Τυπικές Γλώσσες και Υπολογιστικά Μοντέλα με Περιορισμένους Πόρους

Κωνσταντίνα Γαρούφη

Διπλωματική εργασία
Επιβλέπων: Κωνσταντίνος Κούτρας

$\mu\Pi\lambda\forall$

Λογικη και Θεωρια Αλγοριθμων και Υπολογισμου
Διαπανεπιστημιακό Πρόγραμμα Μεταπτυχιακών Σπουδών

Αθήνα, Σεπτέμβριος 2005

Τυπικές Γλώσσες και Υπολογιστικά Μοντέλα με Περιορισμένους Πόρους

Πρόλογος

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι διττό. Αφενός γίνεται μία επισκόπηση της θεωρίας τυπικών γλωσσών, παρατηρούμενη μέσα από το πρίσμα της διάσημης ιεραρχίας γραμματικών Chomsky, και με σχετική έμφαση στις πρακτικές εφαρμογές της. Αναλύεται η σχέση των τυπικών γραμματικών με μία σειρά υπολογιστικών μοντέλων, που κι αυτά με τη σειρά τους κτίζουν μία ιεραρχία, και παρουσιάζονται σταδιακά. Στη μελέτη της πολυπλοκότητας κάποιων από τα παραπάνω υπολογιστικά μοντέλα, όταν περιορίσουμε ορισμένους από τους πόρους τους—έναν πολλά υποσχόμενο κλάδο, από τον οποίο εκπροσωπείται κατά σημαντικό βαθμό η σύγχρονη έρευνα στο χώρο—αφιερώνεται το δεύτερο μέρος της εργασίας.

Ξεκινούμε εισάγοντας τόσο το κίνητρο για τη μελέτη του χώρου, όσο και μερικές από τις κεντρικές ιδέες που τον διαποτίζουν, στο Κεφάλαιο 1. Στο Κεφάλαιο 2 μελετώνται οι κανονικές γλώσσες, καθώς και η σύνδεση των κανονικών γραμματικών με τις κανονικές εκφράσεις και τα πεπερασμένα αυτόματα. Το Κεφάλαιο 3 παρουσιάζει τις γλώσσες χωρίς συμφραζόμενα, αλλά και τη σχέση τους με τα αυτόματα στοίβας. Σκιαγραφούνται επίσης κάποιες θεμελιώδεις αρχές, στις οποίες βασίζεται η λειτουργία ορισμένων ιδιαίτερα αποδοτικών συντακτικών αναλυτών που συμπεριφέρονται ως αυτόματα στοίβας. Στη συνέχεια, στο Κεφάλαιο 4 στρεφόμαστε στις άλλες δύο από τις τέσσερις κλάσεις γλωσσών που σχηματίζει η ιεραρχία Chomsky: τις γλώσσες χωρίς περιορισμούς και τα αντίστοιχα υπολογιστικά μοντέλα, τις μηχανές Turing, και τις γλώσσες με συμφραζόμενα και τα αντίστοιχα υπολογιστικά μοντέλα, τα γραμμικά φραγμένα αυτόματα. Τέλος, το Κεφάλαιο 5 συνίσταται στο κομμάτι της εργασίας που έχει να κάνει με τον κλάδο της πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους.

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου, κ. Κ. Κούτρα, που, παρά την πίεση χρόνου και τις αυξημένες υποχρεώσεις του, φορτώθηκε όχι μόνο το βάρος της επίβλεψης της εργασίας μου, αλλά και μια ειλικρινή, φιλική ανησυχία για πολύ περισσότερα πράγματα, την οποία εκτιμώ ιδιαίτερα. Ακόμη ευχαριστώ τον κ. Κ. Δημητρακόπουλο για όλη του τη στήριξη, καθώς και τον κ. Σ. Ζάχο για τις σημειώσεις από το μάθημά του «Αυτόματα και Τυπικές Γραμματικές». Θα ήθελα επίσης να ευχαριστήσω όλους τους καθηγητές μου στο ΜΠΛΑ.

Ιδιαίτερα θερμά επιθυμώ να ευχαριστήσω τον κ. Γ. Μοσχοβάκη, στον οποίο οφείλω την ιδέα για τη συγκεκριμένη εργασία, αλλά και πολλά ακόμη. Του είμαι βαθιά ευγνώμων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η θεωρία τυπικών γλωσσών έχει τις ρίζες της σε ένα πλήθος από διαφορετικούς χώρους. Πριν ακόμα από τη δημιουργία του ηλεκτρονικού υπολογιστή, μαθηματικοί ασχολούμενοι με τα θεμέλια της λογικής ενδιαφέρθηκαν για τον αλγοριθμικό υπολογισμό και σχεδίασαν αφηρημένες μηχανές για τη μελέτη των δυνατοτήτων και περιορισμών της υπολογιστικής διαδικασίας. Ήδη το 1936 ο A. Turing δημιούργησε μια μηχανή με όλες τις ικανότητες ενός πραγματικού σημερινού υπολογιστή.

Λίγο αργότερα, κατά τις δεκαετίες 1940 και 1950, απλούστερα είδη μηχανών, που σήμερα είναι γνωστά ως πεπερασμένα αυτόματα, μελετήθηκαν από διάφορους ερευνητές. Τα αυτόματα αυτά γεννήθηκαν για να εξυπηρετήσουν τους βιολόγους στη μελέτη συστημάτων νευρικών κυττάρων (neuron nets) και στη μοντελοποίηση της λειτουργίας του εγκεφάλου, αλλά και τους ηλεκτρολόγους μηχανικούς στην ανάπτυξη κυκλωμάτων διακοπτών (switching circuits) ως εργαλείο για το σχεδιασμό μηχανικού εξοπλισμού (hardware).

Παράλληλα, στα τέλη της δεκαετίας του 1950 ο γλωσσολόγος N. Chomsky ξεκίνησε τη μελέτη των τυπικών γραμματικών, με στόχο τη δημιουργία μηχανισμών για την περιγραφή των φυσικών γλωσσών. Οι τυπικές αυτές γραμματικές φάνηκε πως έχουν πολύ στενές σχέσεις με αφηρημένες έννοιες της επιστήμης υπολογιστών, όπως οι αλγόριθμοι και τα αυτόματα, και σήμερα υπηρετούν ως βάση για κάποια θεμελιακά τμήματα λογισμικού, όπως τμήματα μεταγλωττιστών (compilers) και επεξεργαστών κειμένου (text processors).

Στις δεκαετίες που ακολούθησαν, η μελέτη των τυπικών γλωσσών, και ιδιαίτερα της όψης τους που σχετίζεται με τα υπολογιστικά μοντέλα, άνθησε και έδωσε τέλος στη μάχη για τη διευθέτηση πολλών πρακτικών προβλημάτων, ανάμεσα στα οποία και το κρίσιμο ζήτημα της αυτοματοποίησης της γραφής των μεταγλωττιστών. Σε αντίθεση λοιπόν με το πλήθος των εφαρμογών τους,

η άμεση έρευνα πάνω στις τυπικές γλώσσες στις μέρες μας είναι μικρή.

Το επίκεντρο της έρευνας σήμερα έχει μετατοπιστεί σε επιμέρους ζητήματα, που φέρουν ξεχωριστό ενδιαφέρον, αλλά και αισιοδοξία για την επίλυση διαφόρων νέων προβλημάτων. Ένας τέτοιος κλάδος, που υπόσχεται να αλλάξει πολλά στο χώρο της αξιοπιστίας λογισμικού (software reliability), είναι αυτός της πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους.

Προτού όμως μιλήσουμε περισσότερο για όλα τα παραπάνω, θα παρουσιάσουμε στο κεφάλαιο αυτό συνοπτικά ορισμένες από τις βασικές έννοιες που κυριαρχούν στη θεωρία των τυπικών γλωσσών.

1.1 Αλφάβητα και συμβολοσειρές

Θεμελιακή είναι η έννοια του **συμβόλου** (symbol), που είναι μια αφηρημένη οντότητα όπως γράμμα ή λέξη κάποιας φυσικής γλώσσας, και δε θα ορίσουμε πιο συγκεκριμένα. Ένα μη κενό, πεπερασμένο σύνολο συμβόλων ονομάζεται **αλφάβητο** (alphabet). Τέτοια είναι το *λατινικό*, ή το *δυαδικό* αλφάβητο {0,1}.

Μία **συμβολοσειρά** (string) ή **λέξη** (word) ενός αλφαβήτου είναι μία πεπερασμένη ακολουθία συμβόλων του αλφαβήτου. Ως **μήκος** (length) της συμβολοσειράς w ορίζουμε το μήκος της ως ακολουθία, δηλαδή το πλήθος των συμβόλων της, και το συμβολίζουμε με $|w|$. Με ε θα συμβολίζουμε την **κενή συμβολοσειρά** (empty string), μήκους 0.

Δύο συμβολοσειρές x και y ενός αλφαβήτου μπορούν να συνδυαστούν για να σχηματίσουν μια τρίτη xy, με την πράξη της **παράθεσης** (concatenation), που είναι προσεταιριστική, αλλά όχι γενικά αντιμεταθετική. Η κενή συμβολοσειρά ε λειτουργεί ως *ουδέτερο* στοιχείο για αυτή την πράξη, καθώς $\varepsilon w = w\varepsilon = w$ για κάθε συμβολοσειρά w.

Η συμβολοσειρά v είναι **υποσυμβολοσειρά** (substring) της συμβολοσειράς w αν και μόνο αν υπάρχουν συμβολοσειρές x και y, τέτοιες ώστε $w = xvy$. Αν $x = \varepsilon$ ή $y = \varepsilon$, τότε η v είναι **πρόθεμα** (prefix) ή **κατάληξη** (suffix) της w, αντίστοιχα. Για κάθε φυσικό i, η συμβολοσειρά w^i ορίζεται αναδρομικά ως

$$\begin{cases} w^0 = \varepsilon \\ w^{i+1} = w^i w, \end{cases}$$

ενώ η **αντίστροφη** (reversal, mirror image) της $w = a_1 a_2 \dots a_n$ είναι η $w^R = a_n \dots a_2 a_1$.

Ακόμα, αν Σ είναι ένα αλφάβητο, συμβολίζουμε με Σ^k το σύνολο των συμβολοσειρών μήκους k του Σ. Σημειώνουμε ότι $\Sigma^0 = \{\varepsilon\}$, ανεξαρτήτως

του Σ. Έτσι, το σύνολο όλων των συμβολοσειρών του Σ

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i$$

περιέχει πάντα την κενή συμβολοσειρά. Συμβολίζοντας με Σ^+ το σύνολο των μη κενών συμβολοσειρών του Σ

$$\Sigma^+ = \bigcup_{i > 0} \Sigma^i,$$

έχουμε

$$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}.$$

1.2 Γλώσσες

Η έννοια της γλώσσας εμπερικλείει μια ποικιλία από φαινομενικά διαφορετικές κατηγορίες: φυσικές γλώσσες, γλώσσες προγραμματισμού, μαθηματικές γλώσσες. Ένας γενικός ορισμός της έννοιας οφείλει λοιπόν να συμπεριλάβει όλους αυτούς τους τύπους γλωσσών, χωρίς εγγενείς περιορισμούς ως προς το είδος των στοιχείων που θα αποτελούν μία γλώσσα.

Έτσι, **γλώσσα** (language) θα ονομάζεται ένα οποιοδήποτε υποσύνολο του Σ^*, όπου Σ ένα αλφάβητο. Επομένως Σ^*, $\varnothing$ και Σ είναι γλώσσες. Καθώς το Σ^* είναι αριθμήσιμα άπειρο, το πλήθος όλων των γλωσσών ενός δεδομένου αλφαβήτου είναι υπεραριθμήσιμο. Βέβαια μία γλώσσα μπορεί να είναι *πεπερασμένη ή άπειρη* (φυσικά οι πιο ενδιαφέρουσες είναι οι άπειρες), όμως κάθε γλώσσα είναι αναγκασμένη να αντλεί τις συμβολοσειρές της από ένα σταθερό, πεπερασμένο αλφάβητο και αυτό είναι *ο μόνος σημαντικός περιορισμός στο τι μπορεί να είναι μια γλώσσα.*

Με βάση λοιπόν αυτόν τον συνολοθεωρητικό ορισμό, μπορούμε να ορίσουμε άπειρο πλήθος από γλώσσες, σύμφωνα με το σχήμα

$$L = \{w \in \Sigma^* \mid \text{η } w \text{ έχει την ιδιότητα } P\}.$$

Για παράδειγμα, γλώσσες είναι οι $\{0, 01, 011, 0111, \ldots\}$, $\{w \in \{0,1\}^* \mid \text{η } w$ έχει ίσο πλήθος από 0 και 1$\}$, $\{w \in \{a,b,c\}^* \mid w = w^R\}$.

Εφόσον οι γλώσσες δεν είναι παρά σύνολα, ορίζονται σε αυτές άμεσα οι συνήθεις συνολοθεωρητικές πράξεις: *ένωση, τομή, διαφορά, συμπλήρωμα.* Ακόμα ορίζονται με τον προφανή τρόπο για τις γλώσσες L_1 και L_2 η **παράθεση**

$$L_1 L_2 = \{w \in \Sigma^* | \ w = xy, \text{ για κάποια } \ x \in L_1 \text{ και } y \in L_2\},$$

και για τη γλώσσα L και το φυσικό αριθμό i,

$$\begin{cases} L^0 = \{\varepsilon\} \\ L^{i+1} = L^i L. \end{cases}$$

Η **κλειστότητα** (closure) της παραπάνω πράξης

$$L^* = \bigcup_{i \geq 0} L^i$$

ονομάζεται συχνά **Kleene star**[1] και είναι το σύνολο όλων των συμβολοσειρών που προκύπτουν από την παράθεση τυχόντος πλήθους συμβολοσειρών της L. Η $\varnothing$ και η $\{\varepsilon\}$ είναι έτσι οι μόνες γλώσσες, των οποίων η κλειστότητα *δεν* είναι άπειρη. Τέλος, ορίζουμε τη γλώσσα

$$L^+ \;=\; \bigcup_{i>0} L^i \;=\; LL^*$$

καθώς και την **αντίστροφη** της L,

$$L^R = \{w \mid w^R \in L\}.$$

1.3 Γραμματικές

Ένα τυπικό σύστημα που κατέχει εξέχουσα θέση στη θεωρία μας, καθώς υπηρετεί την παραγωγή των λέξεων μιας γλώσσας, είναι η τυπική γραμματική. Μία **τυπική γραμματική** (generative grammar) G είναι μία *διατεταγμένη τετράδα* $G = (V, T, P, S)$, όπου

- V είναι ένα *αλφάβητο*. Τα σύμβολα του V καλούνται **μη τερματικά σύμβολα** (nonterminal symbols) ή **μεταβλητές** (variables).
- T είναι ένα *αλφάβητο*, τέτοιο ώστε $V \cap T = \emptyset$, και τα σύμβολα του οποίου καλούνται **τερματικά σύμβολα** (terminal symbols).
- P είναι ένα *πεπερασμένο σύνολο από διατεταγμένα ζεύγη* (α, β), όπου $\alpha, \beta \in (V \cup T)^*$ και το α περιέχει τουλάχιστον ένα σύμβολο από το V. Τα στοιχεία του T τα ονομάζουμε **κανόνες** (rules, productions) και κατά σύμβαση γράφουμε $\alpha \to \beta$ αντί για (α, β).

[1] Ο όρος αναφέρεται στον S. C. Kleene, που εισήγαγε την έννοια.

- S είναι ένα *στοιχείο του* V και το καλούμε **αρχικό σύμβολο** (start symbol, initial symbol).

Οι κανόνες μιας γραμματικής χρησιμοποιούνται για να παράγουμε νέες λέξεις από δοσμένες, αντικαθιστώντας ένα κομμάτι της λέξης όμοιο με το αριστερό μέλος ενός κανόνα, με το δεξί του μέλος.

Αυστηρά, δεδομένης μίας γραμματικής $G = (V, T, P, S)$ και των συμβολοσειρών $u, v \in \Sigma^*$, λέμε ότι **η** v **παράγεται** (is derivable) **από τη** u **σε ένα βήμα**, και γράφουμε

$$u \Rightarrow_G v,$$

αν και μόνο αν υπάρχουν συμβολοσειρές $x, y \in (V \cup T)^*$ και κανόνας $A \rightarrow v' \in P$, έτσι ώστε $u = xAy$ και $v = xv'y$. Κάθε ακολουθία της μορφής

$$u_0 \Rightarrow_G u_1 \Rightarrow_G \ldots \Rightarrow_G u_n$$

θα ονομάζεται **παραγωγή** (derivation) **σε** n **βήματα** της u_n από τη u_0. Η σχέση $\Rightarrow_G^*$ είναι *η ανακλαστική μεταβατική κλειστότητα της* $\Rightarrow_G$. Δηλαδή λέμε ότι **η** v **παράγεται από τη** u, και γράφουμε

$$u \Rightarrow_G^* v,$$

αν και μόνο αν $u = v$ ή υπάρχει συμβολοσειρά u', έτσι ώστε $u \Rightarrow_G^* u'$ και $u' \Rightarrow_G v$. Σε κάποιες περιπτώσεις μελετάμε τη *μεταβατική κλειστότητα της* $\Rightarrow_G$, που συμβολίζεται με $\Rightarrow_G^+$, και δηλώνει παραγωγή τουλάχιστον ενός βήματος.

