πίνακα που αντιστοιχούν σε υπολέξεις της w. Αρχικά συμπληρώνει τα στοιχεία που αντιστοιχούν στις υπολέξεις μήκους 1, κατόπιν αυτά που αντιστοιχούν στις υπολέξεις μήκους 2, κ.ο.κ. Τα στοιχεία πίνακα που αφορούν τα μεγαλύτερα μήκη υπολογίζονται με βάση τα στοιχεία που αφορούν τα μικρότερα μήκη.

Παραδείγματος χάριν, ας υποθέσουμε ότι ο αλγόριθμος έχει ήδη προσδιορίσει για όλες τις υπολέξεις μήκους έως και k ποιες μεταβλητές μπορούν να παραγάγουν την κάθε υπόλεξη. Για να προσδιορίσει στη συνέχεια εάν κάποια μεταβλητή A μπορεί να παραγάγει μια συγκεκριμένη υπόλεξη μήκους k + 1, διαιρεί την υπόλεξη αυτή σε δύο μη κενά τμήματα με όλους τους (k το πλήθος) δυνατούς τρόπους. Για κάθε τρόπο διαίρεσης, εξετάζει όλους τους κανόνες της μορφής A→ BC για να προσδιορίσει εάν η μεταβλητή B μπορεί να παραγάγει το πρώτο τμήμα και η μεταβλητή C το δεύτερο. Για να εκτελέσει αυτόν τον έλεγχο, χρησιμοποιεί τα στοιχεία του πίνακα που έχουν ήδη υπολογιστεί. Εάν καθεμία από τις B και C μπορεί πράγματι να παραγάγει το αντίστοιχο τμήμα, τότε η A μπορεί να παραγάγει την υπόλεξη, οπότε προστίθεται στην κατάλληλη θέση του πίνακα. Στην πρώτη φάση της διαδικασίας, που αφορά τις υπολέξεις μήκους 1, ο αλγόριθμος εξετάζει μόνο τους κανόνες της μορφής A → b.

ΑΠΟΔΕΙΞΗ:

Ο αλγόριθμος D που ακολουθεί υλοποιεί την παραπάνω αποδεικτική ιδέα. Έστω G μια γραμματική ανεξάρτητη συμφραζομένων σε κανονική μορφή Chomsky που παράγει την γλώσσα ανεξάρτητη συμφραζομένων L, και έστω S η αρχική μεταβλητή της. (Υπενθυμίζουμε ότι σε μια γραμματική που βρίσκεται σε κανονική μορφή Chomsky η κενή λέξη αντιμετωπίζεται με ιδιαίτερο τρόπο. Αντίστοιχα, ο αλγόριθμος αντιμετωπίζει την περίπτωση w = ε ξεχωριστά, στο στάδιο 1.) Δίπλα σε ορισμένα στάδια παρατίθενται, σε διπλές αγκύλες, διευκρινιστικά σχόλια.

D= ‘Για είσοδο w = w_i …..w_n :

1. Αν w = ε και υπάρχει ο κανόνας S → ε, αποδεχόμαστε διαφορετικά, απορρίπτουμε. \\χειριζόμαστε την περίπτωση w = ε\\

2. Για i = 1 έως n: \\εξετάζουμε κάθε υπόλεξη μήκους 1\\

3. Για κάθε μεταβλητή A:

4. Ελέγχουμε αν υπάρχει ο κανόνας A → b, όπου b = w_i.

5. Αν υπάρχει, καταχωρίζουμε την A στη θέση (i,i) του πίνακα.

6. Για I= 2 έως n: \\I, το μήκος της υπόλεξης\\

7. Για i = 1 έως n - I + 1: \\ ί, η αρχή της υπόλεξης\\

8. Θέτουμε j = i +I - 1. \\ j, το τέλος της υπόλεξης \\

9. Για k = i έως j - 1: \\k, το σημείο διαίρεσης \\

10. Για κάθε κανόνα A→ BC:

11. Αν τα στοιχεία [i, k) και (k + 1,j) περιέχουν αντίστοιχα τις μεταβλητές B και C, καταχωρίζουμε τη μεταβλητή A στη θέση (i, j).

12. Αν η μεταβλητή S εμπεριέχεται στο στοιχείο (1, n), αποδεχόμαστε. διαφορετικά, απορρίπτουμε’

Ας αναλύσουμε αυτόν τον αλγόριθμο. Κατ' αρχάς, είναι προφανές ότι κάθε στάδιο μπορεί να υλοποιηθεί εύκολα ώστε να εκτελείται σε πολυωνυμικό χρόνο. Περαιτέρω, τα στάδια 4 και 5 εκτελούνται το πολύ nυ φορές, όπου υ το πλήθος των μεταβλητών της G, που είναι μια προκαθορισμένη σταθερά, ανεξάρτητη του n. Άρα τα στάδια αυτά εκτελούνται O(n) φορές. Στη συνέχεια, το στάδιο 6 εκτελείται το πολύ n φορές. Σε κάθε εκτέλεση του, το στάδιο 7 εκτελείται επίσης το πολύ n φορές. Σε καθεμία από αυτές τις εκτελέσεις, τα στάδια 8 και 9 εκτελούνται και αυτά το πολύ n φορές. Σε κάθε επανάληψη του σταδίου 9, το στάδιο 10 εκτελείται το πολύ r φορές, όπου r το πλήθος των κανόνων της G, που είναι επίσης μια προκαθορισμένη σταθερά. Άρα το στάδιο 11, ο εσώτατος βρόχος, εκτελείται $O(n^3)$ φορές. Αθροίζοντας το σύνολο, βρίσκουμε ότι το πλήθος των σταδίων που εκτελεί ο αλγόριθμος D είναι $O(n^3)$, δηλαδή πολυωνυμικό.

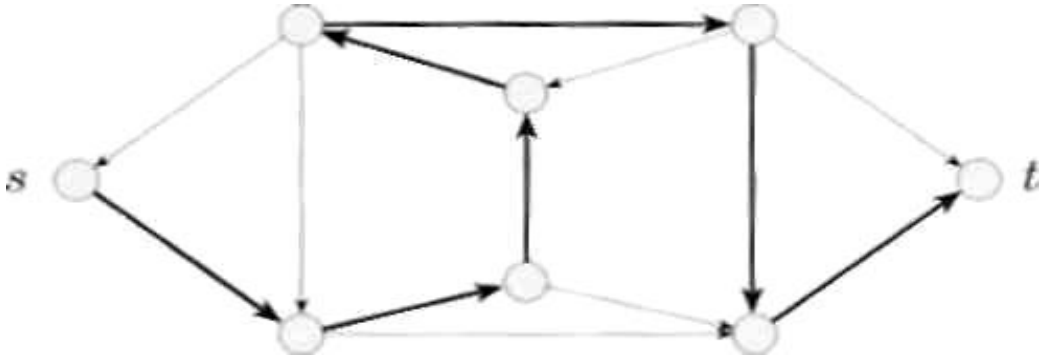
5.4 Η ΚΛΑΣΗ NP

Όπως αναφέραμε, σε πολλά προβλήματα μπορούμε να αποφύγουμε την εξαντλητική αναζήτηση και να βρούμε λύσεις πολυωνυμικού χρόνου. Ωστόσο, σε ορισμένα άλλα προβλήματα, πολλά από τα οποία είναι ενδιαφέροντα και χρήσιμα, όλες οι προσπάθειες να αποφευχθεί η εξαντλητική αναζήτηση έχουν μέχρι στιγμής αποτύχει. Απ' όσο γνωρίζουμε αυτή τη στιγμή, δεν υπάρχουν αλγόριθμοι πολυωνυμικού χρόνου που να επιλύουν αυτά τα προβλήματα.