Προκειμένου να περιορίσουμε τις επιλογές που έχουμε κατά την παραγωγή μιας λέξης, είναι συχνά χρήσιμο να απαιτήσουμε σε κάθε βήμα να αντικαθιστούμε την *αριστερότερη* μεταβλητή της τρέχουσας συμβολοσειράς με το δεξιό μέλος του αντίστοιχου κανόνα. Μία τέτοια παραγωγή λέγεται **αριστερότερη παραγωγή** (leftmost derivation) και συμβολίζεται με $\Rightarrow_{lm}$ και $\Rightarrow_{lm}^*$ για ένα ή πολλά βήματα, αντίστοιχα. Ομοίως, σε κάθε βήμα μιας **δεξιότερης παραγωγής** (rightmost derivation) αντικαθιστούμε τη *δεξιότερη* μεταβλητή και στην περίπτωση αυτή χρησιμοποιούμε τους συμβολισμούς $\Rightarrow_{rm}$ και $\Rightarrow_{rm}^*$.

Ιδιαίτερης σημασίας είναι οι συμβολοσειρές που παράγονται από το αρχικό σύμβολο S. Αυτές καλούνται **προτασιακές μορφές** (sentential forms). Η **γλώσσα που παράγεται** από την G (language generated by G) ορίζεται ως

$$\begin{aligned} L(G) &= \{w \in T^* \mid S \Rightarrow_G^* w\} \\ &= \{w \mid S \Rightarrow_G^* w\} \bigcap T^*, \end{aligned}$$

δηλαδή περιέχει ακριβώς εκείνες τις λέξεις που είναι προτασιακές μορφές *και* αποτελούνται αποκλειστικά από τερματικά σύμβολα. Τέλος, καλούμε

τις γραμματικές G_1 και G_2 **ισοδύναμες** (equivalent) εάν παράγουν την ίδια γλώσσα, δηλαδή ισχύει $L(G_1) = L(G_2)$.

Παράδειγμα 1.3.1. Μια απλή τυπική γραμματική είναι η $G = (\{S\}, \{a, b\}, P, S)$, όπου το P αποτελείται από τους κανόνες $S \to aSb$ και $S \to \varepsilon$.[2] Παρατηρούμε πως μια δυνατή παραγωγή είναι η

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb.$$

Δεν είναι δύσκολο να δούμε ότι $L(G) = \{a^n b^n \mid n \geq 0\}$. (Η συμπερίληψη $L(G) \subseteq \{a^n b^n \mid n \geq 0\}$ είναι προφανής και η αντίστροφη δείχνεται με απλή επαγωγή στο n.)

1.4 Συντακτικά δέντρα και αμφισημία

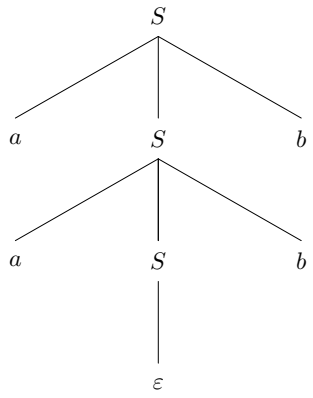
Υπάρχει ένας τρόπος αναπαράστασης των παραγωγών μίας γραμματικής που έχει αποδειχθεί ιδιαίτερα χρήσιμος, καθώς βρίσκει εφαρμογή στη λειτουργία των μεταγλωττιστών, στη συντακτική ανάλυση (parsing, syntax analysis) λέξεων, αλλά και στη διερεύνηση αμφισημίας (ambiguity) γραμματικών.

Έστω η γραμματική $G = (V, T, P, S)$. Τα **συντακτικά δέντρα** (parse trees) για την G είναι τα δέντρα που έχουν τις ακόλουθες ιδιότητες:

1. Κάθε εσωτερικός κόμβος έχει ως επιγραφή ένα στοιχείο του V.
2. Κάθε φύλλο έχει ως επιγραφή ένα στοιχείο του $V \cup T \cup \{\varepsilon\}$. Ωστόσο, αν έχει ως επιγραφή το ε, τότε είναι το μοναδικό παιδί του γονέα του.
3. Αν ένας εσωτερικός κόμβος έχει επιγραφή A, και τα παιδιά του έχουν επιγραφές
$$X_1, X_2, \ldots, X_k$$
αντίστοιχα, από τα αριστερά προς τα δεξιά, τότε ο
$$A \to X_1 X_2 \ldots X_k$$
είναι κανόνας του P.

Αν κοιτάξουμε από τα αριστερά προς τα δεξιά τις επιγραφές των φύλλων ενός συντακτικού δέντρου και τις παραθέσουμε, παίρνουμε μία συμβολοσειρά που καλείται **παραγόμενη συμβολοσειρά του συντακτικού δέντρου** (yield of

[2] Για συντομία γράφουμε $S \to aSb \mid \varepsilon$. Γενικότερα, οι κανόνες $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_n$ θα συμβολίζονται με $A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$.



Σχήμα 1.1: Το συντακτικό δέντρο που απεικονίζει την παραγωγή $S \Rightarrow^* aabb$ του Παραδείγματος 1.3.1.

the parse tree) και εύκολα βλέπουμε ότι παράγεται από τη μεταβλητή που επιγράφει τη ρίζα του. Αλλά και από μία δεδομένη παραγωγή μιας συμβολοσειράς μπορούμε μεθοδικά να κατασκευάσουμε το αντίστοιχο συντακτικό δέντρο. Έτσι φαίνεται ότι η σχέση συντακτικών δέντρων και παραγωγών συμβολοσειρών είναι άμεση.

Βέβαια είναι δυνατόν σε ένα συντακτικό δέντρο να αντιστοιχούν περισσότερες από μία παραγωγές, των οποίων όμως οι διαφορές οφείλονται μόνο στη σειρά εφαρμογής των κανόνων. Συνεπώς τα συντακτικά δέντρα στην ουσία αναπαριστούν *κλάσεις ισοδυναμίας παραγωγών*, και συγκεκριμένα τις κλάσεις παραγωγών που, καθώς οι διαφορές τους είναι επιφανειακές, θεωρούνται όμοιες. Εντούτοις, δεν είναι δύσκολο να δει κανείς ότι κάθε συντακτικό δέντρο έχει *ακριβώς μία* αριστερότερη και *ακριβώς μία* δεξιότερη παραγωγή, και αντίστροφα.

Πέραν όμως των επιφανειακών διαφορών στις παραγωγές, μία συμβολοσειρά μπορεί να παράγεται από μία γραμματική με δύο παραγωγές που διαφέρουν κατά τρόπο ουσιαστικό και δε θεωρούνται όμοιες, δηλαδή με δύο διαφορετικά συντακτικά δέντρα ή, ισοδύναμα, με δύο διαφορετικές αριστερότερες παραγωγές (και δύο διαφορετικές δεξιότερες). Έτσι, μία γραμματική G ονομάζεται **διφορούμενη** (ambiguous) αν παράγει μία λέξη $w \in L(G)$, στην οποία μπορούν να αντιστοιχηθούν δύο διαφορετικά συντακτικά δέντρα. Αντίθετα, στην περίπτωση που σε κάθε λέξη $w \in L(G)$ αντιστοιχεί ένα μόνο συντακτικό δέντρο, η G ονομάζεται **μη διφορούμενη** (unambiguous). Μία γλώσσα λέγεται **εγγενώς διφορούμενη** (inherently ambiguous) εάν όλες οι γραμματικές που την παράγουν είναι διφορούμενες.

Παράδειγμα 1.4.1. Ας θεωρήσουμε τη γραμματική $G = (\{S, A, B, C, D\}, \{a, b, c, d\}, P, S)$, όπου το P περιέχει τους κανόνες:

$$S \to AB \,|\, C$$
$$A \to aAb \,|\, ab$$
$$B \to cBd \,|\, cd$$
$$C \to aCd \,|\, aDd$$
$$D \to bDc \,|\, bc.$$

Εύκολα βλέπουμε ότι η G είναι διφορούμενη, καθώς υπάρχει λέξη της $L(G)$, στην οποία αντιστοιχούν δύο διαφορετικές αριστερότερες παραγωγές (και, συνεπώς, δύο διαφορετικά συντακτικά δέντρα):

$$S \Rightarrow_{lm} AB \Rightarrow_{lm} aAbB \Rightarrow_{lm} aabbB \Rightarrow_{lm} aabbcBd \Rightarrow_{lm} aabbccdd$$
$$S \Rightarrow_{lm} C \Rightarrow_{lm} aCd \Rightarrow_{lm} aaDdd \Rightarrow_{lm} aabDcdd \Rightarrow_{lm} aabbccdd$$

Αποδεικνύεται μάλιστα ότι η γλώσσα $L(G) = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$ είναι εγγενώς διφορούμενη.

Φυσικά οι διφορούμενες γραμματικές είναι άκρως ανεπιθύμητες στη συντακτική ανάλυση, καθώς δεν αποδίδουν μοναδική *«ανάγνωση»* ή *«σημασία»* στις λέξεις τους. Δυστυχώς, το πρόβλημα αν μία δεδομένη γραμματική είναι διφορούμενη είναι αναποκρίσιμο, και, ακόμα χειρότερα, αποδεικνύεται η ύπαρξη εγγενώς διφορούμενων γραμματικών. Από την άλλη, είναι παρήγορο το γεγονός ότι για πολλές χρήσιμες γραμματικές, όπως εκείνες που περιγράφουν τη δομή των προγραμμάτων μίας τυπικής γλώσσας προγραμματισμού, είναι πάντοτε δυνατό να απαλείψουμε την αμφισημία κατασκευάζοντας μία ισοδύναμη, μη διφορούμενη γραμματική.

1.5 Η Ιεραρχία Chomsky

Παρά τις επιθυμίες μας, το πρόβλημα εάν δύο δεδομένες γραμματικές είναι ισοδύναμες είναι και αυτό αναποκρίσιμο, κι έτσι δεν υπάρχει γενικός τρόπος να ταξινομήσουμε τις γραμματικές ως προς τη γλώσσα που παράγουν. Ο γνωστός γλωσσολόγος, και θεμελιωτής της θεωρίας τυπικών γραμματικών, Noam Chomsky, όμως, εισήγαγε τα έτη 1956 και 1959 μία μέθοδο ταξινόμησης των γραμματικών με βάση τη μορφή των κανόνων τους. Η ταξινόμηση αυτή, που στόχευε αρχικά (και μάταια) στην εύρεση κατάλληλων μοντέλων περιγραφής των φυσικών γλωσσών, τοποθετεί τις γραμματικές σε μία ιεραρχία τεσσάρων κλάσεων και έμεινε γνωστή ως **Ιεραρχία Chomsky**.

Έστω $G = (V, T, P, S)$ μία τυπική γραμματική. Η G είναι:

τύπου 0 αν δεν υπάρχει κανενός είδους πρόσθετος περιορισμός στη μορφή των κανόνων της,

τύπου 1 αν κάθε κανόνας στο P είναι της μορφής $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, όπου $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$, $A \in V$ και $\beta \neq \varepsilon$. Κατ' εξαίρεσιν επιτρέπεται στο P ο κανόνας $S \to \varepsilon$· τότε όμως το S δεν μπορεί να εμφανίζεται στο δεξί μέλος κανενός κανόνα,

τύπου 2 αν κάθε κανόνας στο P έχει τη μορφή $A \to \alpha$, όπου $A \in V$ και $\alpha \in (V \cup T)^*$,

τύπου 3 αν όλοι οι κανόνες στο P είναι σε μία από τις μορφές

3a. $A \to wB$ ή $A \to w$

3b. $A \to Bw$ ή $A \to w$,

όπου $A, B \in V$, $w \in T^*$.

Μία *γλώσσα* καλείται τύπου i $(i = 0, 1, 2, 3)$ εάν παράγεται από μία γραμματική του ίδιου τύπου. Η *κλάση των γλωσσών* τύπου i συμβολίζεται με $\mathcal{L}_i$.

Οι γραμματικές τύπου 0 καλούνται γραμματικές **χωρίς περιορισμούς** (unrestricted, phrase structure, semi-Thue), ενώ οι τύπου 1 ονομάζονται γραμματικές **με συμφραζόμενα** (context-sensitive), καθώς, σύμφωνα με τη μορφή τους, η αντικατάσταση μίας μεταβλητής με μία λέξη είναι δυνατή μόνο ανάμεσα σε συγκεκριμένα συμφραζόμενα. Αντίθετα, οι γραμματικές τύπου 2 ονομάζονται γραμματικές **χωρίς συμφραζόμενα** (context-free), αφού επιτρέπουν τέτοιες αντικαταστάσεις σε οποιοδήποτε περιβάλλον. Οι γραμματικές των οποίων οι κανόνες είναι στη μορφή 3a λέγονται **δεξιογραμμικές** (rightlinear) γραμματικές, ενώ όταν πρόκειται για τη μορφή 3b μιλάμε για **αριστερογραμμικές** (leftlinear). Δεν είναι δύσκολο να δειχτεί ότι οι δεξιογραμμικές και αριστερογραμμικές γραμματικές έχουν την ίδια παραγωγική ισχύ, δηλαδή είναι ισοδύναμες. Έτσι, ο συγκεκριμένος διαχωρισμός δεν είναι ουσιώδης, και οι γραμματικές τύπου 3 λέγονται γενικά **κανονικές** (regular) γραμματικές.

Βασικό είναι το παρακάτω θεώρημα, που μας πληροφορεί ότι όλες οι γλώσσες στην ιεραρχία διαθέτουν τις επιθυμητές ιδιότητες κλειστότητας, τουλάχιστον ως προς κάποιες θεμελιώδεις πράξεις. Η απόδειξη γίνεται ξεχωριστά για καθεμία από τις πράξεις και μπορεί να βρεθεί στο [16].

Θεώρημα 1.5.1. *Κάθε μία από τις κλάσεις $\mathcal{L}_i$, $i = 0, 1, 2, 3$, είναι κλειστή ως προς ένωση, παράθεση και Kleene star.*

Είναι προφανές από τους παραπάνω ορισμούς ότι κάθε γλώσσα τύπου 3 είναι και τύπου 2, όπως επίσης ότι κάθε γλώσσα οποιουδήποτε τύπου είναι τετριμμένα και τύπου 0. Αυτό σημαίνει ότι

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_0 \qquad \text{και} \qquad \mathcal{L}_1 \subseteq \mathcal{L}_0 \ ,$$

όμως η σχέση ανάμεσα στις κλάσεις $\mathcal{L}_2$ και $\mathcal{L}_1$ δεν είναι προφανής, λόγω του περιορισμού για το ε. Αποδεικνύεται πως και η συμπερίληψη

$$\mathcal{L}_2 \subseteq \mathcal{L}_1$$

αληθεύει. Επιπλέον, όπως θα δούμε, όλες οι παραπάνω συμπεριλήψεις είναι γνήσιες, δίνοντάς μας μία γνήσια ιεραρχία:

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0.$$

Στη συνέχεια θα εξετάσουμε αναλυτικά κάθε μία από τις κλάσεις γλωσσών που συνιστούν την ιεραρχία Chomsky. Θα διερευνήσουμε επίσης μία ιεραρχία όλο και πιο εξελιγμένων υπολογιστικών μοντέλων—*αυτομάτων*—που αντιστοιχεί ακριβώς στην παραπάνω ιεραρχία γλωσσών.

Κεφάλαιο 2

Κανονικές Γλώσσες

Ξεκινώντας τη μελέτη της ιεραρχίας Chomsky από τη βάση προς την κορυφή, δηλαδή από τις πιο περιορισμένες κλάσεις γλωσσών προς τις ευρύτερες, εξετάζουμε κατ' αρχάς την κλάση $\mathcal{L}_3$, τις κανονικές γλώσσες. Οι γλώσσες αυτές συνδέονται άρρηκτα με τα πεπερασμένα αυτόματα, που θα ορίσουμε παρακάτω, και τα οποία, παρά τις περιορισμένες δυνατότητές τους, βρίσκουν πληθώρα εφαρμογών στο σχεδιασμό πολλών κοινών τύπων αλγορίθμων και προγραμμάτων. Χαρακτηριστικά παραδείγματα αποτελούν η φάση της λεκτικής ανάλυσης ενός μεταγλωττιστή (κατά την οποία αναγνωρίζονται στοιχεία ενός προγράμματος, όπως το «`begin`» και το «`if`»), αλλά και το πρόβλημα εντοπισμού εμφάνισης μίας λέξης.