Πού οφείλεται η αποτυχία αυτή; Η απάντηση στο σημαντικό αυτό ερώτημα παραμένει άγνωστη. Ίσως τα συγκεκριμένα προβλήματα να επιδέχονται αλγορίθμους πολυωνυμικού χρόνου, που όμως δεν έχουν ακόμη επινοηθεί διότι βασίζονται σε άγνωστες αρχές. Πιθανόν, όμως, ορισμένα από αυτά απλώς να μην είναι επιλύσιμα σε πολυωνυμικό χρόνο, δηλαδή να είναι εγγενώς δύσκολα.

Ένα αξιοσημείωτο γεγονός σχετικά με αυτό το ερώτημα είναι ότι οι πολυπλοκότητες πολλών τέτοιων προβλημάτων συνδέονται μεταξύ τους. Συγκεκριμένα, ένας αλγόριθμος πολυωνυμικού χρόνου για κάποιο από αυτά τα προβλήματα μπορεί να χρησιμοποιηθεί για την επίλυση μιας ολόκληρης κλάσης προβλημάτων. Για να κατανοήσουμε αυτό το φαινόμενο, ας ξεκινήσουμε με ένα παράδειγμα.

Χαμιλτονιανή διαδρομή σε ένα κατευθυνόμενο γράφημα G είναι οποιαδήποτε κατευθυνόμενη διαδρομή που διέρχεται από κάθε κόμβο του γραφήματος ακριβώς μία φορά. θα εξετάσουμε το πρόβλημα του προσδιορισμού εάν ένα κατευθυνόμενο γράφημα περιέχει κάποια χαμιλτονιανή διαδρομή μεταξύ δύο καθορισμένων κόμβων, όπως αυτή που απεικονίζεται στο παρακάτω σχήμα. Έστω λοιπόν η γλώσσα ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ = { $\langle G, s, t \rangle$ | το G είναι ένα κατευθυνόμενο γράφημα που περιέχει χαμιλτονιανή διαδρομή από τον s στον t}.



ΣΧΗΜΑ 5.2 Μια χαμιλτονιανή διαδρομή η οποία διέρχεται από κάθε κόμβο ακριβώς μια φορά.

Ορισμένα προβλήματα ενδέχεται να μην είναι καν πολυωνυμικώς επαληθεύσιμα. Παραδείγματος χάριν, θεωρήστε το πρόβλημα ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ, το συμπλήρωμα του προβλήματος της χαμιλτονιανής διαδρομής. Ακόμη και αν μπορούσαμε με κάποιον τρόπο να εξακριβώσουμε ότι ένα συγκεκριμένο γράφημα δεν έχει χαμιλτονιανή διαδρομή, δεν γνωρίζουμε κανέναν τρόπο με τον οποίο θα μπορούσε κάποιος να επαληθεύσει την ανυπαρξία τέτοιας διαδρομής χωρίς να χρησιμοποιήσει τον ίδιο αλγόριθμο εκθετικού χρόνου με τον οποίο εξακριβώθηκε η ανυπαρξία αυτή. Ένας σχετικός τυπικός ορισμός είναι ο ακόλουθος.

Ονομάζουμε επαληθευτή μιας γλώσσας A οποιονδήποτε αλγόριθμο V τέτοιον ώστε A = {w | ο V αποδέχεται τη λέξη $\langle w, c \rangle$, όπου c κάποια λέξη}.

Ο χρόνος εκτέλεσης του επαληθευτή υπολογίζεται μόνο συναρτήσει του μήκους της λέξης w. Επομένως, ένας επαληθευτής πολυωνυμικού χρόνου έχει πολυωνυμικό χρόνο εκτέλεσης ως προς το μήκος της w. Μια γλώσσα A ονομάζεται **πολυωνυμικά επαληθεύσιμη** εάν υπάρχει για αυτήν επαληθευτής πολυωνυμικού χρόνου.

Για να επαληθεύσει ότι μια λέξη w ανήκει στη γλώσσα A, ο επαληθευτής χρησιμοποιεί κάποια επιπλέον πληροφορία, την οποία αναπαριστά η λέξη c. Η πληροφορία αυτή λέγεται πιστοποιητικό, ή απόδειξη, της συμμετοχής της w στην A. Σημειωτέον ότι, για επαληθευτές πολυωνυμικού χρόνου, το πιστοποιητικό έχει πάντοτε πολυωνυμικό μήκος (ως προς το μήκος της w), διότι ο επαληθευτής είναι αδύνατον να προσπελάσει μεγαλύτερη ποσότητα πληροφορίας εντός του χρονικού του φράγματος. Ας δούμε τώρα πώς εφαρμόζεται ο ορισμός αυτός στις γλώσσες ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ και ΣΥΝΘΕΤΟΙ.

Στο πρόβλημα της χαμιλτονιανής διαδρομής, πιστοποιητικό για μια λέξη $\langle G, s, t \rangle \in$ ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ είναι απλώς οποιαδήποτε χαμιλτονιανή διαδρομή από τον κόμβο s μέχρι τον κόμβο t. Στο πρόβλημα της συνθετότητας αριθμών, πιστοποιητικό για έναν αριθμό $x \in$ ΣΥΝΘΕΤΟΙ είναι απλώς οποιοσδήποτε διαιρέτης του. Και στις δύο περιπτώσεις, όταν δίνεται το πιστοποιητικό ο επαληθευτής μπορεί να ελέγξει σε πολυωνυμικό χρόνο εάν η είσοδος ανήκει ή όχι στη γλώσσα.

NP είναι η κλάση όλων των γλωσσών που επιδέχονται επαληθευτή πολυωνυμικού χρόνου (Sipser, 2006)**.**

Η κλάση NP είναι πολύ σημαντική, καθότι περιλαμβάνει πολλά προβλήματα πρακτικού ενδιαφέροντος. Σύμφωνα με τα προηγούμενα, τα προβλήματα ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ και ΣΥΝΘΕΤΟΙ ανήκουν αμφότερα στην NP. Βεβαίως, όπως έχουμε ήδη αναφέρει, το πρόβλημα της συνθετότητας ανήκει επίσης στην P, η οποία είναι υποσύνολο της NP, αλλά η απόδειξη αυτού του ισχυρότερου συμπεράσματος είναι πολύ πιο δύσκολη. Ο όρος NP προέρχεται από το αγγλικό αντίστοιχο της φράσης **μη αιτιοκρατικός πολυωνυμικός χρόνος** (non-deterministic polynomial time), η οποία εκφράζει έναν εναλλακτικό ορισμό της NP μέσω μη αιτιοκρατικών μηχανών Turing πολυωνυμικού χρόνου. Ας εξετάσουμε λεπτομερέστερα αυτόν τον ορισμό.

Η παρακάτω μηχανή N_1, είναι μια μη αιτιοκρατική μηχανή Turing που διαγιγνώσκει το πρόβλημα της χαμιλτονιανής διαδρομής σε μη αιτιοκρατικό πολυωνυμικό χρόνο. Υπενθυμίζουμε ότι, ο χρόνος εκτέλεσης μιας μη αιτιοκρατικής μηχανής είναι ο χρόνος που απαιτεί ο πλέον χρονοβόρος κλάδος του υπολογισμού της.