2.1 Κανονικές εκφράσεις

Πέρα από τη διάκρισή τους σε δεξιογραμμικές και αριστερογραμμικές, οι κανονικές γλώσσες έχουν ακόμα έναν ενδιαφέροντα χαρακτηρισμό. Είναι σαφές κατ' αρχάς ότι κάθε πεπερασμένη γλώσσα είναι κανονική, αφού παράγεται από την κανονική γραμματική που έχει τον κανόνα $S \to w$ για κάθε λέξη w της γλώσσας. Επιπλέον όμως, ξέρουμε από το Θεώρημα 1.5.1 ότι η κλάση των κανονικών γλωσσών είναι κλειστή ως προς ένωση, παράθεση και Kleene star. Κατά συνέπεια, αν ξεκινήσουμε από ένα πεπερασμένο σύνολο λέξεων και εφαρμόσουμε σε αυτές τις παραπάνω πράξεις, η γλώσσα που θα προκύψει θα είναι κανονική. Σε αυτή τη μέθοδο συνίσταται η ιδέα των **κανονικών εκφράσεων** ενός αλφαβήτου Σ, που γεννήθηκε από τον S. C. Kleene το 1956 με στόχο τη μελέτη ιδιοτήτων των συστημάτων νεύρων (nerve nets) και ορίζεται αυστηρά ως εξής:

1. Οι σταθερές ε και $\emptyset$, καθώς και κάθε στοιχείο $a \in \Sigma$ είναι κανονικές

εκφράσεις.

2. Αν οι α και β είναι κανονικές εκφράσεις, τότε και η $\alpha+\beta$ είναι κανονική έκφραση.

3. Αν οι α και β είναι κανονικές εκφράσεις, τότε και η $\alpha\beta$ είναι κανονική έκφραση.

4. Αν η α είναι κανονική έκφραση, τότε και η α^* είναι κανονική έκφραση.

Οι κανονικές εκφράσεις αναπαριστούν κανονικές γλώσσες, όπως ορίζει η συνάρτηση $\mathscr{L}$, που σε κάθε κανονική έκφραση α αντιστοιχεί τη γλώσσα $\mathscr{L}(\alpha)$ που αναπαρίσταται από την α με τον ακόλουθο τρόπο:

1. $\mathscr{L}(\varepsilon) = \{\varepsilon\}$, $\mathscr{L}(\emptyset) = \varnothing$ και $\mathscr{L}(a) = \{a\}$ για κάθε $a \in \Sigma$.

2. Αν οι α και β είναι κανονικές εκφράσεις, τότε $\mathscr{L}(\alpha+\beta) = \mathscr{L}(\alpha) \cup \mathscr{L}(\beta)$.

3. Αν οι α και β είναι κανονικές εκφράσεις, τότε $\mathscr{L}(\alpha\beta) = \mathscr{L}(\alpha)\mathscr{L}(\beta)$.

4. Αν η α είναι κανονική έκφραση, τότε $\mathscr{L}(\alpha^*) = (\mathscr{L}(\alpha))^*$.

Για παράδειγμα, η $\emptyset^*$ είναι μία κανονική έκφραση που αναπαριστά τη γλώσσα $\mathscr{L}(\emptyset^*) = \{\varepsilon\}$ και η 0^*10^* είναι μία κανονική έκφραση που αναπαριστά την κανονική γλώσσα $\mathscr{L}(0^*10^*) = \{w \in \{0,1\}^* \mid \text{η } w \text{ περιέχει ακριβώς ένα } 1\}$. Η $(a+b)^*a$ είναι μία κανονική έκφραση με $\mathscr{L}((a+b)^*a) = \{a,b\}^*\{a\} = \{w \in \{a,b\}^* \mid \text{η } w \text{ τελειώνει σε } a\}$.

Το ερώτημα που ανακύπτει είναι εάν ισχύει και το αντίστροφο, αν, δηλαδή, τελικά μια γλώσσα είναι κανονική τότε και μόνο τότε, όταν υπάρχει μία κανονική έκφραση που την αναπαριστά. Εύκολα αποδεικνύεται ότι η απάντηση είναι καταφατική.

Θεώρημα 2.1.1. *Κάθε κανονική έκφραση αναπαριστά μία κανονική γλώσσα και, αντίστροφα, κάθε κανονική γλώσσα αναπαρίσταται από μία κανονική έκφραση.*

Ιδέα απόδειξης. Το ευθύ είναι άμεση συνέπεια του Θεωρήματος 1.5.1, όπως εξηγήσαμε στην αρχή της ενότητας. Για το αντίστροφο, δεδομένης μιας κανονικής γλώσσας L, κατασκευάζουμε μία γλώσσα που ισούται με την L και για την οποία δείχνεται εύκολα ότι αναπαρίσταται από κανονική έκφραση. Λεπτομέρειες βρίσκονται στο [16]. □

2.2 Πεπερασμένα Αυτόματα

Ως το σημείο αυτό χρησιμοποιήσαμε διαφόρων ειδών γραμματικές για να παράγουμε τυπικές γλώσσες. Φαίνεται όμως αρκετά λογικό και να ορίσει κανείς μία τυπική γλώσσα με τη βοήθεια κάποιας μηχανής που επεξεργάζεται σύμβολα και συμβολοσειρές. Μια τέτοια μηχανή είναι το **πεπερασμένο αυτόματο** (finite automaton) ή **μηχανή πεπερασμένων καταστάσεων** (finite state machine), που συνιστά ένα εξαιρετικά περιορισμένο μοντέλο πραγματικού υπολογιστή. Το πεπερασμένο αυτόματο αποτελεί ένα μοντέλο υπολογιστή με πεπερασμένη και προκαθορισμένη μνήμη, που δέχεται ως είσοδο μία συμβολοσειρά και δεν παράγει καθόλου έξοδο, παρά μόνο μία ένδειξη για το εάν η είσοδος θεωρείται αποδεκτή ή όχι.

Αν και ένα τόσο απλό υπολογιστικό μοντέλο ίσως να μοιάζει υπερβολικά απλοϊκό ώστε να αξίζει σοβαρή μελέτη, στην πραγματικότητα μπορεί να κάνει πολλά χρήσιμα πράγματα. Μάλιστα αλληλεπιδρούμε καθημερινά με τέτοιου είδους μηχανές, καθώς βρίσκονται στην καρδιά διαφόρων ηλεκτρομηχανικών συσκευών. Για να ονοματίσουμε μερικές μόνο, οι επεξεργαστές αυτόματων θυρών, ανελκυστήρων, ηλεκτρικών θερμοστατών, όπως επίσης και πολλών οικιακών συσκευών, είναι σχεδιασμένοι με βάση τη μεθοδολογία των πεπερασμένων αυτομάτων.[1]

Βέβαια, όπως έχουμε ήδη αναφέρει, υπάρχει και πληθώρα εφαρμογών «γλωσσολογικής» φύσης. Τα πεπερασμένα αυτόματα και οι πιθανοτικοί ομόλογοί τους, οι **αλυσίδες Markov**[2] (Markov chains), είναι βασικά εργαλεία για την αναγνώριση λεκτικών σχημάτων (patterns) σε δεδομένα και χρησιμοποιούνται ευρέως στην επεξεργασία λόγου (speech processing). Εκτός αυτού, πλήθος άλλων χρήσεων έχει βρεθεί για τα πεπερασμένα αυτόματα τόσο σε διαφόρων ειδών προγράμματα επεξεργασίας κειμένου ή αναζήτησης φακέλων, όσο και ως μαθηματικές έννοιες με εφαρμογή σε άλλες περιοχές, όπως αυτή της λογικής.

Επανερχόμενοι όμως στη σχέση των υπολογιστικών μοντέλων με τις τυπικές γραμματικές, δε θα ήταν παρακινδυνευμένο να υποστηρίξουμε πως, κατά μία έννοια, οι γραμματικές, από τη μία, και τα αυτόματα, από την άλλη, αντι-

[1] Ακόμα και ο ίδιος ο ηλεκτρονικός υπολογιστής θα μπορούσε πιθανόν να θεωρηθεί ως ένα πεπερασμένο αυτόματο, αν υποθέσουμε ότι οι ικανότητες μνήμης του είναι πεπερασμένες. Όμως μία τέτοια αντίληψη δε δείχνει να ωφελεί, καθώς η επιβολή ενός τεχνητού ορίου στη μνήμη δε φαίνεται να συμβαδίζει με την έννοια του υπολογισμού. Για να πάμε ακόμα πιο μακριά, και ο ανθρώπινος εγκέφαλος θα ήταν ενδεχομένως δυνατό να μοντελοποιηθεί ως ένα πεπερασμένο αυτόματο. Στην προκειμένη περίπτωση όμως το μέγεθος του αυτομάτου θα ήταν τόσο μεγάλο, που μία τέτοια προσέγγιση δύσκολα θα είχε κάποιο πρακτικό όφελος.

[2] Οι αλυσίδες Markov έχουν χρησιμοποιηθεί ακόμα και για την πρόβλεψη αλλαγών στις τιμές των χρηματιστηρίων.

κατοπτρίζουν τις δύο όψεις του ίδιου νομίσματος, δηλαδή της χρήσης μιας γλώσσας. Συγκεκριμένα, ο ομιλητής (η πηγή) ενεργεί ως *παραγωγός γλώσσας*, χρησιμοποιώντας τη γραμματική ως μέσο σύνθεσης, ενώ ο ακροατής (ο δέκτης) ενεργεί ως *αναγνωριστής γλώσσας* και χρησιμοποιεί το αυτόματο ως μέσο ανάλυσης. Η εικόνα αυτή σαφώς θα γίνει περισσότερο αντιληπτή στην πορεία, όπου θα δούμε από αυστηρή σκοπιά τη λειτουργία των αυτομάτων.

2.2.1 Ντετερμινιστικά πεπερασμένα αυτόματα

Ένα πεπερασμένο αυτόματο αποτελείται κατά βάση από ένα πεπερασμένο σύνολο καταστάσεων και από ένα σύνολο δυνατών μεταβάσεων από τη μία κατάσταση στην άλλη, που πραγματοποιούνται με την ανάγνωση της συμβολοσειράς με την οποία έχει τροφοδοτηθεί ως είσοδο. Ο όρος «ντετερμινιστικό» αναφέρεται στο γεγονός ότι για κάθε είσοδο υπάρχει μία μοναδική κατάσταση στην οποία το αυτόματο μπορεί να μεταβεί από την τρέχουσα, σε αντίθεση με το «μη ντετερμινιστικό», που θα δούμε αμέσως μετά.

Αυστηρά, ένα **ντετερμινιστικό πεπερασμένο αυτόματο** (deterministic finite automaton) DFA είναι μία διατεταγμένη πεντάδα $A = (Q, \Sigma, \delta, q_0, F)$, όπου

- Q είναι ένα μη κενό, πεπερασμένο σύνολο, τα στοιχεία του οποίου καλούνται **καταστάσεις** (states),
- Σ είναι ένα αλφάβητο, το **αλφάβητο εισόδου** (input alphabet),
- $\delta : Q \times \Sigma \to Q$ καλείται **συνάρτηση μετάβασης** (transition function),
- $q_0 \in Q$ είναι η **αρχική κατάσταση** (start state) και
- $F \subseteq Q$ είναι το σύνολο των **τελικών καταστάσεων** (final, accepting states).

Αρχικά το A είναι στην κατάσταση q_0 και διαβάζει το πρώτο σύμβολο της εισόδου του, που είναι μια συμβολοσειρά από το Σ. Τότε εκτελεί μία ακολουθία μεταβάσεων ή κινήσεων[3] από μία κατάσταση σε άλλη, όπως καθορίζει

[3]Είναι προφανές τι εννοούμε με τον όρο **μετάβαση** ή **κίνηση** (move), που αυστηρά μπορεί να οριστεί ως μία τριάδα $\mu = (p, a, q) \in Q \times \Sigma \times Q$, όπου $q \in \delta(p, \alpha)$. Η μ καλείται **μη ντετερμινιστική** εάν $|\delta(p, a)| > 1$.

Επίσης, θα χρησιμοποιούμε συχνά τον όρο **υπολογισμός** (computation) **για τη λέξη** $w = a_1 \dots a_n \in \Sigma^*$, που είναι μια ακολουθία κινήσεων $\mu_1 \dots \mu_n$, όπου $\mu_i = (p_{i-1}, a_i, p_i)$ και $p_0 = q_0$. Ο παραπάνω υπολογισμός θα λέμε ότι **οδηγεί σε αποδοχή της** w, αν $p_n \in F$. Στο εξής θα χρησιμοποιήσουμε τις παραπάνω έννοιες και για άλλα υπολογιστικά μοντέλα, παραλείποντας τους αυστηρούς ορισμούς.

η δ, ενώ διαβάζει ένα-ένα τα υπόλοιπα σύμβολα της εισόδου. Έτσι, αν το A είναι στην κατάσταση $q \in Q$ και το σύμβολο που διαβάζει είναι το $a \in \Sigma$, τότε η $\delta(q, a)$ είναι η μονοσήμαντα ορισμένη κατάσταση στην οποία περνά. Αν με την ανάγνωση του τελευταίου συμβόλου βρεθεί σε μία κατάσταση από το F, τότε η είσοδος γίνεται δεκτή—διαφορετικά απορρίπτεται.

Είναι φανερό ότι ένα ντετερμινιστικό πεπερασμένο αυτόματο είναι αδύνατον να μπει σε άπειρη ακολουθία κινήσεων (infinite loops) αν η είσοδός του είναι πεπερασμένη, αφού για κάθε νέα κίνηση οφείλει να διαβάσει ένα σύμβολο από την είσοδο. Επομένως, μετά από ισάριθμες με το μήκος της εισόδου κινήσεις, είναι αναγκασμένο να δώσει μία σαφή απάντηση για το εάν τη δέχεται ή όχι.

Η λειτουργία ενός δεδομένου DFA καθορίζεται λοιπόν από την τρέχουσα κατάσταση και το τμήμα της συμβολοσειράς που δεν έχει ακόμα διαβαστεί. Έτσι, ορίζουμε ως **συνολική κατάσταση** (configuration) ή **στιγμιαία περιγραφή** (instantaneous description) του DFA $A = (Q, \Sigma, \delta, q_0, F)$ ένα οποιοδήποτε στοιχείο του $Q \times \Sigma^*$. Αν (q, w), (q', w') είναι δύο συνολικές καταστάσεις του A, λέμε ότι η (q, w) **παράγει** (yields) την (q', w') **σε ένα βήμα**, και γράφουμε

$$(q, w) \vdash_A (q', w'),$$

αν και μόνο αν $w = aw'$ και $\delta(q, a) = q'$, για κάποιο $a \in \Sigma$. Συμβολίζουμε την ανακλαστική, μεταβατική κλειστότητα της σχέσης αυτής με $\vdash_A^*$ και η

$$(q, w) \vdash_A^* (q', w')$$

διαβάζεται: η (q, w) **παράγει** την (q', w'). Μία **συμβολοσειρά** $w \in \Sigma^*$ **γίνεται δεκτή** (is accepted) από το A αν και μόνο αν υπάρχει $q \in F$, έτσι ώστε

$$(q_0, w) \vdash_A^* (q, \varepsilon).$$

Τέλος, **η γλώσσα που γίνεται δεκτή από το** A ή **η γλώσσα του** A είναι

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash_A^* (q, \varepsilon) \text{ για κάποιο } q \in F\},$$

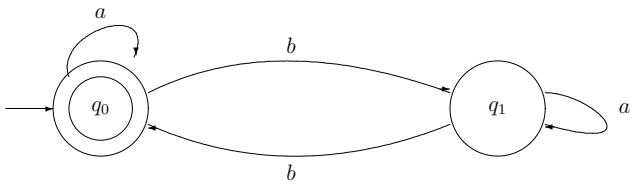
δηλαδή το σύνολο των λέξεων που δέχεται το A.

Παράδειγμα 2.2.1. Απεικονίζουμε με δύο εύχρηστους τρόπους, τον πίνακα καταστάσεων (transition table) και το διάγραμμα καταστάσεων (transition diagram), ένα DFA A που δέχεται τη γλώσσα $L(A) = \{w \in \{a, b\}^* \mid \text{η } w \text{ περιέχει άρτιο πλήθος από } b\}$.

Στον πίνακα καταστάσεων σημειώνουμε την αρχική κατάσταση με $\rightarrow$ και τις τελικές με $*$, ενώ στο διάγραμμα καταστάσεων η αρχική κατάσταση ξεχωρίζει χάρη στο βέλος χωρίς σημείο εκκίνησης που οδηγεί σε αυτή, και οι

	a	b
$* \longrightarrow q_0$	q_0	q_1
q_1	q_1	q_0

Πίνακας 2.1: Πίνακας καταστάσεων για το DFA του Παραδείγματος 2.2.1. Η είσοδος του πίνακα για τη γραμμή q και τη στήλη x είναι η τιμή $\delta(q, x)$.



Σχήμα 2.1: Διάγραμμα καταστάσεων για το DFA του Παραδείγματος 2.2.1.

τελικές από το διπλό κύκλο. Στην προκειμένη περίπτωση τυχαίνει η μοναδική τελική κατάσταση q_0 να συμπίπτει με την αρχική.