N_1 = “Για είσοδο $\langle G, s, t \rangle$, όπου G ένα κατευθυνόμενο γράφημα και s, t δύο κόμβοι του:

1. Καταρτίζουμε έναν κατάλογο m αριθμών, $\rho_1,\ldots,\rho_m$ όπου m το πλήθος των κόμβων του G. Κάθε αριθμός στον κατάλογο αυτό επιλέγεται μη αιτιοκρατικά στο διάστημα από 1 έως m.
2. Ελέγχουμε εάν ο κατάλογος περιέχει επαναλήψεις. Εάν βρούμε έστω και μία, απορρίπτουμε.
3. Ελέγχουμε εάν $s = \rho_1$ και $t = \rho_m$. Εάν κάτι από τα δύο δεν ισχύει, απορρίπτουμε.
4. Για κάθε i από 1 έως m - 1, ελέγχουμε εάν το ζεύγος (ρ_i,ρ_{i+1}) αποτελεί ακμή του G. Εάν έστω και ένα ζεύγος δεν αποτελεί ακμή, απορρίπτουμε. Διαφορετικά, όλοι οι έλεγχοι έχουν δώσει θετικό αποτέλεσμα, οπότε αποδεχόμαστε.”

Για να αναλύσουμε αυτόν τον αλγόριθμο και να βεβαιωθούμε ότι εκτελείται σε μη αιτιοκρατικό πολυωνυμικό χρόνο, εξετάζουμε καθένα από τα στάδια του ξεχωριστά. Η μη αιτιοκρατική επιλογή του σταδίου 1 μπορεί προφανώς να εκτελεστεί σε πολυωνυμικό χρόνο. Στα στάδια 2 και 3, κάθε επιμέρους έλεγχος είναι αρκετά απλός, οπότε ο χρόνος που απαιτούν τα δύο στάδια συνολικά είναι

επίσης πολυωνυμικός. Τέλος, το ίδιο προφανώς ισχύει και για το στάδιο 4. Συμπεραίνουμε λοιπόν ότι ο αλγόριθμος μας εκτελείται πράγματι σε μη αιτιοκρατικό πολυωνυμικό χρόνο.

Μια γλώσσα ανήκει στην κλάση NP εάν και μόνο εάν υπάρχει μηχανή Turing μη αιτιοκρατικού πολυωνυμικού χρόνου που να τη διαγιγνώσκει (Sipser, 2006)**.**

Θα δείξουμε πώς μπορούμε να μετατρέψουμε έναν επαληθευτή πολυωνυμικού χρόνου σε έναν ισοδύναμο μη αιτιοκρατικό διαγνώστη πολυωνυμικού χρόνου, και αντιστρόφως. Ο διαγνώστης προσομοιώνει τον επαληθευτή μαντεύοντας το πιστοποιητικό. Ο επαληθευτής προσομοιώνει τον διαγνώστη χρησιμοποιώντας ως πιστοποιητικό τον αποδεκτικό κλάδο υπολογισμού.

ΑΠΟΔΕΙΞΗ:

Για την ορθή κατεύθυνση του θεωρήματος, υποθέτουμε ότι A € NP και θα δείξουμε ότι η A μπορεί να διαγνωστεί από κάποια μη αιτιοκρατική μηχανή Turing N πολυωνυμικού χρόνου. Έστω V ο επαληθευτής πολυωνυμικού χρόνου για την A ο οποίος προβλέπεται από τον ορισμό της NP. Εάν ο V έχει χρόνο εκτέλεσης n^k, τότε κατασκευάζουμε την N ως εξής.

N= “Για είσοδο w, μια λέξη μήκους n:

1. Επιλέγουμε μη αιτιοκρατικά μια λέξη c μήκους το πολύ n^k.

2. Εκτελούμε τον V για είσοδο $\langle w, c \rangle$.

3. Αν ο V αποδέχεται, αποδεχόμαστε διαφορετικά, απορρίπτουμε”

Για την αντίστροφη κατεύθυνση του θεωρήματος, υποθέτουμε ότι η A διαγιγνώσκεται από μια μη αιτιοκρατική μηχανή Turing N πολυωνυμικού χρόνου, και κατασκευάζουμε έναν επαληθευτή πολυωνυμικού χρόνου V, ως εξής.

V = “Για είσοδο $\langle w, c \rangle$, όπου w και c δύο λέξεις:

1. Προσομοιώνουμε την N για είσοδο w, χρησιμοποιώντας κάθε σύμβολο της c ως μια περιγραφή της μη αιτιοκρατικής επιλογής που πρέπει να κάνουμε σε κάθε βήμα .

2. Εάν ο συγκεκριμένος κλάδος του υπολογισμού της N αποδέχεται, αποδεχόμαστε διαφορετικά, απορρίπτουμε"

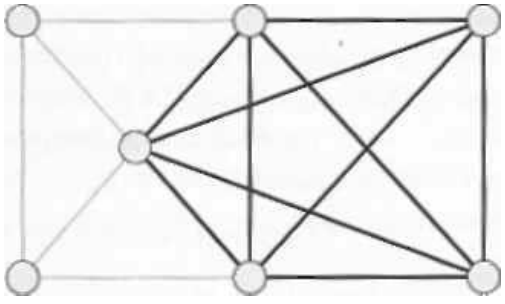
Το μη αιτιοκρατικό αντίστοιχο της κλάσης TIME(t(n)), η κλάση μη αιτιοκρατικής χρονικής πολυπλοκότητας NTIME(t(n)), ορίζεται τυπικά ως εξής.
NTIME(t(n)) = {L |η γλώσσα L διαγιγνώσκεται από κάποια μη αιτιοκρατική μηχανή Turing χρόνου O)(t(n))}.

ΠΟΡΙΣΜΑ : NP = $\bigcup_k$ NTIME(n^k).

Η κλάση NP δεν επηρεάζεται από την επιλογή του όποιου εύλογου μη αιτιοκρατικού υπολογιστικού μοντέλου, διότι όλα αυτά τα μοντέλα είναι πολυωνυμικώς ισοδύναμα. Επομένως, κατά την περιγραφή και ανάλυση ενός αλγορίθμου μη αιτιοκρατικού πολυωνυμικού χρόνου, αρκεί να ακολουθούμε τις συμβάσεις που εισαγάγαμε για τους αλγορίθμους αιτιοκρατικού πολυωνυμικού χρόνου. Με άλλα λόγια, κάθε στάδιο του αλγορίθμου θα πρέπει να έχει μια προφανή υλοποίηση πολυωνυμικού χρόνου σε κάποιο εύλογο μη αιτιοκρατικό υπολογιστικό μοντέλο. Επιπλέον, από την ανάλυση του αλγορίθμου θα πρέπει να προκύπτει ότι το πλήθος των σταδίων που εκτελούνται σε οποιονδήποτε κλάδο του υπολογισμού είναι το πολύ πολυωνυμικό (Sipser, 2006).

5.4.1 ΚΛΙΚΑ ∈ NP.

Κλίκα σε ένα μη κατευθυντικό γράφημα είναι ένα υπογράφημα στο οποίο οποιοιδήποτε δύο κόμβοι συνδέονται μέσω ακμής. Μια κλίκα που περιέχει k κόμβους ονομάζεται k-κλίκα (συντομογραφία της έκφρασης «Α:-κομβική κλίκα»). Στο Σχήμα 5.3 βλέπουμε κάποιο ενδεικτικό γράφημα που περιέχει μια 5-κλίκα.