2.2.2 Μη ντετερμινιστικά πεπερασμένα αυτόματα

Στα πεπερασμένα αυτόματα που είδαμε κάθε βήμα ενός υπολογισμού ακολουθεί το προηγούμενο κατά μοναδικό τρόπο, δηλαδή ο υπολογισμός είναι ντετερμινιστικός. Η έννοια της *μη ντετερμινιστικής* μηχανής, που κατά κάποιο τρόπο τυποποιεί την ανθρώπινη συμπεριφορά του *μαντέματος*, δεν προορίζεται για την περιγραφή ρεαλιστικών υπολογιστικών μοντέλων. Είναι όμως εξαιρετικής χρησιμότητας στη θεωρία υπολογισμού, καθώς συχνά επιτυγχάνει να απλοποιήσει σημαντικά την περιγραφή των αυτομάτων, όπως, βέβαια, και τις αποδείξεις που σχετίζονται με αυτά.

Το μη ντετερμινιστικό πεπερασμένο αυτόματο διαφέρει από το ντετερμινιστικό στο σημείο ότι στο πρώτο, για μία δεδομένη συνολική κατάσταση, είναι επιτρεπτές μηδέν, μία, ή περισσότερες μεταβάσεις, και, επιπλέον, επιτρέπεται η αλλαγή κατάστασης χωρίς καθόλου ανάγνωση εισόδου. Αυστηρά λοιπόν, ένα **μη ντετερμινιστικό πεπερασμένο αυτόματο** (nondeterministic finite automaton) NFA είναι μία διατεταγμένη πεντάδα $A = (Q, \Sigma, \delta, q_0, F)$, όπου

- Q είναι ένα μη κενό, πεπερασμένο σύνολο **καταστάσεων**,
- Σ είναι το **αλφάβητο εισόδου**,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ είναι η **συνάρτηση μετάβασης**,

- $q_0 \in Q$ είναι η **αρχική κατάσταση** και
- $F \subseteq Q$ είναι το σύνολο των **τελικών καταστάσεων**.

Έτσι, η συνάρτηση μετάβασης ενός NFA παίρνει τιμές στο δυναμοσύνολο του συνόλου καταστάσεων και για κάθε κατάσταση και κάθε στοιχείο του $\Sigma \cup \{\varepsilon\}$ επιστρέφει ένα *σύνολο* καταστάσεων. Αυτό σημαίνει ότι ο μη ντετερμινισμός μπορεί να θεωρηθεί ως ένα είδος *παράλληλου υπολογισμού*, κατά τον οποίο διάφορες «διεργασίες» εκτελούνται ταυτόχρονα.

Όπως και στην περίπτωση του ντετερμινιστικού πεπερασμένου αυτομάτου, η **συνολική κατάσταση** του NFA $A = (Q, \Sigma, \delta, q_0, F)$ είναι και πάλι οποιοδήποτε στοιχείο του $Q \times \Sigma^*$. Αν (q, w), (q', w') είναι δύο συνολικές καταστάσεις του A, η (q, w) **παράγει** την (q', w') **σε ένα βήμα**,

$$(q, w) \vdash_A (q', w'),$$

αν και μόνο αν υπάρχει $a \in \Sigma \cup \{\varepsilon\}$, τέτοιο ώστε $w = aw'$ και $q' \in \delta(q, a)$. Παρατηρούμε ότι η $\vdash_A$ δεν είναι πια υποχρεωτικά συνάρτηση, αλλά σχέση, αφού μια συνολική κατάσταση μπορεί να παράγει πολλές συνολικές καταστάσεις, ή και καμία. Οι υπόλοιπες έννοιες που σχετίζονται με τα πεπερασμένα αυτόματα μένουν αναλλοίωτες με την προσθήκη του μη ντετερμινισμού. Η ανακλαστική, μεταβατική κλειστότητα της $\vdash_A$ συμβολίζεται πάλι με $\vdash_A^*$. Μία **συμβολοσειρά** $w \in \Sigma^*$ **γίνεται δεκτή** από το A αν και μόνο αν υπάρχει $q \in F$, έτσι ώστε

$$(q_0, w) \vdash_A^* (q, \varepsilon),$$

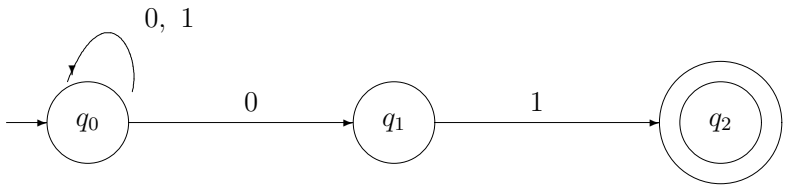
και **η γλώσσα του** A είναι η

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \vdash_A^* (q, \varepsilon) \text{ για κάποιο } q \in F\}.$$

Παράδειγμα 2.2.2. Το NFA A με τον Πίνακα καταστάσεων 2.2 και το διάγραμμα καταστάσεων του Σχήματος 2.2 δέχεται τη γλώσσα $L(A) = \{w \in \{0,1\}^* \mid \text{η } w \text{ τελειώνει σε } 01\}$.

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$* \; q_2$	$\emptyset$	$\emptyset$

Πίνακας 2.2: Πίνακας καταστάσεων για το NFA του Παραδείγματος 2.2.2.



Σχήμα 2.2: Διάγραμμα καταστάσεων για το NFA του Παραδείγματος 2.2.2.

Είναι σαφές ότι ένα ντετερμινιστικό πεπερασμένο αυτόματο δεν είναι παρά μια ειδική περίπτωση μη ντετερμινιστικού. Επομένως η κλάση των γλωσσών που γίνονται δεκτές από ντετερμινιστικά αυτόματα είναι υποσύνολο εκείνης των γλωσσών που γίνονται δεκτές από μη ντετερμινιστικά. Παρ' όλη όμως την προφανή ισχύ και γενικότητα των μη ντετερμινιστικών αυτομάτων, αυτά δεν προκύπτουν πιο ισχυρά από τα ντετερμινιστικά σε ό,τι αφορά τις γλώσσες που δέχονται. Οι παραπάνω κλάσεις είναι στην πραγματικότητα ίσες, γεγονός όχι μόνο εντυπωσιακό, αλλά και ιδιαίτερα χρήσιμο, καθώς τις περισσότερες φορές η περιγραφή μίας δεδομένης γλώσσας γίνεται πολύ πιο εύκολα μέσω ενός NFA παρά ενός DFA. Λέμε ότι δύο αυτόματα A_1 και A_2 είναι **ισοδύναμα** (equivalent) εάν δέχονται την ίδια γλώσσα, δηλαδή $L(A_1) = L(A_2)$.

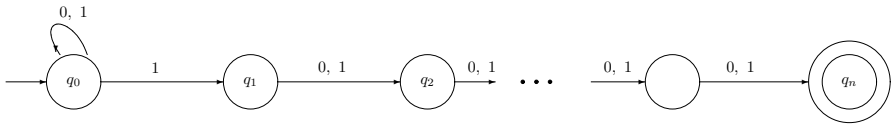
Θεώρημα 2.2.1. *Για κάθε μη ντετερμινιστικό πεπερασμένο αυτόματο υπάρχει ένα ισοδύναμο ντετερμινιστικό πεπεπρασμένο αυτόματο.*

Ιδέα απόδειξης. Μία βαθύτερη εξέταση της έννοιας του μη ντετερμινισμού στην περίπτωση των πεπερασμένων αυτομάτων αρκεί για να καταλάβουμε την ιδέα της απόδειξης. Πράγματι, το σύνολο των τρεχουσών καταστάσεων ενός NFA είναι, όπως είδαμε, ένα υποσύνολο του πεπερασμένου συνόλου καταστάσεων. Έτσι, μπορούμε να φανταστούμε τον τρόπο λειτουργίας του ως ένα, διαφορετικό κάθε στιγμή, σύνολο τρεχουσών καταστάσεων. Κάθε τέτοιο σύνολο όμως είναι δυνατόν να θεωρηθεί ως *μία μοναδική κατάσταση* ενός DFA με τόσες καταστάσεις, όσα και τα υποσύνολα του συνόλου καταστάσεων του NFA. Με αυτόν τον τρόπο προσομοιώνουμε τη συμπεριφορά του NFA μέσω ενός DFA, που όμως διαθέτει εκθετικά περισσότερες καταστάσεις. Η κατασκευαστική αυτή μέθοδος είναι γνωστή ως *μέθοδος κατασκευής υποσυνόλου* (subset construction) και περιγράφεται σε πλήρη έκταση στο [6]. □

Το γεγονός ότι το DFA που παράγεται από ένα NFA με τη μέθοδο κατασκευής υποσυνόλου έχει ως σύνολο καταστάσεων το δυναμοσύνολο του συνόλου καταστάσεων του NFA, έχει ως συνέπεια την εκθετική αύξηση του

πλήθους καταστάσεων του αρχικού NFA. Πολύ συχνά, όμως, συμβαίνει κάποιες από τις νέες καταστάσεις να είναι απροσπέλαστες, και επομένως άχρηστες. Έτσι, στην πράξη το DFA έχει συνήθως τόσες περίπου καταστάσεις, όσες και το NFA, αν και συχνά αρκετά περισσότερες μεταβάσεις. Ωστόσο, αυτό δεν είναι πάντα εφικτό· υπάρχουν περιπτώσεις, στις οποίες το μικρότερο DFA που δέχεται μία γλώσσα έχει 2^n καταστάσεις, ενώ το μικρότερο ισοδύναμο NFA μόνο n.

Στο Σχήμα 2.3 απεικονίζεται το διάγραμμα καταστάσεων ενός NFA με $n+1$ καταστάσεις, που δέχεται όλες τις συμβολοσειρές από 0 και 1, των οποίων το n-οστό σύμβολο από το τέλος είναι 1. Επειδή ένα οποιοδήποτε ισοδύναμο DFA θα πρέπει να «θυμάται» τα τελευταία n σύμβολα που έχει διαβάσει, εύκολα βλέπουμε πως δεν μπορεί να έχει λιγότερες από 2^n καταστάσεις. Σχετικά με αυτό το θέμα θα μιλήσουμε ξανά στο Κεφάλαιο 5.



Σχήμα 2.3: Μία «κακή» περίπτωση για τη μέθοδο κατασκευής υποσυνόλου.

2.2.3 Πεπερασμένα αυτόματα δύο κατευθύνσεων

Θεωρήσαμε το πεπερασμένο αυτόματο ως ένα είδος μονάδας ελέγχου με μία κεφαλή και μία ταινία εισόδου, στην οποία η κεφαλή μπορεί να κινείται διαβάζοντας το περιεχόμενό της μόνο προς τα δεξιά. Στη συνέχεια του προσθέσαμε το μη ντετερμινισμό, που επέτρεψε σε πολλά «αντίγραφα» της μονάδας ελέγχου να λειτουργούν παράλληλα. Είδαμε ότι αυτό το πλεονέκτημα δεν αύξησε την ισχύ του ντετερμινιστικού πεπερασμένου αυτομάτου από υπολογιστικής πλευράς. Μία άλλη ενδιαφέρουσα επέκταση, που ομοίως αδυνατεί να το ενισχύσει υπολογιστικά, είναι η δυνατότητα κίνησης της κεφαλής όχι μόνο προς τα δεξιά, αλλά και προς τα αριστερά.

Ένα **ντετερμινιστικό πεπερασμένο αυτόματο δύο κατευθύνσεων** (two-way deterministic finite automaton 2DFA) είναι μία διατεταγμένη πεντάδα

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

όπου τα Q, Σ, q_0, F είναι όπως ορίστηκαν στο ντετερμινιστικό πεπερασμένο αυτόματο και $\delta : Q \times \Sigma \to Q \times \{L, R\}$ είναι η συνάρτηση μετάβασης. Αν για $q, p \in Q$ και $a \in \Sigma$, $\delta(q, a) = (p, L)$, τότε στην κατάσταση q και διαβάζοντας

το σύμβολο a, το 2DFA περνά στην κατάσταση p κινώντας την κεφαλή του ένα σύμβολο αριστερά· αν $\delta(q,a) = (p,R)$, τότε περνά στην κατάσταση p κινώντας την κεφαλή του ένα σύμβολο δεξιά.

Ως **συνολική κατάσταση** του 2DFA $A = (Q, \Sigma, \delta, q_0, F)$ ορίζουμε ένα στοιχείο του $Q \times \Sigma^* \times \Sigma^*$ και η συνολική κατάσταση $(q, \alpha, X\beta)$ υποδηλώνει τότε ότι η μηχανή είναι στην κατάσταση q, με την κεφαλή να διαβάζει το X και με α στα αριστερά της. Αν $(q, \alpha X, Y\beta)$, (q', α', β') είναι δύο συνολικές καταστάσεις του A, η $(q, \alpha X, Y\beta)$ **παράγει** την (q', α', β') **σε ένα βήμα**,

$$(q, \alpha X, Y\beta) \vdash_A (q', \alpha', \beta'),$$

αν και μόνο αν $\delta(q, Y) = (q', \mathbf{D})$ και ισχύει ένα από τα παρακάτω:

1. $\mathbf{D} = L, \qquad \alpha' = \alpha, \qquad \beta' = XY\beta$
2. $\mathbf{D} = R, \qquad \alpha' = \alpha XY, \qquad \beta' = \beta$

Η ανακλαστική, μεταβατική κλειστότητα της $\vdash_A$ συμβολίζεται με $\vdash_A^*$. Το 2DFA δέχεται μία συμβολοσειρά εάν φτάσει στο δεξί άκρο της ταινίας εισόδου και την ίδια στιγμή βρεθεί σε τελική κατάσταση. Έτσι, μία **συμβολοσειρά** $w \in \Sigma^*$ **γίνεται δεκτή** από το A αν υπάρχει $q \in F$, τέτοιο ώστε

$$(q_0, \varepsilon, w) \vdash_A^* (q, w, \varepsilon).$$

Επομένως, ως **γλώσσα** του A ορίζεται η

$$L(A) = \{w \in \Sigma^* \mid (q_0, \varepsilon, w) \vdash_A^* (q, w, \varepsilon) \text{ για κάποιο } q \in F\}.$$

Παράδειγμα 2.2.3. Το 2DFA A με τον Πίνακα καταστάσεων 2.3 έχει τρεις καταστάσεις, που όλες είναι τελικές. Ξεκινώντας από την κατάσταση q_0, το A επαναλαμβάνει την εξής ακολουθία κινήσεων: μετακινεί την κεφαλή του προς τα δεξιά μέχρι να συναντήσει δύο 1 (όχι κατ' ανάγκην διαδοχικά). Μόλις συμβεί αυτό, αρχίζει να την κινεί προς τα αριστερά μέχρι να διαβάσει ένα 0, οπότε και ξαναμπαίνει στην κατάσταση q_0, ξαναρχίζοντας τη διαδικασία.

Παρατηρώντας τον Πίνακα 2.3 γίνεται σαφές ότι για να φτάσει στο δεξί άκρο της ταινίας εισόδου η κεφαλή του A, πρέπει και αρκεί να μη συναντήσει

	0	1
$* \to q_0$	(q_0, R)	(q_1, R)
$\to q_1$	(q_1, R)	(q_2, L)
$\to q_2$	(q_0, R)	(q_2, L)

Πίνακας 2.3: Πίνακας καταστάσεων για το 2DFA του Παραδείγματος 2.2.3.

δύο διαδοχικά 1. Άρα η γλώσσα του είναι η $L(A) = \{w \in \{0,1\}^* \mid \eta\ w$ δεν περιέχει δύο διαδοχικά $1\}$.

Φυσικά, το πεπερασμένο αυτόματο δύο κατευθύνσεων μπορεί να οριστεί και στη μη ντετερμινιστική εκδοχή. Έτσι, ανάλογα με τον τρόπο με τον οποίο συνδυάζουν ντετερμινισμό ή μη, και κίνηση σε μία κατεύθυνση ή δύο, τα πεπερασμένα αυτόματα χωρίζονται σε τέσσερις κατηγορίες:

- Ντετερμινιστικά πεπερασμένα αυτόματα (DFAs ή 1DFAs)
- Μη ντετερμινιστικά πεπερασμένα αυτόματα (NFAs ή 1NFAs)
- Ντετερμινιστικά πεπερασμένα αυτόματα δύο κατευθύνσεων (2DFAs)
- Μη ντετερμινιστικά πεπερασμένα αυτόματα δύο κατευθύνσεων (2NFAs)

Όπως πιθανότατα μαντεύει κανείς, ούτε και τα 2NFAs παρουσιάζουν αυξημένες υπολογιστικές δυνατότητες σε σχέση με τα άλλα τρία μοντέλα, με αποτέλεσμα και οι τέσσερις κλάσεις αυτομάτων να δέχονται τις ίδιες ακριβώς γλώσσες. Όμως από μία άλλη οπτική γωνία, αυτή της *πολυπλοκότητας*, τα πράγματα δεν είναι καθόλου τόσο απλά. Γι' αυτή θα πούμε περισσότερα στο Κεφάλαιο 5.