ΣΧΗΜΑ 5.3 Ένα γράφημα που περιέχει 5-κλικα

Το πρόβλημα της κλίκας συνίσταται στο να προσδιοριστεί εάν ένα γράφημα περιέχει κάποια κλίκα δεδομένου μεγέθους. Έστω λοιπόν η γλώσσα

ΚΛΙΚΑ = {$\langle G,k \rangle$ | το G είναι ένα μη κατεύθυντικο γράφημα που περιέχει k-κλίκα}.

ΚΛΙΚΑ ∈ NP ΑΠΟΔΕΙΞΗ:

Ο αλγόριθμος N που ακολουθεί είναι μια μηχανή Turing μη αιτιοκρατικού πολυωνυμικού χρόνου, μπορείτε να αποδείξετε το θεώρημα παραθέτοντας μια τέτοια μηχανή που να διαγιγνώσκει το πρόβλημα της κλίκας.

N = 'Για είσοδο $\langle G,k \rangle$, όπου G ένα μη κατευθυντικό γράφημα και k ένας ακέραιος:

1. Επιλέγουμε μη αιτιοκρατικό ένα σύνολο k κόμβων του G, έστω c
2. Ελέγχουμε εάν το G περιέχει όλες τις ακμές οι οποίες συνδέουν κόμβους του c.
3. Εάν ναι, αποδεχόμαστε διαφορετικά, απορρίπτουμε.'

5.4.2 ΑΘΡΟΙΣΜΑ ΥΠΑΚΟΛΟΥΘΙΑΣ ∈ NP.

Στη συνέχεια θα εξετάσουμε το λεγόμενο πρόβλημα του αθροίσματος υπακολουθίας, το οποίο αφορά αριθμητική ακεραίων. Στο πρόβλημα αυτό, μας δίνεται μια ακολουθία αριθμών $x_1,\ldots,x_k$ και ένας αριθμός-στόχος t. Το ζητούμενο είναι να προσδιοριστεί εάν υπάρχει υπακολουθία της $x_1,\ldots,x_k$ της οποίας τα μέλη να έχουν άθροισμα ακριβώς t. Επομένως, μας ενδιαφέρει η γλώσσα

ΑΘΡΟΙΣΜΑ ΥΠΑΚΟΛΟΥΘΙΑΣ = {$\langle S,t \rangle$ | S = ($x_1,\ldots,x_k$) και υπάρχει υπακολουθία $y_{1,\ldots,}y_i$ της $x_1,\ldots,x_k$ με $\sum yi$ = t }

Για παράδειγμα, ((4,4,11,17,27), 25) ∈ ΑΘΡΟΙΣΜΑ ΥΠΑΚΟΛΟΥΘΙΑΣ, αφού 4 + 4 + 17 = 25. Αντιθέτως, ((4,11,17,27), 25) δεν ανήκει στο ΑΘΡΟΙΣΜΑ ΥΠΑΚΟΛΟΥΘΙΑΣ. Σημειωτέον ότι η υπακολουθία μπορεί να περιέχει έναν αριθμό το πολύ όσες φορές τον περιέχει και η ακολουθία εισόδου.

ΑΠΟΔΕΙΞΗ Ο αλγόριθμος N που ακολουθεί είναι μια μηχανή Turing μη αιτιοκρατικού πολυωνυμικού χρόνου για το πρόβλημα του αθροίσματος υπακολουθίας, η οποία έχει ως εξής.

N = “Για είσοδο $\{\langle S,t\rangle$,όπου S μια ακολουθία ακεραίων και t ένας ακέραιος:

1. Επιλέγουμε μη αιτιοκρατικό μια υπακολουθία c της S.

2. Ελέγχουμε εάν οι αριθμοί που περιλαμβάνει η c έχουν άθροισμα t.

3. Εάν ισχύει αυτό, αποδεχόμαστε διαφορετικά, απορρίπτουμε.”

Παρατηρήστε ότι τα συμπληρώματα ΚΛΙΚΑ και ΑΘΡΟΙΣΜΑ ΥΠΑΚΟΛΟΥΘΙΑΣ αυτών των γλωσσών δεν είναι προφανές ότι ανήκουν στην NP. Γενικά, το να επαληθεύσουμε ότι κάτι δεν υπάρχει φαίνεται να είναι δυσκολότερο από το να επαληθεύσουμε ότι υπάρχει. Ορίζουμε λοιπόν μια νέα κλάση πολυπλοκότητας, που την ονομάζουμε coNP και η οποία περιέχει τις γλώσσες που αποτελούν συμπληρώματα γλωσσών της NP. Μέχρι στιγμής, δεν γνωρίζουμε εάν οι κλάσεις coNP και NP είναι πράγματι διαφορετικές.

5.5 ΤΟ ΕΡΩΤΗΜΑ «P ΕΝΑΝΤΙ NP»

Όπως έχουμε πει, η NP είναι η κλάση των γλωσσών που μπορούν να επιλυθούν σε πολυωνυμικό χρόνο από κάποια μη αιτιοκρατική μηχανή Turing, ή ισοδύναμα η κλάση των γλωσσών στις οποίες η συμμετοχή μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο. Από την άλλη πλευρά, η P είναι η κλάση των γλωσσών στις οποίες η συμμετοχή μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο. Η κατάσταση αυτή συνοψίζεται ως εξής (στην συγκεκριμένη περιγραφή, ο άτυπος όρος «γρήγορα» αναφέρεται σε διαδικασίες πολυωνυμικού χρόνου).

P = η κλάση των γλωσσών στις οποίες η συμμετοχή μπορεί να διαγνωστεί γρήγορα.

NP = η κλάση των γλωσσών στις οποίες η συμμετοχή μπορεί να επαληθευτεί γρήγορα.

Έχουμε ήδη παρουσιάσει περιπτώσεις γλωσσών, όπως παραδείγματος χάριν η ΧΑΜΙΛΤΟΝΙΑΝΗ ΔΙΑΔΡΟΜΗ και η ΚΛΙΚΑ, οι οποίες ανήκουν στην NP αλλά δεν γνωρίζουμε εάν ανήκουν στην P. Γενικά, η πολυωνυμική επαληθευσιμότητα φαίνεται να είναι πολύ ισχυρότερη από την πολυωνυμική διαγωγιμότητα. Παρ' όλα αυτά, δεν έχουμε καταφέρει να αποδείξουμε ότι υπάρχει έστω και μία γλώσσα της NP που να μην ανήκει και στην P. Επομένως, όσο και αν φαίνεται αδιανόητο, οι κλάσεις P και NP μπορεί τελικά να ταυτίζονται.

Το ερώτημα αυτό, εάν P = NP, είναι ένα από τα σημαντικότερα άλυτα προβλήματα της θεωρητικής επιστήμης υπολογιστών και των σύγχρονων μαθηματικών. Εάν οι δύο κλάσεις ταυτίζονται, τότε κάθε πολυωνυμικά επαληθεύσιμο πρόβλημα είναι και πολυωνυμικά διαγνώσιμο. Οι περισσότεροι ερευνητές πιστεύουν ότι οι δύο κλάσεις διαφέρουν, διότι αν και έχει καταβληθεί τεράστια προσπάθεια για να βρεθούν αλγόριθμοι πολυωνυμικού χρόνου για ορισμένα προβλήματα της NP, τα αποτελέσματα υπήρξαν αρνητικά. Από την άλλη πλευρά, οι ερευνητές έχουν προσπαθήσει επίσης να αποδείξουν ότι οι δύο κλάσεις διαφέρουν. Αυτό, όμως, θα ισοδυναμούσε με το να αποδειχθεί ότι δεν υπάρχει ταχύς αλγόριθμος που να μπορεί να αντικαταστήσει την εξαντλητική αναζήτηση, κάτι που προς το παρόν (2007) υπερβαίνει τα όρια των αναλυτικών δυνατοτήτων (Sipser, 2006). Τα δύο ενδεχόμενα απεικονίζονται στο παρακάτω σχήμα.

ΣΧΗΜΑ 5.4 Ένα από τα δυο ενδεχόμενα Πρέπει να ισχύει

Η ταχύτερη γνωστή μέθοδος αιτιοκρατικής επίλυσης προβλημάτων της NP έχει εκθετικό χρόνο εκτέλεσης. Με άλλα λόγια, μπορεί να αποδειχθεί ότι

NP $\subseteq$ EXPTIME = $\bigcup_k$ TIME(2^{n^k}), αλλά δεν γνωρίζουμε εάν η NP εμπεριέχεται και σε κάποια μικρότερη κλάση μη αιτιοκρατικής χρονικής πολυπλοκότητας.

5.6 NP-ΠΛΗΡΟΤΗΤΑ

Μια σημαντική εξέλιξη στο ζήτημα «P έναντι NP» πραγματοποιήθηκε στις αρχές της δεκαετίας του 1970, με τις εργασίες των Stephen Cook και Leonid Levin. Οι ερευνητές αυτοί ανακάλυψαν ορισμένα προβλήματα στην κλάση NP τα οποία έχουν την εξής ιδιαιτερότητα: η πολυπλοκότητα καθενός από αυτά συνδέεται με την πολυπλοκότητα ολόκληρης της κλάσης. Συγκεκριμένα, εάν οποιοδήποτε από αυτά τα προβλήματα επιδέχεται αλγόριθμο πολυωνυμικού χρόνου, τότε το ίδιο ισχύει και για οποιοδήποτε άλλο πρόβλημα της NP. Τα προβλήματα αυτά ονομάζονται NP-πλήρη, και η ύπαρξη τους είναι εξαιρετικής σημασίας, τόσο για θεωρητικούς όσο και για πρακτικούς λόγους.

Όσον αφορά τη θεωρητική σημασία, ένας ερευνητής που προσπαθεί να δείξει ότι οι P και NP διαφέρουν μπορεί να εστιάσει την προσοχή του σε κάποιο NP-πλήρες πρόβλημα. Διότι, εάν πράγματι η NP περιέχει προβλήματα που απαιτούν υπερπολυωνυμικό χρόνο, τότε κάθε NP-πλήρες πρόβλημα είναι ένα από αυτά. Αντίστοιχα, ένας ερευνητής που προσπαθεί να αποδείξει ότι οι κλάσεις P και N P ταυτίζονται μπορεί επίσης να εστιάσει την προσοχή του σε κάποιο NP-πλήρες πρόβλημα. Διότι, για να πετύχει τον στόχο του, αρκεί να βρει έναν αλγόριθμο πολυωνυμικού χρόνου για το πρόβλημα αυτό.

Όσον αφορά την πρακτική σημασία, χάρις στην NP-πληρότητα μπορούμε να αποφύγουμε την απώλεια χρόνου που θα συνεπαγόταν η αναζήτηση αλγορίθμων πολυωνυμικού χρόνου οι οποίοι στην πραγματικότητα δεν υπάρχουν. Συγκεκριμένα, παρ' όλο που τα μαθηματικά μας ίσως δεν επαρκούν για να αποδείξουμε ότι οι κλάσεις P και NP διαφέρουν, πιστεύουμε ότι πράγματι αυτό ισχύει. Επομένως, η απόδειξη της NP-πληρότητας ενός προβλήματος αποτελεί σοβαρή ένδειξη της μη πολυωνυμικότητάς του.

Ένα NP-πλήρες πρόβλημα είναι το λεγόμενο πρόβλημα της αληθευσιμότητας. Προτού περάσουμε στον ορισμό του, θα υπενθυμίσουμε τη σχετική ορολογία. Οι μεταβλητές οι οποίες μπορούν να πάρουν μόνο τις τιμές ΑΛΗΘΕΣ και ΨΕΥΔΕΣ ονομάζονται λογικές μεταβλητές. Συχνά, η τιμή ΑΛΗΘΕΣ αναπαρίσταται με το 1 και η τιμή ΨΕΥΔΕΣ με το 0. Οι πράξεις ΚΑΙ, Ή, και ΟΧΙ, που αναπαρίστανται με τα σύμβολα $\wedge$, $\vee$, και $\neg$, αντίστοιχα, ονομάζονται λογικές πράξεις. Λογικός τύπος είναι οποιαδήποτε έκφραση που περιλαμβάνει λογικές μεταβλητές και πράξεις. Η περιγραφή τους παρατίθεται στον πίνακα που

ακολουθεί, όπου χρησιμοποιούμε ως συντομογραφία του συμβόλου $\neg$ την υπεργράμμιση (δηλαδή $\bar{x}$ σημαίνει $\neg\ x$).

0 ∧ 0 = 0	0 ∨ 0 = 0	$\bar{0}$ = 1
0 ∧ 1 = 0	0 ∨ 1 = 1	$\bar{1}$ = 0
1 ∧ 0 = 0	1 ∨ 0 = 1	
1 ∧ 1 = 1	1 ∨ 1 = 1	

Λογικός τύπος είναι οποιαδήποτε έκφραση που περιλαμβάνει λογικές μεταβλητές και πράξεις. Ένας τέτοιος τύπος είναι:

Φ =($\bar{\mathbf{x}}$ ∧ y) ∨ (x ∧ $\bar{z}$)

Ένας λογικός τύπος φ ονομάζεται αληθεύσιμος εάν υπάρχει κάποιος συνδυασμός τιμών 0 και 1 τέτοιος ώστε εάν οι μεταβλητές του λάβουν αυτές τις τιμές ο τύπος να παίρνει την τιμή 1. Λέμε ότι η τιμοδοσία αυτή καθιστά τον φ αληθή, ή ότι αποτελεί αληθοποιό τιμοδοσία για τον φ. Το πρόβλημα της αληθευσιμότητας συνίσταται στο να ελεγχθεί εάν ένας λογικός τύπος είναι αληθεύσιμος. Έστω λοιπόν,

ΑΛΗΘΕΥΣΙΜΟΤΗΤΑ ={(φ)| ο φ είναι ένας αληθεύσιμος λογικός τύπος}.

Για την παραπάνω γλώσσα, χρησιμοποιούμε επίσης την ονομασία SAT, από τον αγγλικό όρο «satisfiability», που δηλώνει την αληθευσιμότητα.

5.7 ΑΝΑΓΩΓΙΜΟΤΗΤΑ ΠΟΛΥΩΝΥΜΙΚΟΥ ΧΡΟΝΟΥ

Σε προηγούμενη παράγραφο ορίσαμε την έννοια της αναγωγής ενός προβλήματος σε κάποιο άλλο: όταν ένα πρόβλημα Α ανάγεται στο Β, τότε μια λύση για το Β μπορεί να χρησιμοποιηθεί για να επιλυθεί το Α. Στην ενότητα αυτή, θα ορίσουμε μια εκδοχή της αγωγιμότητας στην οποία συνυπολογίζεται η δραστικότητα των υπολογισμών: όταν το Α είναι δραστικά αναγώγιμο στο Β, τότε μια ταχεία λύση για το Β μπορεί να χρησιμοποιηθεί για να επιλυθεί ταχέως το Α.