2.3 Πεπερασμένα αυτόματα και κανονικές εκφράσεις

Το κύριο συμπέρασμα της τελευταίας ενότητας ήταν ότι η κλάση των γλωσσών που γίνονται δεκτές από τα πεπερασμένα αυτόματα παραμένει η ίδια, έστω κι αν προστεθεί το τόσο ισχυρό χαρακτηριστικό του μη ντετερμινισμού. Αυτό υποδεικνύει ότι η συγκεκριμένη κλάση διαθέτει κάποιο είδος *σταθερότητας*, εφόσον ορίζεται από δύο διαφορετικές προσεγγίσεις. Στην ενότητα αυτή θα δούμε ακόμα ένα σημαντικό χαρακτηριστικό της ίδιας κλάσης, περαιτέρω απόδειξη για το πόσο αξιοθαύμαστα σταθερή είναι: Η κλάση των γλωσσών που γίνονται δεκτές από πεπερασμένα αυτόματα—ντετερμινιστικά ή μη, μίας ή δύο κατευθύνσεων—είναι η ίδια κλάση με εκείνη των κανονικών γλωσσών, των γλωσσών, δηλαδή, που παράγονται από κανονικές γραμματικές.

Το γεγονός ότι οι κανονικές γραμματικές και τα πεπερασμένα αυτόματα είναι ισοδύναμα ως προς την παραγωγική ισχύ τους δείχνει μάλλον αξιοσημείωτο, εάν σκεφτεί κανείς πόσο (επιφανειακά) διαφορετικοί είναι οι εμπλεκόμενοι μηχανισμοί στην κάθε περίπτωση. Όμως αν εμμείνουμε στις ομοιότητες που βρίσκονται στην ουσία των δύο κατασκευών, δεν είναι δύσκολο να αντιληφθούμε ότι οι υπολογισμοί ενός πεπερασμένου αυτομάτου αντιστοιχούν σε

αντίστροφες παραγωγές μιας αριστερογραμμικής γραμματικής. Στο [16] δείχνεται και αυστηρά πώς μπορούμε να κατασκευάσουμε μία κανονική γραμματική από ένα πεπερασμένο αυτόματο, και αντίστροφα.

Παρ' όλα αυτά, η παραπάνω ισοδυναμία είναι ίσως πιο άμεσο διαισθητικά να δειχτεί χρησιμοποιώντας την έννοια των κανονικών εκφράσεων, όπως την ορίσαμε στην Ενότητα 2.1. Συγκεκριμένα, δείχνεται ότι κάθε γλώσσα που γίνεται δεκτή από κάποιον τύπο πεπερασμένου αυτομάτου (για λόγους ευκολίας υποθέτουμε DFA) αναπαρίσταται από κάποια κανονική έκφραση, αλλά και αντίστροφα, ότι κάθε γλώσσα που αναπαρίσταται από κανονική έκφραση γίνεται δεκτή από κάποιον τύπο πεπερασμένου αυτομάτου (για την κατεύθυνση αυτή είναι ευκολότερο να υποθέσουμε NFA).

Πρόταση 2.3.1. *Αν $L = L(A)$ για ένα DFA A, τότε υπάρχει κανονική έκφραση α, τέτοια ώστε $L = \mathscr{L}(\alpha)$.*

Ιδέα απόδειξης. Η ιδέα είναι να κατασκευάσουμε τις κανονικές εκφράσεις που αναπαριστούν τα σύνολα συμβολοσειρών που «διαβάζουμε» διατρέχοντας τα μονοπάτια του διαγράμματος καταστάσεων του A. Αναλυτική απόδειξη υπάρχει στο [6]. □

Πρόταση 2.3.2. *Αν $L = \mathscr{L}(\alpha)$ για μία κανονική έκφραση α, τότε υπάρχει NFA A, τέτοιο ώστε $L = L(A)$.*

Ιδέα απόδειξης. Με επαγωγή στην πολυπλοκότητα της κανονικής έκφρασης α. Αναλυτικά στο [6]. □

Μάλιστα είναι χαρακτηριστικό το γεγονός ότι έχει αναπτυχθεί πλήθος από συστήματα γραφής μεταγλωττιστών, που μετατρέπουν αυτόματα κανονικές εκφράσεις σε πεπερασμένα αυτόματα, ώστε να χρησιμοποιηθούν ως λεκτικοί αναλυτές.

Θεώρημα 2.3.1. *Μία γλώσσα είναι κανονική αν και μόνο αν γίνεται δεκτή από ένα πεπερασμένο αυτόματο.*

Απόδειξη. Άμεση από τις παραπάνω προτάσεις, λόγω των Θεωρημάτων 2.1.1 και 2.2.1. □

Κεφάλαιο 3

Γλώσσες χωρίς συμφραζόμενα

Στο προηγούμενο κεφάλαιο εισηγάγαμε δύο διαφορετικές, αν και ισοδύναμες, μεθόδους περιγραφής γλωσσών: τις κανονικές εκφράσεις και τα πεπερασμένα αυτόματα. Είδαμε ότι πολλές γλώσσες μπορούν να περιγραφούν με αυτούς τους τρόπους. Κάποιες άλλες όμως δεν μπορούν.

Στο παρόν κεφάλαιο θα μελετήσουμε τις γραμματικές χωρίς συμφραζόμενα, μία πολύ πιο ισχυρή μέθοδο περιγραφής γλωσσών. Οι γραμματικές αυτές, οι γραμματικές τύπου 2 της ιεραρχίας του Chomsky, παράγουν την κλάση γλωσσών $\mathcal{L}_2$, που περιέχει την κλάση των κανονικών γλωσσών $\mathcal{L}_3$, και είναι μάλλον η πιο ενδιαφέρουσα κλάση ως προς τις εφαρμογές. Όπως και προηγουμένως, θα ορίσουμε και μία πολύ χρήσιμη κλάση μηχανών που δέχονται ακριβώς αυτές τις γλώσσες, την κλάση των *αυτομάτων στοίβας*.

3.1 Σημασία και κανονικές μορφές των γραμματικών χωρίς συμφραζόμενα

Οι γραμματικές χωρίς συμφραζόμενα (CFGs) χρησιμοποιήθηκαν πρώτα από τους γλωσσολόγους για τη μελέτη των φυσικών γλωσσών, καθώς αιχμαλωτίζουν και μπορούν να αποδώσουν κάποια από τα σημαντικά στοιχεία αναδρομής που εμφανίζονται στην κατασκευή των προτάσεων μίας φυσικής γλώσσας. Ωστόσο, για διάφορους λόγους οι γραμματικές αυτές δε θεωρούνται γενικά επαρκές μοντέλο για την περιγραφή των ανθρωπίνων γλωσσών, αφού για κάτι τέτοιο είναι απαραίτητη τουλάχιστον και η ύπαρξη κάποιας *σημασιολογικής* πληροφορίας που θα αποκλείει την παραγωγή συντακτικά σωστών, αλλά χωρίς νόημα προτάσεων. Ο σχεδιασμός τέτοιων μηχανισμών για μεγάλα υποσύνολα φυσικών γλωσσών άλλωστε έχει αποτελέσει ερευνητική πρόκληση για αρκετές δεκαετίες. Η ιδέα, όμως, των CFGs έχει θεωρητική αξία για την

προσπάθεια συζήτησης για την ανθρώπινη γλώσσα και κατέχει κεντρική θέση στην επιστήμη της *υπολογιστικής γλωσσολογίας* (computational linguistics).

Από την άλλη, το πρακτικό όφελος των CFGs για τις γλώσσες προγραμματισμού είναι τεράστιο, καθώς οι τυπικές περιγραφές των γλωσσών προγραμματισμού μέσω CFGs αντικατέστησαν τις παλιές μακροσκελείς, και πολλές φορές ελλιπείς, ή και διφορούμενες περιγραφές. Αλλά και στη χρήση της XML (eXtensible Markup Language), της γλώσσας που περιγράφει τον τρόπο αποθήκευσης των δεδομένων σε ένα ηλεκτρονικό έντυπο και χρησιμοποιείται ευρέως στο ηλεκτρονικό εμπόριο (e-commerce), διαδραματίζουν οι CFGs ζωτικό ρόλο.

Πέρα απ' αυτά, οι γραμματικές χωρίς συμφραζόμενα είναι μεγάλης πρακτικής σπουδαιότητας για την τεχνολογία μεταγλωττιστών (compiler technology), όπου η υιοθέτησή τους τη δεκαετία του 1960 μετέτρεψε την κατασκευή συντακτικών αναλυτών (parsers) από περίπλοκη, χρονοβόρα διαδικασία με συχνά ανεπαρκές αποτέλεσμα, σε απλή δουλειά ρουτίνας που δεν καταλαμβάνει παρά ένα μικρό ποσοστό του συνολικού χρόνου μεταγλώττισης. Για παράδειγμα, οι γραμματικές αυτές είναι χρήσιμες κατά την περιγραφή αριθμητικών εκφράσεων με τυχαίο πλήθος ορθά τοποθετημένων (balanced) παρενθέσεων, ή κατά την περιγραφή δομημένων τμημάτων κώδικα (block structures), όπως αυτών που σχηματίζουν τα ζεύγη «`begin`» – «`end`» ή «`if`» – «`else`», στις γλώσσες προγραμματισμού.

Για καμία από τις παραπάνω λειτουργίες *συντακτικής ανάλυσης* (parsing, syntax analysis) δεν επαρκούν οι κανονικές γραμματικές, οι οποίες, ωστόσο, μέσω αλγορίθμων που βασίζονται στις κανονικές εκφράσεις και τα πεπερασμένα αυτόματα, εμπλέκονται, όπως είδαμε, στο στάδιο της *λεκτικής ανάλυσης* (lexical analysis). Σημειώνουμε ότι το στάδιο αυτό προηγείται εκείνου της συντακτικής ανάλυσης και συνίσταται στον εντοπισμό δεσμευμένων λέξεων της γλώσσας, ονομάτων μεταβλητών (identifiers) ή και αριθμητικών σταθερών.

Συχνά είναι βολικό κατά την εργασία με μία CFG, οι κανόνες της να βρίσκονται σε όσο το δυνατόν απλούστερη μορφή, ενώ, φυσικά, η γλώσσα που παράγει να παραμένει η ίδια. Μία από τις απλούστερες και χρησιμότερες τέτοιες μορφές είναι η κανονική μορφή Chomsky, που εισήχθη από το N. Chomsky το 1959, και, παρ' όλο που ούτε αυτή φάνηκε ιδιαίτερα αξιόλογη για τη γλωσσολογία, έχει βρει σημαντικές άλλες χρήσεις, όπως έναν πολύ αποδοτικό έλεγχο για το εάν μία λέξη μπορεί να παραχθεί από μία δεδομένη CFG. Μία γραμματική χωρίς συμφραζόμενα $G = (V, T, P, S)$ είναι σε **κανονική μορφή Chomsky** (Chomsky normal form), αν κάθε κανόνας στο P είναι σε μία από τις εξής δύο απλές μορφές:

- $A \to BC$, όπου $A, B, C \in V$, ή
- $A \to a$, όπου $A \in V$ και $a \in T$.

Για να φέρουμε την G σε κανονική μορφή Chomsky, εκτελούμε πρώτα μία σειρά από μετασχηματισμούς, κάθε ένας από τους οποίους απλοποιεί τη μορφή της γραμματικής και έχει ξεχωριστή χρησιμότητα. Συγκεκριμένα, απαλείφουμε:

1. τους *ε-κανόνες* (ε-productions), δηλαδή τους κανόνες της μορφής $A \to \varepsilon$, $A \in V$,
2. τους *μοναδιαίους κανόνες* (unit productions), δηλαδή τους κανόνες της μορφής $A \to B$, $A, B \in V$,
3. τα *άχρηστα σύμβολα* (useless symbols), δηλαδή τα σύμβολα του $V \cup T$ που δεν εμφανίζονται σε καμία παραγωγή συμβολοσειράς τερματικών από το S.

Αποδεικνύεται ότι κάθε μη τετριμμένη CFG μπορεί να μετασχηματιστεί σε CFG σε κανονική μορφή Chomsky. Αυτό παρέχει ένα πολύ σημαντικό πλεονέκτημα, διότι η κανονική μορφή Chomsky επιτρέπει σε έναν απλό πολυωνυμικό αλγόριθμο δυναμικού προγραμματισμού να αποφασίζει, δεδομένης μίας CFG G και μίας συμβολοσειράς w, αν $w \in L(G)$.

Θεώρημα 3.1.1. *Αν η γραμματική χωρίς συμφραζόμενα G παράγει τουλάχιστον μία λέξη εκτός του ε, τότε υπάρχει γραμματική χωρίς συμφραζόμενα G' σε κανονική μορφή Chomsky, τέτοια ώστε $L(G') = L(G) - \{\varepsilon\}$.*

Ιδέα απόδειξης. Αρχικά δείχνεται ότι για κάθε CFG G που παράγει τουλάχιστον μία λέξη εκτός του ε, υπάρχει μία CFG G'' τέτοια ώστε να μην έχει κανόνες της μορφής 1. και 2. ή άχρηστα σύμβολα, και $L(G'') = L(G) - \{\varepsilon\}$. Από το σημείο αυτό είναι πολύ εύκολο πια να έρθει η γραμματική σε κανονική μορφή Chomsky και γίνεται μέσω της εισαγωγής μιας σειράς νέων μεταβλητών που δε διαδραματίζουν ουσιαστικό ρόλο και απλά βοηθούν να ικανοποιηθούν οι συνθήκες της κανονικής μορφής. Λεπτομέρειες μπορεί να βρει κανείς στο [6]. □

Μία άλλη ενδιαφέρουσα κανονική μορφή γραμματικών χωρίς συμφραζόμενα εισήχθη το 1965 από τη S. Greibach και συγγενεύει κάπως με τη μορφή των κανονικών γραμματικών, με την ουσιαστική διαφορά ότι στην προκειμένη περίπτωση τα δεξιά μέλη των κανόνων ενδέχεται να περιέχουν περισσότερες από μία μεταβλητές. Συγκεκριμένα, μία γραμματική χωρίς συμφραζόμενα

$G = (V, T, P, S)$ είναι σε **κανονική μορφή Greibach** (Greibach normal form), αν κάθε κανόνας στο P είναι της μορφής

$$A \to aX, \quad \text{όπου } A \in V,\ a \in T,\ X \in V^*.$$

Αποδεικνύεται πως κάθε CFG χωρίς το ε στη γλώσσα της είναι ισοδύναμη με μία CFG σε κανονική μορφή Greibach.

Θεώρημα 3.1.2. *Αν η γραμματική χωρίς συμφραζόμενα G δεν παράγει το ε, τότε υπάρχει γραμματική χωρίς συμφραζόμενα G' σε κανονική μορφή Greibach, τέτοια ώστε $L(G') = L(G)$.*

Ιδέα απόδειξης. Η απόδειξη είναι αρκετά τεχνική και βασίζεται σε ορισμένα λήμματα, σύμφωνα με τα οποία μπορούμε να αλλάξουμε ελαφρώς τους κανόνες μιας CFG κατά συγκεκριμένους τρόπους, χωρίς να επηρεαστεί η γλώσσα που παράγει η CFG. Λεπτομέρειες βρίσκονται στο, κλασικό για τη θεωρία τυπικών γλωσσών, [7]. □

Οι γραμματικές σε κανονική μορφή Greibach έχουν το ιδιαίτερο χαρακτηριστικό ότι σε αυτές η παραγωγή μίας συμβολοσειράς τερματικών έχει πάντα ακριβώς τόσα βήματα, όσο και το μήκος της συμβολοσειράς.

3.2 Αυτόματα στοίβας

Στο προηγούμενο κεφάλαιο είδαμε ότι τα πεπερασμένα αυτόματα είναι ισοδύναμα σε παραγωγική ισχύ με τις κανονικές γραμματικές. Εύκολα μπορεί κανείς να μαντέψει όμως ότι μία γλώσσα όπως η $L = \{a^n b^n \mid n \geq 0\}$, που είναι γλώσσα χωρίς συμφραζόμενα, θα απαιτούσε για την αποδοχή της από ένα πεπερασμένο αυτόματο *άπειρο* πλήθος καταστάσεων, προκειμένου το αυτόματο να «θυμάται» πόσα a έχει διαβάσει προτού συναντήσει το πρώτο b της εισόδου. Αυτό υποδεικνύει την ανάγκη να συμβιβαστούμε με την ιδέα ενός άπειρου αυτομάτου. Ωστόσο, επειδή μία απευθείας γενίκευση αυτού του είδους είναι υπερβολικά ευρεία, προς το παρόν θα ασχοληθούμε με ειδικές δομές που εργάζονται με θεωρητικά απεριόριστο όγκο δεδομένων με έναν σχετικά απλό τρόπο.