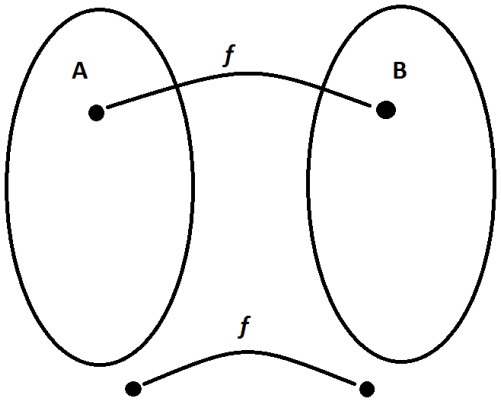
Μια συνάρτηση f : Σ*→Σ* είναι υπολογίσιμη σε πολυωνυμικό χρόνο εάν υπάρχει μηχανή Turing Μ πολυωνυμικού χρόνου η οποία, για κάθε είσοδο w,

τερματίζει έχοντας στην ταινία της μόνο τη λέξη f(w). Εναλλακτικά, λέμε απλώς ότι η f είναι μια συνάρτηση πολυωνυμικού χρόνου.

Έστω δύο γλώσσες A και B. Λέμε ότι η A είναι απεικονιστικά αναγώγιμη σε πολυωνυμικό χρόνο, ή απλώς αναγώγιμη σε πολυωνυμικό χρόνο, στη B (σε συμβολική μορφή, $A \leq_p B$), εάν υπάρχει συνάρτηση πολυωνυμικού χρόνου $f : \Sigma^* \rightarrow \Sigma^*$ τέτοια ώστε, για κάθε w, να ισχύει

$w \in A \leftrightarrow f(w) \in B$.

Στην περίπτωση αυτή, λέμε ότι η f είναι η πολυωνυμικού χρόνου αναγωγή της A στην B. Μια πολυωνυμικού χρόνου συνάρτηση f που αναγάγει τη γλώσσα A στη B, απεικονίζεται στο παρακάτω σχήμα.



ΣΧΗΜΑ 5.5 Η συνάρτηση *f ανάγει τη γλώσσα A στη B*

Όπως και μια συνήθης απεικονιστική αναγωγή, έτσι και μια αναγωγή πολυωνυμικού χρόνου της A στη B μας παρέχει έναν τρόπο να μετατρέπουμε τον έλεγχο συμμετοχής στην A σε έλεγχο συμμετοχής στη B, με τη μόνη διαφορά ότι τώρα η μετατροπή αυτή μπορεί να γίνει με δραστικό τρόπο. Για να ελέγξουμε εάν $w \in A$, μετασχηματίζουμε αρχικά τη λέξη w στη λέξη f(w) μέσω της αναγωγής f, και στη συνέχεια ελέγχουμε εάν $f(w) \in B$.

Εάν κάποια γλώσσα είναι αναγώγιμη σε πολυωνυμικό χρόνο σε κάποια άλλη για την οποία γνωρίζουμε ήδη μια λύση πολυωνυμικού χρόνου τότε η αρχική γλώσσα είναι επίσης διαγνώσιμη σε πολυωνυμικό χρόνο.

Πριν παρουσιάσουμε ένα παράδειγμα αναγωγής πολυωνυμικού χρόνου, θα περιγράψουμε το πρόβλημα της αληθευσιμότητας $_3$ΣΚΜ. Πρόκειται για μια ειδική περίπτωση του προβλήματος της αληθευσιμότητας, στην οποία όλοι οι τύποι βρίσκονται σε μια ιδιαίτερη μορφή, θα ξεκινήσουμε με κάποια απαραίτητη ορολογία. Ονομάζουμε λεξίγραμμα κάθε λογική μεταβλητή ή άρνηση λογικής

μεταβλητής, όπως π.χ. τα x και $\bar{x}$. Φράση ονομάζεται οποιοδήποτε σύνολο λεξιγραμμάτων που συνδέονται μέσω της πράξης της διάζευξης, όπως π.χ. ($x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4$) Λέμε ότι ένας λογικός τύπος βρίσκεται σε συζευκτική κανονική μορφή, οπότε ονομάζεται τύπος ΣΚΜ, εάν αποτελεί σύζευξη φράσεων, π.χ.

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6).$$

Εάν επιπλέον η κάθε φράση περιέχει τρία λεξιγράμματα, όπως στην έκφραση

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6),$$

τότε πρόκειται για έναν τύπο $_3$ΣΚΜ. Έστω λοιπόν η γλώσσα $_3$SAT = {<Ø> | ο Ø είναι ένας αληθεύσιμος τύπος $_3$ΣΚΜ}. Σημειωτέον ότι σε έναν τύπο ΣΚΜ, για να είναι μια τιμοδοσία αληθοποιός θα πρέπει σε κάθε φράση να υπάρχει τουλάχιστον ένα λεξίγραμμα που να παίρνει τιμή 1.

Το επόμενο θεώρημα περιγράφει μια πολυωνυμικού χρόνου αναγωγή του προβλήματος της αληθευσιμότητας $_3$ΣΚΜ ($_3$SAT) στο πρόβλημα της κλίκας.

Η γλώσσα $_3$SAT ανάγεται στη γλώσσα ΚΛΙΚΑ σε πολυωνυμικό χρόνο.

Η πολυωνυμικού χρόνου αναγωγή / του προβλήματος της αληθευσιμότητας $_3$ΣΚΜ στο πρόβλημα της κλίκας την οποία θα παρουσιάσουμε μετατρέπει λογικούς τύπους σε γραφήματα. Στα προκύπτοντα γραφήματα, κάθε κλίκα ενός συγκεκριμένου μεγέθους αντιστοιχεί σε μια αληθοποιό τιμοδοσία για τον τύπο. Τα γραφήματα είναι σχεδιασμένα έτσι ώστε κάποια δομικά στοιχεία τους να μιμούνται τη συμπεριφορά των μεταβλητών και των φράσεων.

ΑΠΟΔΕΙΞΗ Έστω ϕ ένας τύπος με k φράσεις, όπως ο

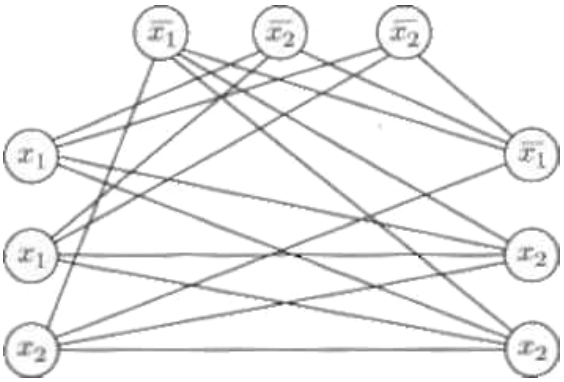
$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_k \vee b_k \vee c_k).$$

Για τον τύπο αυτό, η αναγωγή f παράγει τη λέξη $\langle G, k \rangle$, όπου G ένα μη κατευθυντικό γράφημα που ορίζεται ως εξής.

Οι κόμβοι του G είναι χωρισμένοι σε k ομάδες $t_1, \ldots, t_k$ των τριών κόμβων η καθεμία, που ονομάζονται τριάδες. Κάθε τριάδα αντιστοιχεί σε μία από τις φράσεις του Ø, και κάθε κόμβος της αντιστοιχεί σε ένα από τα λεξιγράμματα αυτής της φράσης. Ο κάθε κόμβος επιγράφεται με το σύμβολο του αντίστοιχου λεξιγράμματος.

Οι κόμβοι του G συνδέονται όλοι μεταξύ τους ανά δύο με ακμές, εκτός από τους κόμβους της ίδιας τριάδας και τους κόμβους με αντίθετες επιγραφές, όπως

π.χ. x_2 και $\bar{x}_2$. Στο παρακάτω σχήμα (5.6) απεικονίζεται η κατασκευή αυτή για τον τύπο $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$



ΣΧΗΜΑ 5.6 Το γράφημα το οποίο παράγει αναγωγή για τον τύπο ϕ =
$(x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$

Για να αποδείξουμε την ορθότητα της παραπάνω κατασκευής, θα πρέπει να δείξουμε ότι ο τύπος ϕ είναι αληθεύσιμος εάν και μόνο εάν το γράφημα G εμπεριέχει κάποια k-κλίκα. Θα εξετάσουμε την κάθε κατεύθυνση της απόδειξης ξεχωριστά.

Έστω ότι υπάρχει αληθοποιός τιμοδοσία για τον ϕ. Στην τιμοδοσία αυτή, κάθε φράση περιέχει τουλάχιστον ένα αληθές λεξίγραμμα. Επιλέγουμε λοιπόν από κάθε τριάδα του G έναν κόμβο που να αντιστοιχεί σε αληθές λεξίγραμμα της αληθοποιού τιμοδοσίας. (Εάν κάποια φράση περιέχει περισσότερα από ένα αληθή λεξιγράμματα, τότε από την αντίστοιχη τριάδα επιλέγουμε αυθαίρετα οποιονδήποτε από τους αντίστοιχους κόμβους.) Οι επιλεγμένοι κατ' αυτόν τον τρόπο κόμβοι σχηματίζουν k-κλίκα. Το συμπέρασμα αυτό προκύπτει ως εξής. Κατ' αρχάς, το πλήθος των επιλεγμένων κόμβων είναι k, διότι επιλέξαμε ακριβώς έναν κόμβο για καθεμία από τις k τριάδες. Επιπλέον, κάθε ζεύγος επιλεγμένων κόμβων συνδέεται μέσω ακμής, διότι κανένα τέτοιο ζεύγος δεν εμπίπτει στις εξαιρέσεις που περιγράψαμε πιο πάνω. Συγκεκριμένα, οποιοιδήποτε δύο επιλεγμένοι κόμβοι αποκλείεται να ανήκουν στην ίδια τριάδα, αφού κάθε τριάδα συνεισφέρει μόνο έναν κόμβο επίσης, αποκλείεται να έχουν αντίθετες επιγραφές, διότι τα αντίστοιχα λεξιγράμματα είναι αμφότερα αληθή στην επιλεγμένη αληθοποιό τιμοδοσία. Συνεπώς, το G περιέχει μια k-κλίκα.

Αντιστρόφως, έστω ότι το G περιέχει μια k-κλίκα. Στην περίπτωση αυτή, καμία τριάδα δεν μπορεί να περιέχει δύο ή περισσότερους κόμβους αυτής της κλίκας, διότι οι κόμβοι της ίδιας τριάδας δεν συνδέονται με ακμές. Επομένως, καθεμία από τις k τριάδες του G περιέχει ακριβώς έναν από τους k κόμβους της κλίκας. Αποδίδουμε στις μεταβλητές του ϕ τέτοιες τιμές ώστε τα λεξιγράμματα που εμφανίζονται στις επιγραφές των κόμβων της κλίκας να είναι όλα αληθή. Τέτοια τιμοδοσία υπάρχει σίγουρα, διότι η κλίκα αποκλείεται να περιέχει δύο κόμβους με αντίθετες επιγραφές, αφού τέτοιοι κόμβοι δεν θα συνδέονταν μεταξύ τους μέσω ακμής. Η συγκεκριμένη τιμοδοσία είναι αληθοποιός για τον ο, αφού κάθε τριάδα περιέχει κάποιον κόμβο της κλίκας, και επομένως κάθε φράση περιέχει κάποιο λεξίγραμμα που παίρνει την τιμή ΑΛΗΘΗΣ Κατά συνέπεια, ο τύπος ϕ είναι αληθεύσιμος.

Από τα Θεωρήματα έπεται ότι εάν το πρόβλημα της κλίκας επιλύεται σε πολυωνυμικό χρόνο τότε το ίδιο ισχύει και για το πρόβλημα της αληθευσιμότητας $_3$ΣΚΜ. Η συσχέτιση αυτή φαίνεται μάλλον αναπάντεχη, αφού εκ πρώτης όψεως πρόκειται για δύο εντελώς διαφορετικά προβλήματα. Ωστόσο, η αναγωγιμότητα πολυωνυμικού χρόνου μάς επιτρέπει να συσχετίσουμε τις πολυπλοκότητες τους. Με παρόμοιο τρόπο μπορούμε να συσχετίσουμε τις πολυπλοκότητες μιας ολόκληρης κλάσης προβλημάτων. Για τον σκοπό αυτό, χρειαζόμαστε τον ακόλουθο ορισμό.

Μια γλώσσα B είναι NP-πλήρης εάν ικανοποιεί τις εξής δύο συνθήκες:

1. Η β ανήκει στην κλάση NP.

2. Κάθε γλώσσα A $\in$ NP ανάγεται στην B σε πολυωνυμικό χρόνο.

Εάν η B είναι N P-πλήρης και B $\in$ P, τότε P = NP.

ΑΠΟΔΕΙΞΗ Το θεώρημα έπεται άμεσα από τον ορισμό της αναγωγιμότητας πολυωνυμικού χρόνου.

Εάν η B είναι NP-πλήρης, η C ανήκει στην κλάση NP, και επιπλέον B $\leq_P$ C, τότε η C είναι επίσης NP-πλήρης.

ΑΠΟΔΕΙΞΗ Γνωρίζουμε ήδη ότι η C $\in$ NP, οπότε αρκεί να δείξουμε ότι κάθε γλώσσα A $\in$ NP ανάγεται στην C σε πολυωνυμικό χρόνο. Δεδομένου ότι η B είναι NP-πλήρης, κάθε γλώσσα της NP ανάγεται σε πολυωνυμικό χρόνο στην B, η οποία με τη σειρά της ανάγεται σε πολυωνυμικό χρόνο στην C. Η σύνθεση

αναγωγών πολυωνυμικού χρόνου δίνει επίσης αναγωγή πολυωνυμικού χρόνου. Με άλλα λόγια, εάν η A ανάγεται σε πολυωνυμικό χρόνο στην B και η B ανάγεται σε πολυωνυμικό χρόνο στην C, τότε και η A ανάγεται σε πολυωνυμικό χρόνο στην C. Συνεπώς, κάθε γλώσσα της NP ανάγεται όντως σε πολυωνυμικό χρόνο στην C.

5.8 NP-ΠΛΗΡΗ ΠΡΟΒΛΗΜΑΤΑ

Το φαινόμενο της NP-πληρότητας εμφανίζεται αρκετά συχνά. Διάφορα NP-πλήρη προβλήματα απαντούν σε πολλούς διαφορετικούς τομείς μελέτης. Στην ενότητα αυτή, θα αποδείξουμε την NP-πληρότητα διαφόρων γλωσσών. Αυτό θα μας δώσει τη δυνατότητα να μελετήσουμε κάποιες τεχνικές που χρησιμοποιούνται σε αποδείξεις αυτού του είδους. Η γενική στρατηγική που ακολουθούμε είναι να καταστρώνουμε μια πολυωνυμικού χρόνου αναγωγή της γλώσσας $_3$SAT στην εκάστοτε υπό μελέτη γλώσσα. Ωστόσο, σε κάποιες περιπτώσεις είναι ευκολότερο να ανάγουμε άλλες NP-πλήρεις γλώσσες.