Ένας εξαιρετικής σημασίας τύπος τέτοιας δομής είναι το αυτόματο στοίβας, το οποίο στην ουσία είναι ένα πεπερασμένο αυτόματο, με το επιπρόσθετο χαρακτηριστικό ότι διαθέτει μία άπειρης χωρητικότητας *στοίβα* (stack, store), χάρη στην οποία μπορεί να αποθηκεύει—και συνεπώς να «θυμάται»—πληροφορία άπειρου μεγέθους. Με έναν, όμως, καθόλου ασήμαντο περιορισμό: Το αυτόματο έχει πρόσβαση στη στοίβα κατά έναν *last-in, first-out* τρόπο,

καθώς αποθηκεύει τα νέα δεδομένα πάντα στην κορυφή της στοίβας και μπορεί να διαβάσει μόνο ό,τι αποθηκεύτηκε τελευταία. Για να γίνουμε πιο σαφείς, ένα **αυτόματο στοίβας** (pushdown automaton PDA) είναι μία διατεταγμένη επτάδα $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, όπου

- Q είναι ένα μη κενό, πεπερασμένο σύνολο **καταστάσεων**,
- Σ είναι ένα αλφάβητο (το **αλφάβητο εισόδου**),
- Γ είναι ένα αλφάβητο (το **αλφάβητο στοίβας**),
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \ \rightarrow \ \mathcal{P}(Q \times \Gamma^*)$ είναι η **συνάρτηση μετάβασης**,
- $q_0 \in Q$ είναι η **αρχική κατάσταση**,
- $Z_0 \in \Gamma$ είναι το **αρχικό σύμβολο στοίβας** (το μόνο σύμβολο που βρίσκεται στα περιεχόμενα της στοίβας όταν ξεκινά ένας υπολογισμός),
- $F \subseteq Q$ είναι το σύνολο των **τελικών καταστάσεων**.

Το αυτόματο στοίβας λαμβάνει λοιπόν υπόψη για τη μετάβασή του κάθε φορά όχι μόνο την τρέχουσα κατάσταση και το σύμβολο της εισόδου που διαβάζει, αλλά και το σύμβολο που βρίσκεται στην κορυφή της στοίβας.[1] Καθώς μεταβαίνει στο επόμενο στάδιο καταναλώνοντας είσοδο, μπορεί παράλληλα να αντικαταστήσει το σύμβολο της κορυφής της στοίβας με κάποια συμβολοσειρά· έτσι, αν αντικαταστήσει το X με το γX, όπου $X \in \Gamma, \gamma \in \Gamma^*$, στην πράξη **εισάγει** (pushes) στη στοίβα το γ, ενώ αν αντικαταστήσει το X, $X \in \Gamma$, με το ε, στην πράξη **εξάγει** (pops) το X από τη στοίβα. Προφανώς μπορεί και να αντικαταστήσει κάποιο σύμβολο με τον εαυτό του, αφήνοντας τη στοίβα ανέπαφη.

Για τη συνολική κατάσταση ενός PDA θα πρέπει φυσικά να συνυπολογιστεί και η παράμετρος της στοίβας. Έτσι, η **συνολική κατάσταση** του PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ είναι ένα στοιχείο του $Q \times \Sigma^* \times \Gamma^*$. Αν (q, w, γ), (q', w', γ') είναι δύο συνολικές καταστάσεις, λέμε ότι η (q, w, γ) **παράγει σε ένα βήμα** τη (q', w', γ'),

$$(q, w, \gamma) \vdash_P (q', w', \gamma'),$$

αν $w' = aw$, $\gamma = X\beta$, $\gamma' = \alpha\beta$ για κάποια $a \in \Sigma$, $X \in \Gamma$, $\alpha, \beta \in \Gamma^*$ και $(q', \alpha) \in \delta(q, a, X)$. Ως συνήθως, συμβολίζουμε την ανακλαστική, μεταβατική κλειστότητα της $\vdash_P$ με $\vdash_P^*$.

[1] Κατά σύμβαση, θεωρούμε ότι αν $\gamma \in \Gamma^*$ είναι το περιεχόμενο της στοίβας, το σύμβολο που βρίσκεται στην κορυφή της είναι εκείνο στο αριστερό άκρο του γ, ενώ το σύμβολο του πάτου της είναι εκείνο στο δεξί.

Αποδοχή και γλώσσα

Υπάρχουν διάφορες προσεγγίσεις σχετικά με το υπό ποιες συνθήκες θα θεωρούμε ότι ένα PDA δέχεται μία λέξη. Παραμένοντας στο πνεύμα της αποδοχής από ένα πεπερασμένο αυτόματο, θα λέμε ότι η συμβολοσειρά $w \in \Sigma^*$ **γίνεται δεκτή σε τελική κατάσταση** (is accepted by final state) από το P, αν υπάρχουν $q \in F,\ \alpha \in \Gamma^*$, έτσι ώστε

$$(q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \alpha).$$

Εναλλακτικά, δίνοντας έμφαση στη στοίβα και όχι στις καταστάσεις, θα λέμε ότι η συμβολοσειρά $w \in \Sigma^*$ **γίνεται δεκτή με κενή στοίβα** (is accepted by empty stack) από το P, αν για τυχούσα κατάσταση $q \in Q$ έχουμε

$$(q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \varepsilon).$$

Αντίστοιχα, **η γλώσσα που γίνεται δεκτή σε τελική κατάσταση** από το P είναι η

$$L(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \alpha) \ \text{ για κάποια } q \in F,\ \alpha \in \Gamma^*\},$$

ενώ **η γλώσσα που γίνεται δεκτή με κενή στοίβα** από το P είναι η

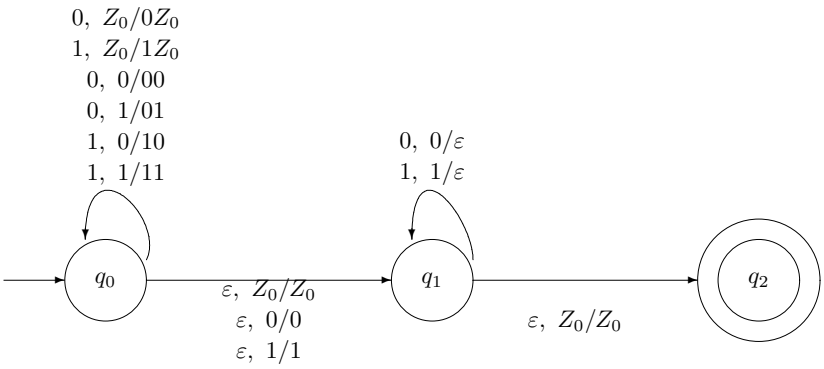
$$N(P) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \varepsilon) \text{ για κάποιο } q \in Q\}.$$

Οι δύο αυτές μέθοδοι είναι ισοδύναμες, με την έννοια ότι μία γλώσσα γίνεται δεκτή σε τελική κατάσταση από κάποιο PDA αν και μόνο αν γίνεται δεκτή με κενή στοίβα από κάποιο (όχι απαραίτητα το ίδιο) PDA.

Θεώρημα 3.2.1. *$L = L(P_F)$ για κάποιο PDA P_F αν και μόνο αν $L = N(P_N)$ για κάποιο PDA P_N.*

Ιδέα απόδειξης. Για το ευθύ, ξεκινώντας από ένα PDA P_F, κατασκευάζουμε ένα PDA P_N που λειτουργεί κατά βάση προσομοιώνοντας τη λειτουργία του P_F, με τη διαφορά ότι αδειάζει τη στοίβα του όταν το P_F βρεθεί σε τελική κατάσταση διαβάζοντας μια λέξη. Για το αντίστροφο, ξεκινώντας από ένα PDA P_N κατασκευάζουμε από αυτό ένα PDA P_F, το οποίο προσομοιώνει το P_N και οδηγείται σε τελική κατάσταση μόλις η στοίβα του P_N αδειάσει. Καμία κατεύθυνση δεν παρουσιάζει ιδιαίτερες δυσκολίες, αλλά έχουν κάποιες τεχνικές λεπτομέρειες που υπάρχουν στο [6]. □

Ακόμα, κάποιες φορές απαιτούμε η αποδοχή να γίνεται **σε τελική κατάσταση και με κενή στοίβα**. Και πάλι, όμως, προκύπτει ότι η προσέγγιση αυτή είναι



Σχήμα 3.1: Γενικευμένο διάγραμμα καταστάσεων που αναπαριστά το PDA του Παραδείγματος 3.2.1.

ισοδύναμη με τις υπόλοιπες. Έτσι, χρησιμοποιούμε τους διάφορους τρόπους αποδοχής κατά βούληση και μιλάμε για **τη γλώσσα** του P. Εντούτοις, δεδομένου ενός PDA, οι γλώσσες που αυτό δέχεται σε τελική κατάσταση και οι γλώσσες που δέχεται με κενή στοίβα είναι συνήθως διαφορετικές.

Παράδειγμα 3.2.1. Το PDA P με το γενικευμένο διάγραμμα καταστάσεων του Σχήματος 3.1 δέχεται τη γλώσσα $L_{wwR} = \{ww^R \mid w \in \{0,1\}^*\}$.

Για να απεικονίσουμε τις μεταβολές στη στοίβα χρησιμοποιούμε τον εξής συμβολισμό: Η ετικέτα $a, X/\alpha$ πάνω από το βέλος με αρχή την κατάσταση p και τέλος την κατάσταση q υποδηλώνει ότι $(p, \alpha) \in \delta(q, a, X)$.

3.3 Αυτόματα στοίβας και γραμματικές χωρίς συμφραζόμενα

Όπως ήδη έχουμε αναφέρει, ένα αυτόματο στοίβας είναι ακριβώς ό,τι χρειάζεται για την αποδοχή οποιασδήποτε γλώσσας χωρίς συμφραζόμενα. Το γεγονός αυτό είναι μεγάλης σημασίας, και μαθηματικής, αλλά και πρακτικής: μαθηματικής, διότι, όπως και με τα πεπερασμένα αυτόματα, συνδυάζονται διαφορετικές αυστηρές προσεγγίσεις της ίδιας κλάσης γλωσσών, προσφέροντας έτσι τη δυνατότητα για περισσότερη εποπτεία· και πρακτικής, διότι παρέχονται τα θεμέλια για την κατασκευή συντακτικών αναλυτών για «πραγματικές» γλώσσες χωρίς συμφραζόμενα, όπως είναι οι γλώσσες προγραμματισμού.

Για να δείξουμε το παραπάνω αποτέλεσμα, αφενός κατασκευάζουμε, δεδομένης μίας CFG, ένα PDA που δέχεται με κενή στοίβα ακριβώς τη γλώσσα

που παράγει η γραμματική.

Πρόταση 3.3.1. *Αν L=L(G) για μία CFG G, τότε υπάρχει PDA P, τέτοιο ώστε L=N(P).*

Ιδέα απόδειξης. Η ιδέα είναι να κατασκευάσουμε το PDA έτσι, ώστε να προσομοιώνει την ακολουθία των αριστερότερων παραγωγών που χρησιμοποιεί η γραμματική για να παραγάγει μία λέξη. Για να το πετύχει αυτό, το PDA χρησιμοποιεί με δραστικό τρόπο τη στοίβα του, ενώ οι καταστάσεις δεν παίζουν κανένα ρόλο. Με λίγα λόγια, το PDA ξεκινά με το αρχικό σύμβολο της γραμματικής στη στοίβα του και σε κάθε βήμα, είτε αντικαθιστά το σύμβολο της κορυφής της στοίβας, σε περίπτωση που αυτό είναι μια μεταβλητή A, με το δεξί μέλος κάποιου κανόνα $A \to \beta$, είτε, σε περίπτωση που είναι τερματικό και συμπίπτει με το τρέχον σύμβολο της εισόδου, το εξάγει από τη στοίβα, προκειμένου να αποκαλυφθεί η επόμενη αριστερότερη μεταβλητή και να συνεχιστεί η διαδικασία. Λεπτομέρειες στο [6]. □

Αντίστροφα, δεδομένου ενός PDA, κατασκευάζουμε μία CFG που παράγει ακριβώς τη γλώσσα που δέχεται το αυτόματο με κενή στοίβα.

Πρόταση 3.3.2. *Αν L=N(P) για ένα PDA P, τότε υπάρχει CFG G, τέτοια ώστε L=L(G).*

Ιδέα απόδειξης. Αυτή η κατασκευή βασίζεται στη διαπίστωση ότι κατά τους υπολογισμούς ενός PDA, αυτό που στην ουσία μετρά είναι η εισαγωγή και εξαγωγή συμβόλων από τη στοίβα. Έτσι, η γραμματική φτιάχνεται με τρόπο ώστε οι μεταβλητές της να παράγουν τις λέξεις εκείνες, με τις οποίες τροφοδοτούμενο το αυτόματο στοίβας θα κάνει έναν υπολογισμό καταλήγοντας να εξάγει ένα σύμβολο από τη στοίβα. Λεπτομέρειες στο [6]. □

Έτσι απορρέει άμεσα το Θεώρημα ισοδυναμίας γραμματικών χωρίς συμφραζόμενα και αυτομάτων στοίβας.

Θεώρημα 3.3.1. *Μία γλώσσα είναι γλώσσα χωρίς συμφραζόμενα αν και μόνο αν γίνεται δεκτή από ένα αυτόματο στοίβας.*

Απόδειξη. Άμεση από τις Προτάσεις 3.3.1 και 3.3.2, σε συνδυασμό με το Θεώρημα 3.2.1. □

3.4 Αυτόματα στοίβας και ντετερμινισμός

Αν και ορίσαμε το αυτόματο στοίβας ως μία μη ντετερμινιστική μηχανή, ιδιαίτερου ενδιαφέροντος είναι η ντετερμινιστική περίπτωση. Αυτό συμβαίνει λόγω

της ανάγκης να βρεθούν αποδοτικοί συντακτικοί αναλυτές για τους μεταγλωττιστές των γλωσσών προγραμματισμού, δηλαδή αλγόριθμοι που αποφασίζουν αν μία λέξη είναι στη γλώσσα που παράγει μία δεδομένη γραμματική χωρίς συμφραζόμενα, και, αν ναι, κατασκευάζουν το συντακτικό της δέντρο. Τέτοιου είδους αλγόριθμοι υπάρχουν διάφοροι, όπως ο αλγόριθμος δυναμικού προγραμματισμού στον οποίο αναφερθήκαμε στην Ενότητα 3.1, αλλά και αρκετές παραλλαγές του. Όμως οι αλγόριθμοι αυτοί, αν και πολυωνυμικοί, είναι υπερβολικά αργοί για να χειριστούν προγράμματα με δεκάδες χιλιάδες εντολών, καθώς ο πιο αποδοτικός από αυτούς τρέχει σε χρόνο ανάλογο προς το χρόνο που απαιτείται για τον πολλαπλασιασμό δύο $n \times n$ πινάκων.[2] Έτσι, πλήθος εναλλακτικών προσεγγίσεων στο πρόβλημα της συντακτικής ανάλυσης έχει αναπτυχθεί τις τελευταίες τέσσερις δεκαετίες από σχεδιαστές μεταγλωττιστών. Οι πιο επιτυχημένες από αυτές συμβαίνει να έχουν τη ρίζα τους στην ιδέα του αυτομάτου στοίβας.

Παρ' όλα αυτά, ένα αυτόματο στοίβας δεν είναι κατάλληλο για άμεση χρήση στην πρακτική της συντακτικής ανάλυσης, αφού λειτουργεί μη ντετερμινιστικά. Το κρίσιμο, λοιπόν, ερώτημα, είναι *εάν μπορούμε πάντα να κάνουμε τα αυτόματα στοίβας να λειτουργούν ντετερμινιστικά*, εάν, δηλαδή, ο μη ντετερμινισμός στην περίπτωση των αυτομάτων στοίβας δεν προσφέρει από πλευράς υπολογισιμότητας επιπλέον ισχύ, όπως συνέβη με τα πεπερασμένα αυτομάτα. Δυστυχώς η απάντηση αυτή τη φορά είναι *αρνητική*.

Υπάρχουν γλώσσες χωρίς συμφραζόμενα που δε γίνονται δεκτές από κανένα ντετερμινιστικό αυτόματο στοίβας. Για την ακρίβεια, η κλάση των γλωσσών που γίνονται δεκτές από ντετερμινιστικά αυτόματα στοίβας περιέχει γνήσια όλες τις κανονικές γλώσσες (αφού ένα ντετερμινιστικό πεπερασμένο αυτόματο προσομοιώνεται τετριμμένα από ένα ντετερμινιστικό αυτόματο στοίβας που απλά αγνοεί τη στοίβα του), αλλά περιέχεται γνήσια στην κλάση των γλωσσών χωρίς συμφραζόμενα. Μάλιστα, το πρόβλημα εάν μία τυχαία γλώσσα χωρίς συμφραζόμενα γίνεται δεκτή από κάποιο ντετερμινιστικό αυτόματο στοίβας είναι αναποκρίσιμο. Η θετική, όμως, όψη του ζητήματος είναι ότι τουλάχιστον για τις περισσότερες γλώσσες προγραμματισμού προκύπτει πάντοτε εφικτή η κατασκευή ντετερμινιστικών αυτομάτων στοίβας που δέχονται όλα τα συντακτικά ορθά προγράμματα.