Όταν κατασκευάζουμε μια πολυωνυμικού χρόνου αναγωγή της γλώσσας $_3$SAT σε κάποια άλλη, αναζητούμε στην άλλη γλώσσα δομές που μπορούν να προσομοιώσουν τις μεταβλητές και τις φράσεις των λογικών τύπων. Οι δομές αυτές αποκαλούνται ενίοτε εξαρτήματα. Παραδείγματος χάριν, στην αναγωγή της $_3$SAT στο πρόβλημα της κλίκας την οποία παρουσιάσαμε παραπάνω, οι μεμονωμένοι κόμβοι προσομοιώνουν λεξιγράμματα και οι τριάδες κόμβων προσομοιώνουν φράσεις. Το αληθές ή ψευδές ενός λεξιγράμματος υπό κάποια αληθοποιό τιμοδοσία αντιστοιχεί στη συμμετοχή ή τη μη συμμετοχή ενός μεμονωμένου κόμβου στην κλίκα. Παρομοίως, το γεγονός ότι κάθε φράση θα πρέπει να περιέχει ένα αληθές λεξϊγραμμα αντιστοιχεί στο ότι κάθε τριάδα θα πρέπει να περιέχει κάποιον κόμβο που ανήκει στην κλίκα, έτσι ώστε αυτή να έχει το επιθυμητό μέγεθος. Από το Θεώρημα (παράγραφος 5.7) **Η γλώσσα $_3$SAT ανάγεται στη γλώσσα ΚΛΙΚΑ σε πολυωνυμικό χρόνο** , έπεται το ο πόρισμα ότι γλώσσα ΚΛΙΚΑ είναι NP-πλήρης.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία αυτή αρχικά παρουσιάσαμε κάποια βασικά στοιχεία τα οποία μας βοήθησαν αργότερα στον υπολογισμό της πολυπλοκότητας του χρόνου επίλυσης ενός προβλήματος όπως και στην κατανόηση της ανάγκης να την υπολογίσουμε. Έννοιες όπως η ασυμπτωτική ανάλυση ήταν ένα από τους απαραίτητους παράγοντες που μας βοήθησαν στην κατανόηση του τρόπου μέτρησης του χρόνου που απαιτεί ο αλγόριθμος όταν εκτελείται για εισόδους μεγάλου μήκους.

Για τον σκοπό του υπολογισμού της πολυπλοκότητας εξετάσαμε κάποια μοντέλα υπολογιστικών συσκευών όπως τα πεπερασμένα αυτόματα τα οποία βέβαια αναπαριστούν ικανοποιητικά συσκευές με ελάχιστη μνήμη, στη συνέχεια είδαμε τα αυτόματα στοίβας, που αποτελούν μια ικανοποιητική αναπαράσταση συσκευών με απεριόριστη μνήμη, η οποία όμως είναι προσπελάσιμη μόνο ως στοίβα. Επίσης εξετάσαμε ένα πολύ πιο ισχυρό μοντέλο, την μηχανή Turing η οποία μοιάζει αρκετά με το πεπερασμένο αυτόματο, αλλά διαθέτει άπειρη μνήμη την οποία μπορεί να προσπελάσει χωρίς περιορισμούς.

Τα μοντέλα αυτά χρησιμοποιούνται σε εφαρμογές της Επιστήμης των Υπολογιστών όπως στη λεξική ανάλυση ενός μεταγλωττιστή που βασίζεται στη προσομοίωση ενός κατάλληλου πεπερασμένου αυτομάτου και στην συντακτική ανάλυση ενός μεταγλωττιστή που βασίζεται στη προσομοίωση ενός αυτόματου στοίβας.

Έπειτα δείξαμε πως ταξινομούνται τα διάφορα προβλήματα ανάλογα με τον χρόνο που απαιτούν. Πιο αναλυτικά, είδαμε την κλάση P που αποτελείται από γλώσσες που μπορούν να διαγνωστούν σε πολυωνυμικό χρόνο από κάποια αιτιοκρατική μονοταινιακή μηχανή Turing, στην οποία αντιστοιχούν προβλήματα που είναι πρακτικώς επιλύσιμα με υπολογιστές. Προβλήματα δηλαδή όπως η εύρεση διαδρομής σε κατευθυνόμενα γραφήματα, ο ευκλείδειος αλγόριθμος και οι γλώσσες ανεξάρτητες συμφραζομένων. Επίσης αναλύσαμε κάποια προβλήματα για τα οποία δεν μπορούν σε πρώτη φάση να επιλυθούν σε πολυωνυμικό χρόνο, αυτά τα κατατάξαμε αρχικά στην κλάση NP, την κλάση δηλαδή που ανήκουν οι γλώσσες που μπορούν να διαγνωστούν από μη αιτιοκρατικές μηχανές Turing

πολυωνυμικού χρόνου. Στην κλάση αυτή είδαμε το πρόβλημα της κλίκας και του αθροίσματος υπακολουθίας.

Στη συνέχεια συναντήσαμε ορισμένα προβλήματα στην κλάση NP των οποίων η πολυπλοκότητα του καθενός από αυτά συνδέεται με την πολυπλοκότητα ολόκληρης της κλάσης δηλαδή προβλήματα στην NP που επιδέχονται αλγόριθμο πολυωνυμικού χρόνου, όποτε το ίδιο ισχύει και για οποιοδήποτε άλλο πρόβλημα στην κλάση αυτή. Αυτά τα προβλήματα είναι τα NP-πλήρη και μας απασχόλησαν πολύ γιατί στην αναζήτηση μιας απάντησης στο ερώτημα αν P = NP, τα προβλήματα αυτά θα μπορούσαν να δώσουν την λύση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Δημητρίου Αναστάσιος (2002). Αυτόματα και τυπικές γλώσσες

Καββαδίας Δημήτριος Ι. (1999), Σημειώσεις μαθήματος Υπολογιστική Πολυπλοκότητα.

Κατωπόδης Κωνσταντίνος (2000). Σημειώσεις μαθήματος Διακριτά Μαθηματικά.

Ρεφανίδης Γιάννης (2002). Διαφάνειες μαθήματος Θεωρία Υπολογισμών και Αυτομάτων.

Ροντογιάννης Π. (2009) Διαφάνειες μαθήματος Θεωρία Υπολογισμού.

Σταμάτης Δημοσθένης (2003). Σημειώσεις μαθήματος Δομές Δεδομένων με Java

Χειμωνίδης Θεοδόσιος (2001). Σημειώσεις μαθήματος Γλώσσες και Μεταγλωττιστές

Cormen Thomas H., Leiserson E.Charles, and Rivest L. Ronald (1991). Introduction to Algorithms, MIT Press.

Cormen H Thomas., Leiserson E.Charles, Rivest L.Ronald and Clifford Stein (2001). Introduction to Algorithms, 2nd edition, MIT Press.

Papadimitriou Christos H. (1994). Computational Complexity, Addison-Wesley Publishing Company.

Rus Teodor (2005). Variants of Turing Machines

Sipser Michael (2006). Εισαγωγή στη Θεωρία Υπολογισμού, Πανεπιστημιακές Εκδόσεις Κρήτης.