Διαισθητικά, ένα αυτόματο στοίβας είναι ντετερμινιστικό εάν δεν έχει ποτέ δυνατότητα επιλογής της επόμενης κίνησής του, εάν, δηλαδή, για κάθε συνολική κατάσταση υπάρχει το πολύ μία συνολική κατάσταση που μπορεί να τη διαδεχτεί. Έτσι, ονομάζουμε το PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ **ντετερμινιστικό αυτόματο στοίβας** DPDA, αν για κάθε $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$,

[2] Τελευταία ο χρόνος αυτός είναι της τάξης του $\mathcal{O}(n^{2.3\cdots})$.

η συνάρτηση μετάβασής του πληροί τις εξής δύο συνθήκες:

1. $|\,\delta(q,a,X)| \leq 1$.
2. Αν $|\,\delta(q,a,X)| = 1$, τότε $|\,\delta(q,\varepsilon,X)| = 0$.

Λέμε ότι η γλώσσα L έχει την **ιδιότητα προθέματος** (prefix property) αν *δεν* υπάρχουν δύο διαφορετικές λέξεις στη γλώσσα, τέτοιες ώστε η μία να είναι πρόθεμα της άλλης. Αποδεικνύεται πως αν μία γλώσσα έχει την ιδιότητα προθέματος και γίνεται δεκτή από ένα DPDA σε τελική κατάσταση, τότε και μόνο τότε γίνεται δεκτή από ένα DPDA με κενή στοίβα. Προκειμένου οι έννοιες της αποδοχής να είναι ισοδύναμες και στην περίπτωση των DPDAs, χρησιμοποιούμε ένα μικρό τέχνασμα: Για να εξετάσουμε αν μια γλώσσα γίνεται δεκτή από ένα DPDA, πρώτα τη μετατρέπουμε ελαφρώς, προσδίδοντάς της την ιδιότητα προθέματος. Συγκεκριμένα, επικολλούμε στο τέλος κάθε λέξης της $L \subseteq \Sigma^*$ ένα καινούριο, ειδικό σύμβολο $\$ \notin \Sigma$ και δημιουργούμε τη νέα γλώσσα $L\$ = \{w\$ \mid w \in L\}$, της οποίας προφανώς καμία λέξη δεν είναι πρόθεμα άλλης. Έτσι, μπορούμε να λέμε ότι η L είναι μία **ντετερμινιστική γλώσσα χωρίς συμφραζόμενα**, αν η $L\$$ είναι γλώσσα κάποιου DPDA.

Αξιοσημείωτο είναι το γεγονός ότι κάθε ντετερμινιστική γλώσσα χωρίς συμφραζόμενα έχει μια μη διφορούμενη γραμματική που την παράγει. Δηλαδή καμία ντετερμινιστική γλώσσα χωρίς συμφραζόμενα δεν είναι εγγενώς διφορούμενη, στοιχείο απαραίτητο για τη συντακτική ανάλυση. Φυσικά δεν ισχύει το ίδιο για όλες τις γλώσσες χωρίς συμφραζόμενα, γεγονός που, και αυτό, δείχνει το βασικό αποτέλεσμα διαχωρισμού των γλωσσών χωρίς συμφραζόμενα.

Θεώρημα 3.4.1. *Η κλάση των ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα είναι γνήσια υποκλάση της $\mathcal{L}_2$.*

Απόδειξη. Η γλώσσα L_{wwR}, που γίνεται δεκτή από το PDA του Παραδείγματος 3.2.1, δεν μπορεί να γίνει δεκτή από κανένα DPDA. Το επιχείρημα γίνεται εύκολα αντιληπτό: Ένα DPDA που έχει ήδη διαβάσει μία συμβολοσειρά της μορφής 0^n110^n θα πρέπει αναγκαστικά να έχει αδειάσει τη στοίβα του, με αποτέλεσμα να μη «θυμάται» πια τον αριθμό n και να μην μπορεί να δεχτεί τη $0^n110^{2n}110^n$, που ανήκει στη γλώσσα. □

3.5 Συντακτική ανάλυση

Έχοντας ξεκαθαρίσει ότι δεν μπορούν όλες οι γλώσσες χωρίς συμφραζόμενα να γίνουν δεκτές από ντετερμινιστικά αυτόματα στοίβας, κι έτσι δεν έχουν

όλες γρήγορους αλγορίθμους συντακτικής ανάλυσης, στρεφόμαστε τώρα στις χρήσιμες κατηγορίες γλωσσών, που έχουν. Οι πιο διαδεδομένοι αλγόριθμοι συντακτικής ανάλυσης για τις γλώσσες αυτές βασίζονται στις εξής δύο μεθόδους: τη μέθοδο συντακτικής ανάλυσης *από την κορυφή προς τον πάτο* (top-down) και εκείνη *από τον πάτο προς την κορυφή* (bottom-up), όπου, βέβαια, αναφερόμαστε στον τρόπο κατασκευής του συντακτικού δέντρου.

Το ζητούμενο είναι, καθώς διαβάζεται η είσοδος (η συμβολοσειρά που αναλύεται), να εντοπίζεται σε κάθε βήμα ο σωστός κανόνας της γραμματικής (ο κανόνας, δηλαδή, που οδήγησε στην παραγωγή της συγκεκριμένης συμβολοσειράς)—πράγμα απαραίτητο, εφόσον δουλεύουμε ντετερμινιστικά και δεν μπορούμε να «μαντέψουμε». Πώς, όμως, είναι κάτι τέτοιο εφικτό, όταν πιθανόν περισσότεροι από ένας κανόνας είναι εφαρμόσιμοι, και η λέξη ακόμα δεν έχει διαβαστεί εξ ολοκλήρου; *Γενικά δεν είναι.* Συχνά όμως συμβαίνει, μία σύντομη ματιά σε περιορισμένο πλήθος συμβόλων από το κομμάτι της εισόδου που δεν έχει ακόμα διαβαστεί να είναι αρκετή για να ληφθούν σωστές αποφάσεις σε κάθε σημείο της διαδικασίας. Σε αυτό συνίσταται η ιδέα της **πρόβλεψης** (lookahead), σύμφωνα με την οποία, προκειμένου κατά τη συντακτική ανάλυση να αποφασίσουμε για το σωστό κανόνα της γραμματικής, βασιζόμαστε στη θεώρηση περιορισμένου, σταθερού μήκους κάθε φορά, προθέματος του τμήματος της εισόδου που δεν έχει ακόμα διαβαστεί.

Έτσι φτάνουμε σε δύο βασικές προσεγγίσεις του προβλήματος της συντακτικής ανάλυσης: στην LL(k) συντακτική ανάλυση, κατά την οποία η είσοδος διαβάζεται από αριστερά προς τα δεξιά (Left-to-right), η κατασκευή του συντακτικού δέντρου αναπαριστά μία *αριστερότερη* (Leftmost) παραγωγή και χρησιμοποιείται πρόβλεψη μήκους k, και, στη δυϊκή της έννοια, την LR(k) συντακτική ανάλυση, κατά την οποία η είσοδος διαβάζεται πάλι από αριστερά προς τα δεξιά (Left-to-right), αλλά η κατασκευή του συντακτικού δέντρου εδώ αναπαριστά μία *δεξιότερη* (Rightmost) παραγωγή σε αντίστροφη σειρά, ενώ η πρόβλεψη που χρησιμοποιείται είναι μήκους k. Η πρώτη συνιστά μέθοδο ανάλυσης από την κορυφή προς τον πάτο, ενώ η δεύτερη από τον πάτο προς την κορυφή.

3.5.1 Η συντακτική ανάλυση LL(k)

Έστω η γραμματική χωρίς συμφραζόμενα $G = (V, T, P, S)$ και η λέξη $w \in T^*$, την οποία αναλύουμε. Ας υποθέσουμε ότι έχουμε ήδη βρει ένα αρχικό τμήμα κάποιας αριστερότερης παραγωγής της w,

$$S \Rightarrow_{lm}^{*} xA\alpha,$$

όπου $w = xy,$ για κάποια $x, y \in T^*,\ A \in V,\ \alpha \in (V \cup T)^*$. Το επόμενο βήμα στην αριστερότερη παραγωγή οφείλει να είναι η αντικατάσταση της μεταβλητής A με το δεξιό μέλος β κάποιου κανόνα

$$A \to \beta \in P.$$

Σε περίπτωση που στο P υπάρχουν περισσότεροι από ένας κανόνες με το A στο αριστερό μέλος, θα πρέπει να γίνει μία επιλογή ανάμεσά τους, με κριτήριο την πληροφορία που βρίσκεται όχι μόνο στο x, το αρχικό κομμάτι του w, αλλά πιθανόν και στο υπόλοιπο, το y. Αν τώρα σε κάθε τέτοια στιγμή είναι δυνατό να ληφθεί σωστή απόφαση εξετάζοντας τα πρώτα το πολύ k σύμβολα από το y, η G ονομάζεται γραμματική LL(k).

Προτού δώσουμε έναν αυστηρό, ακριβή ορισμό για αυτή την έννοια, έχει ενδιαφέρον να παρατηρήσουμε ότι στην περίπτωση που η συγκεκριμένη επιλογή μπορεί να γίνεται πάντα χωρίς καθόλου επιθεώρηση του y, δηλαδή με 0 σύμβολα πρόβλεψη, έχουμε να κάνουμε με μία LL(0) γραμματική. Οι LL(0) γραμματικές δεν είναι και τόσο αξιόλογες, εφόσον, όπως είναι προφανές, παράγουν το πολύ μία λέξη. Αντίθετα, οι LL(1) γραμματικές έχουν αποδειχθεί ιδιαίτερα χρήσιμες, καθώς χρησιμοποιούνται με επιτυχία για την κατασκευή μεταγλωττιστών για πολλές γλώσσες προγραμματισμού. Εδώ άλλωστε βρίσκει και μία εξαιρετική εφαρμογή η κανονική μορφή Greibach, που περιγράψαμε στην Ενότητα 3.1, αφού όταν μία γραμματική χωρίς συμφραζόμενα είναι σε αυτή τη μορφή, είναι άμεσο να ελέγξουμε εάν είναι LL(1).

Συμβολίζουμε με $w\restriction k$ το μήκους k αρχικό τμήμα του w,

$$w\restriction k = \begin{cases} x, & \text{αν } w = xy \text{ για κάποια } x, y \text{ με } |x| = k \\ w, & \text{αν } |w| \leq k \end{cases}$$

και ονομάζουμε τη γραμματική χωρίς συμφραζόμενα $G = (V, T, P, S)$, **γραμματική LL(k)** (LL(k) grammar) αν και μόνο αν για κάθε ζεύγος *αριστερότερων* παραγωγών της μορφής

$$\begin{aligned} & S \Rightarrow^* xA\alpha \Rightarrow x\beta\alpha \Rightarrow^* xy \\ \text{και} \quad & S \Rightarrow^* xA\alpha \Rightarrow x\beta'\alpha \Rightarrow^* xz, \end{aligned}$$

όπου $x, y, z \in T^*,\ A \in V$ και $\alpha, \beta, \beta' \in (V \cup T)^*$,

$$\text{αν} \quad y\restriction k = z\restriction k, \quad \text{τότε} \quad \beta = \beta'.$$

Από τον ορισμό συνεπάγεται άμεσα ότι μία γραμματική LL(k) δεν μπορεί να είναι διφορούμενη.

Θεώρημα 3.5.1. *Κάθε LL(k) γραμματική είναι μη διφορούμενη.*

Απόδειξη. Σε μία διφορούμενη γραμματική υπάρχει λέξη με δύο διαφορετικές αριστερότερες παραγωγές, γεγονός που προφανώς παραβιάζει την LL(k) συνθήκη. □

Χάρη στις ειδικές ιδιότητές της, λοιπόν, μπορεί μία LL(k) γραμματική να προσομοιωθεί εύκολα από ένα ντετερμινιστικό αυτόματο στοίβας. Η συγκεκριμένη κατασκευή συντακτικού αναλυτή στηρίζεται στην ίδια ιδέα με εκείνη της Πρότασης 3.3.1 και το αυτόματο που κατασκευάζεται χρησιμοποιεί με τον ίδιο τρόπο τη στοίβα του για να προσομοιώσει αριστερότερη παραγωγή, μόνο που τώρα, προκειμένου να είναι ντετερμινιστικό, επιλέγει προς αντικατάσταση της μεταβλητής A το δεξί μέλος β του *μοναδικού σωστού* κανόνα της γραμματικής $A \to \beta$. Έτσι, η επιλογή του πλέον δε γίνεται από το σύνολο όλων των κανόνων, αλλά από ένα *σύνολο πρόβλεψης* (lookahead set), το οποίο σε μία LL(k) γραμματική είναι πάντα μονοσύνολο και φυσικά προσδιορίζεται με τη βοήθεια της πρόβλεψης.

Βέβαια, πρέπει να τονιστεί πως όχι μόνο είναι δυνατόν για μία γραμματική χωρίς συμφραζόμενα να μην επαρκεί η πρόβλεψη—οσοδήποτε μεγάλου μήκους—για να βρεθεί μοναδικός σωστός κανόνας σε κάθε περίπτωση, αλλά, επιπλέον, για μία γραμματική που χρησιμοποιεί πρόβλεψη, ενδεχομένως να μην υπάρχει κανένας τρόπος να μειωθεί το μήκος της τελευταίας. Για την ακρίβεια, για κάθε $k \geq 0$ υπάρχουν LL($k+1$) γραμματικές, που δεν είναι ισοδύναμες με καμία LL(k) γραμματική.

Θεώρημα 3.5.2. *Για κάθε $k \geq 0$ η κλάση των γραμματικών LL(k) περιέχεται γνήσια σε εκείνη των LL(k + 1).*

Παραλείπουμε την πλήρη απόδειξη, που έγινε από τον R. Kurki-Suonio το 1969. Αντί για αυτή, θα δώσουμε παράδειγμα γραμματικής που είναι LL(2) και όχι LL(1). Θεωρούμε την

$$G = (\{S\}, \{a, b\}, \{S \to aaSb \,|\, ab \,|\, bb\}, S),$$

που παράγει τη γλώσσα

$$L(G) = \{a^{2m+1}b^{m+1} \mid m \geq 0\} \cup \{a^{2m}b^{m+2} \mid m \geq 0\}.$$

Η G δεν είναι $LL(1)$, αφού

$$S \Rightarrow^* aaaaSbb \Rightarrow aaaaabbb$$
$$S \Rightarrow^* aaaaSbb \Rightarrow aaaaaaSbbb \Rightarrow aaaaaabbbbb$$

και $abbb \restriction 1 = a = aabbbbb \restriction 1$, ενώ $ab \neq aaSb$, που αντιτίθεται στον ορισμό της $LL(1)$. Από την άλλη, η ίδια γραμματική προφανώς είναι $LL(2)$,

αφού η πρόβλεψη μήκους 2 εύκολα υποδεικνύει το σωστό κανόνα: αν έχουμε aa, ο σωστός κανόνας είναι ο $S \to aaSb$, αν έχουμε ab, είναι ο $S \to ab$, και αν έχουμε bb, ο $S \to bb$.

Ακόμα, δεν υπάρχει γενική μέθοδος μετατροπής μίας γραμματικής χωρίς συμφραζόμενα σε ισοδύναμη γραμματική LL(k). Αρκετές φορές, ωστόσο, έχουν αποτέλεσμα κάποιοι απλοί ευρετικοί κανόνες, που χρησιμοποιούνται όταν οι λόγοι που εμποδίζουν τη διαδικασία της πρόβλεψης να ευοδωθεί είναι επιφανειακοί.

Μία τέτοια περίπτωση είναι όταν η γραμματική έχει τουλάχιστον δύο κανόνες με πανομοιότυπο όχι μόνο το αριστερό μέλος, αλλά και το πρώτο κομμάτι του δεξιού, δηλαδή της μορφής

$$A \to \alpha\beta_1 \,, \ldots, \, A \to \alpha\beta_n,$$

με $\alpha \neq \varepsilon$ και $n \geq 2$. Τότε μπορούμε εύκολα να απαλείψουμε τα όμοια κομμάτια από τους κανόνες με την τεχνική της *αριστερής παραγοντοποίησης* (left factoring), σύμφωνα με την οποία αντικαθιστούμε τους παραπάνω κανόνες με τους

$$\begin{gathered} A \to \alpha A' \\ A' \to \beta_1 \,, \ldots, \, A' \to \beta_n, \end{gathered}$$

όπου η A' είναι μία καινούρια μεταβλητή. Η αντικατάσταση αυτή έχει ως αποτέλεσμα την αναβολή της απόφασης ανάμεσα στους αρχικούς κανόνες, ώσπου να συγκεντρωθούν περισσότερες πληροφορίες, ενώ η γλώσσα που παράγεται παραμένει η ίδια. Αποδεικνύεται ότι οι αριστερά παραγοντοποιημένες γραμματικές είναι ακριβώς οι LL(1) γραμματικές.

Μία άλλη ενδιαφέρουσα τέτοια περίπτωση είναι όταν η γραμματική έχει κανόνες, στους οποίους η μεταβλητή του αριστερού μέλους επαναλαμβάνεται ως πρώτο σύμβολο στο δεξί. Το φαινόμενο αυτό λέγεται *αριστερή αναδρομή* (left recursion) και μπορούμε να απαλλάξουμε τη γραμματική από αυτό με μία μικρή επέμβαση. Αν

$$\begin{gathered} A \to A\alpha_1 \,, \ldots, \, A \to A\alpha_n \\ A \to \beta_1 \,, \ldots, \, A \to \beta_m, \end{gathered}$$

είναι όλοι οι κανόνες με τη μεταβλητή A στο αριστερό μέλος (όπου τα β_i, $1 \leq i \leq m$, είναι συμβολοσειρές που δεν ξεκινούν με A), τότε τους αντικαθιστούμε

με τους

$$A \to \beta_1 A', \ldots, A \to \beta_m A'$$
$$A' \to \alpha_1 A', \ldots, A' \to \alpha_n A'$$
$$A' \to \varepsilon,$$

όπου η A' είναι νέα μεταβλητή. Δεν είναι δύσκολο να δειχτεί ότι η γραμματική που προκύπτει παράγει την ίδια γλώσσα με την αρχική, ενώ τώρα οι αμφιβολίες σχετικά με την επιλογή κανόνα ίσως ξεκαθαρίζονται με την πρόβλεψη. Σημειώνουμε ότι η μετατροπή μίας γραμματικής χωρίς συμφραζόμενα σε κανονική μορφή Greibach φαίνεται και εδώ χρήσιμη, καθώς στην πορεία της απαλείφεται και η αριστερή αναδρομή. Παρ' όλα αυτά, η απαλοιφή της αριστερής αναδρομής δεν αρκεί πάντα για να πάρουμε γραμματική $\mathrm{LL}(k)$. Μάλιστα, αποδεικνύεται ότι αν μία γραμματική έχει τέτοιου είδους αριστερή αναδρομή, ώστε να υπάρχει παραγωγή της μορφής

$$A \Rightarrow^+ A\alpha,$$

και επιπλέον είναι *απέριττη* (nonredundant), δηλαδή δεν έχει άχρηστα σύμβολα, τότε δεν είναι $\mathrm{LL}(k)$ για κανένα k.

3.5.2 Η συντακτική ανάλυση $\mathrm{LR}(k)$

Εκτός από την κατασκευή συντακτικού αναλυτή που περιγράψαμε στην προηγούμενη υποενότητα, υπάρχει και ένας ακόμα, αρκετά διαφορετικός, τρόπος κατασκευής αυτομάτου στοίβας που δέχεται τη γλώσσα που παράγει μία γραμματική χωρίς συμφραζόμενα. Το αποτέλεσμα είναι η ανακατασκευή ενός συντακτικού δέντρου από τα φύλλα προς τη ρίζα, και όχι αντίστροφα, όπως συνέβη προηγουμένως, και οι γραμματικές που προσφέρονται για κάτι τέτοιο είναι οι γραμματικές $\mathrm{LR}(k)$.

Μία γραμματική χωρίς συμφραζόμενα $G = (V, T, P, S)$ λέγεται **γραμματική $\mathbf{LR}(k)$** ($\mathrm{LR}(k)$ grammar) αν και μόνο αν το P δεν περιλαμβάνει τον κανόνα $S \to S$ και, επιπλέον, για κάθε ζεύγος *δεξιότερων* παραγωγών της μορφής

$$S \Rightarrow^* \alpha A x \Rightarrow^* \alpha\alpha' x$$
και
$$S \Rightarrow^* \beta B x' \Rightarrow^* \beta\beta' x',$$

όπου $\alpha, \alpha', \beta, \beta' \in (V \cup T)^*$, $A, B \in V$ και $x, x' \in T^*$,

αν $\alpha\alpha' x \restriction (|\alpha\alpha'| + k) = \beta\beta' x' \restriction (|\alpha\alpha'| + k)$,
τότε $\alpha = \beta, \quad A = B, \quad \alpha' = \beta'$.

Η απαίτηση να μην υπάρχει κανόνας $S \to S$ είναι απαραίτητη, ώστε να μη χαρακτηριστούν ως LR(k) κάποιες γραμματικές που πληρούν τις υπόλοιπες συνθήκες, όμως είναι διφορούμενες.[3] Έτσι εξασφαλίζεται και πάλι ότι μία γραμματική LR(k) οπωσδήποτε δεν είναι διφορούμενη.

Θεώρημα 3.5.3. *Κάθε LR(k) γραμματική είναι μη διφορούμενη.*

Απόδειξη. Και πάλι άμεση από τον ορισμό, αφού μία διφορούμενη γραμματική παράγει δύο λέξεις με διαφορετικές δεξιότερες παραγωγές. □

Επίσης, και πάλι σε αντιστοιχία με την περίπτωση των LL(1) γραμματικών, αποδεικνύεται ότι οι γραμματικές LR(k) σχηματίζουν μία γνήσια αύξουσα ιεραρχία.

Θεώρημα 3.5.4. *Για κάθε $k \geq 0$ η κλάση των γραμματικών LR(k) περιέχεται γνήσια σε εκείνη των LR(k + 1).*

Απόδειξη. Η γραμματική $G = (\{S, A\}, \{a, b, c, d\}, \{S \to ab^k c \mid Ab^k d, A \to a\})$ αποτελεί παράδειγμα LR($k + 1$) γραμματικής που δεν είναι LR(k), για κάθε $k \geq 0$. □

Οι LR(k) γραμματικές έχουν πολλές ενδιαφέρουσες ιδιότητες. Για παράδειγμα, δείχνεται ότι κάθε LR(0) γραμματική παράγει μία ντετερμινιστική γλώσσα χωρίς συμφραζόμενα, και κάθε ντετερμινιστική γλώσσα χωρίς συμφραζόμενα που έχει την ιδιότητα προθέματος παράγεται από μία LR(0) γραμματική. Για την απόδειξη παραπέμπουμε στο [7].

Θεώρημα 3.5.5. *Η γλώσσα L παράγεται από μία LR(0) γραμματική αν και μόνο αν είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα με την ιδιότητα προθέματος.*

Από το παραπάνω οδηγούμαστε άμεσα σε έναν ακριβή χαρακτηρισμό των ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα:

Πόρισμα 3.5.1. *Η L είναι μία ντετερμινιστική γλώσσα χωρίς συμφραζόμενα αν και μόνο αν η L\$ παράγεται από μία γραμματική LR(0).*

[3] Μία τέτοια είναι η $S \to a \mid S$.

Απόδειξη. Αν η L είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα, τότε είναι και η $L\$$, καθώς εύκολα κατασκευάζεται ένα DPDA για την $L\$$. Εξάλλου, για οποιαδήποτε γλώσσα L, η $L\$$ έχει την ιδιότητα προθέματος, και από το Θεώρημα 3.5.5 συνάγεται ότι η $L\$$ παράγεται από μία LR(0) γραμματική.

Αντίστροφα, αν η $L\$$ παράγεται από μία LR(0) γραμματική, τότε λόγω του Θεωρήματος 3.5.5, η $L\$$ είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα. Αποδεικνύεται επίσης ότι αν η L_1 είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα και η L_2 κανονική γλώσσα, τότε η L_1/L_2 είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα, όπου

$$L_1/L_2 = \{x \mid \text{υπάρχει } w \in L_2 \text{ τέτοιο ώστε } xw \in L_1\}.$$

Αφού λοιπόν $L = L\$/\{\$\}$, προκύπτει ότι η L είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα. □

Από το Θεώρημα 3.5.5 φαίνεται ότι η συνθήκη των LR(0) γραμματικών είναι μάλλον υπερβολικά περιοριστική ώστε να παραχθούν βολικές, φυσικές γραμματικές για τις γλώσσες προγραμματισμού. Αν όμως προσθέσουμε τη δυνατότητα για πρόβλεψη ακόμα *ενός* συμβόλου, φτάνουμε σε μία κλάση γραμματικών εξαιρετικής σημασίας για το σχεδιασμό μεταγλωττιστών, καθώς παράγουν γλώσσες αρκετά ευρείες, ώστε να περιλαμβάνουν σχεδόν κάθε γλώσσα προγραμματισμού, και ταυτόχρονα αρκετά περιορισμένες, ώστε να αναλύονται συντακτικά με αποδοτικό τρόπο: τις ντετερμινιστικές γλώσσες χωρίς συμφραζόμενα.

Στο ιδιαίτερα περιεκτικό [12]—που, γενικεύοντας μια σειρά από δημοσιεύσεις σχετικές με υποκλάσεις των CFGs που έχουν αποδοτικούς αλγορίθμους συντακτικής ανάλυσης, εισάγει τις LR(k) γραμματικές—δείχνεται ότι η γλώσσα κάθε LR(k) γραμματικής είναι ντετερμινιστική γλώσσα χωρίς συμφραζόμενα, και κάθε ντετερμινιστική γλώσσα χωρίς συμφραζόμενα παράγεται από μία LR(1) γραμματική.

Θεώρημα 3.5.6. *Η L είναι μία ντετερμινιστική γλώσσα χωρίς συμφραζόμενα αν και μόνο αν παράγεται από μία γραμματική LR(1).*

Έτσι, σε ό,τι αφορά τις γραμματικές LR(k), στην ουσία μόνο οι περιπτώσεις $k = 0$ και $k = 1$ είναι σημαντικές.

Φυσικά, προκειμένου να ελέγξουμε αν μία γραμματική είναι LR(k), υπάρχουν πολύ πιο αποδοτικοί τρόποι από τον άμεσο έλεγχο του εάν ικανοποιείται η συνθήκη που δώσαμε στον ορισμό. Κάποιοι από αυτούς, που χρησιμοποιούνται συχνά στο σχεδιασμό μεταγλωττιστών, εκτελούν ταυτόχρονα και την κατασκευή του αντίστοιχου συντακτικού αναλυτή για τη δεδομένη γραμματική, στην περίπτωση που είναι όντως LR(k). Αυτός, βέβαια, δεν είναι παρά

ένα ντετερμινιστικό αυτόματο στοίβας, που, όπως είπαμε στην αρχή της υποενότητας, λειτουργεί με έναν πρωτότυπο τρόπο.

Το αυτόματο στοίβας αυτή τη φορά τοποθετεί στη στοίβα το πρώτο σύμβολο της εισόδου και, σε κάθε βήμα, είτε αντικαθιστά τη συμβολοσειρά β^R της κορυφής της στοίβας με το αριστερό μέλος κάποιου κανόνα της γραμματικής $A \to \beta$, είτε, αν αποφασίσει ότι δεν έχει φτάσει στο κατάλληλο σημείο για να εφαρμόσει (αντίστροφα) κάποιον κανόνα, διαβάζει το επόμενο σύμβολο της εισόδου και το εισάγει στη στοίβα.[4] Με αυτόν τον τρόπο το αυτόματο πετυχαίνει να προσομοιώσει ακριβώς, αλλά σε αντίστροφη σειρά, δηλαδή από το τελευταίο βήμα προς το πρώτο, τις δεξιότερες παραγωγές της γραμματικής.

Παρατηρούμε, βέβαια, ότι σε κάθε στιγμή το αυτόματο έρχεται αντιμέτωπο με πολλές επιλογές. Συγκεκριμένα, πρέπει να αποφασίζει εάν θα εφαρμόσει κάποιον κανόνα ή θα εισάγει στη στοίβα το επόμενο σύμβολο, αλλά και με ποιον τρόπο θα εφαρμόσει κανόνα, δηλαδή πόσα σύμβολα της στοίβας θα αντικαταστήσει και με ποια μεταβλητή. Όλα τα παραπάνω σε μία $\mathrm{LR}(k)$ γραμματική καθορίζονται με τη βοήθεια της πρόβλεψης. Στην πράξη αυτό γίνεται εφικτό με διάφορα μέσα, όπως με τον ορισμό κάποιας *σχέσης προτεραιότητας*, ή ακόμα και με την κατασκευή ενός *πεπερασμένου αυτομάτου*, τα οποία συμβουλευόμενο το αυτόματο στοίβας μπορεί να λειτουργήσει.

Κλείνουμε την ενότητα της συντακτικής ανάλυσης σχολιάζοντας—αφού τονίσουμε ξανά ότι δεν υπάρχει ένας μοναδικός βέλτιστος τρόπος συντακτικής ανάλυσης, αλλά διαφορετικές μέθοδοι είναι κατάλληλες για διαφορετικές γραμματικές—τη σχέση ανάμεσα στις δύο κλάσεις γραμματικών που εξετάσαμε.

Θεώρημα 3.5.7. *Για κάθε $k \geq 0$, η κλάση των γλωσσών που παράγονται από μία γραμματική LL(k) περιέχεται γνήσια στην κλάση εκείνων που παράγονται από μία γραμματική LR(1).*

Το παραπάνω πηγάζει από τις δημοσιεύσεις των Lewis και Steams (1968), και Rosenkrantz και Steams (1970), και, δεδομένου του Θεωρήματος 3.5.6, δηλώνει ότι, σε αντίθεση με τις $\mathrm{LR}(k)$, οι $\mathrm{LL}(k)$ γραμματικές δεν επαρκούν για την παραγωγή όλων των ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα.

[4] Σημειώνουμε ότι το β βρίσκεται αντίστροφα στη στοίβα, επειδή η φορά με την οποία το αυτόματο διαβάζει τα σύμβολα της εισόδου και τα τοποθετεί στη στοίβα είναι από αριστερά προς τα δεξιά.

Κεφάλαιο 4

Η υπόλοιπη ιεραρχία

Ως το σημείο αυτό παρουσιάσαμε τις δύο πιο σημαντικές (και με τους περισσότερους περιορισμούς) κλάσεις γλωσσών της ιεραρχίας Chomsky, καθώς και τη σύνδεσή τους με διάφορα υπολογιστικά μοντέλα. Είδαμε ότι τα πεπερασμένα αυτόματα είναι «καλά» μοντέλα μηχανών με περιορισμένη μνήμη, ενώ τα αυτόματα στοίβας είναι «καλά» μοντέλα μηχανών με απεριόριστη μνήμη που χρησιμοποιείται με τον last-in, first-out τρόπο μίας στοίβας. Καιρός είναι να επισημάνουμε ότι κάποιες πολύ απλές εργασίες είναι πέραν των δυνατοτήτων αυτών των μοντέλων, αφού, στην ουσία, είναι υπερβολικά περιορισμένα ώστε να υπηρετήσουν ως μοντέλα υπολογιστών γενικής χρήσης.

Στο κεφάλαιο αυτό θα εξετάσουμε τις δύο πιο γενικές κλάσεις γλωσσών της ιεραρχίας, τις $\mathcal{L}_0$ και $\mathcal{L}_1$, που όμως είναι πολύ λιγότερο χρήσιμες από πρακτικής πλευράς, καθώς είναι αρκετά δύσκολο να τις χειριστούμε. Πάλι θα δούμε παράλληλα και τα αντίστοιχα υπολογιστικά μοντέλα που σχετίζονται με αυτές, ενώ θα κλείσουμε με μία συγκεντρωτική επισκόπηση της ιεραρχίας.

4.1 Γλώσσες χωρίς περιορισμούς

Ξεκινούμε με την ευρύτερη από τις δύο κλάσεις, που είναι και η κορυφή της ιεραρχίας, την κλάση $\mathcal{L}_0$ των γλωσσών χωρίς περιορισμούς. Αν και, όπως είπαμε, οι γλώσσες αυτές παρουσιάζουν ελάχιστο πρακτικό ενδιαφέρον, η θεωρητική τους αξία είναι αντίστροφα ανάλογη. Αυτό οφείλεται στην αποκάλυψη ότι συνδέονται άμεσα με τις μηχανές Turing, και, κατ' επέκταση, με τις αναδρομικά απαριθμητές γλώσσες.

4.1.1 Μία κανονική μορφή για τις γραμματικές χωρίς περιορισμούς

Όπως την έχουμε ορίσει, μία γραμματική χωρίς περιορισμούς δεν είναι υπό καμία έννοια περισσότερο εξειδικευμένη από μία τυχαία τυπική γραμματική, όπου όλες οι μορφές κανόνων είναι επιτρεπτές, αρκεί πάντα να πληρούται η συνθήκη που απαιτεί το αριστερό μέλος να περιλαμβάνει μια μεταβλητή. Δείχνεται, όμως, ότι ακόμα και μια γραμματική χωρίς περιορισμούς μπορεί να έρθει σε ένα είδος κανονικής μορφής. Η μορφή αυτή αποτελεί μία γενίκευση της *κανονικής μορφής Kuroda*, που εφαρμόζεται στις γραμματικές με συμφραζόμενα και θα αναπτύξουμε πιο κάτω.

Θεώρημα 4.1.1. *Κάθε γραμματική χωρίς περιορισμούς είναι ισοδύναμη με μία γραμματική* $G' = (V, T, P, S)$*, της οποίας κάθε κανόνας είναι σε μία από τις παρακάτω μορφές:*

- $S \to \varepsilon$
- $A \to a$
- $A \to B$
- $A \to BC$
- $AB \to AC$
- $AB \to CB$
- $AB \to B$

όπου $A, B, C \in V$*,* $a \in T$ *και το* S *εμφανίζεται μόνο στα αριστερά μέλη των κανόνων.*

Ιδέα απόδειξης. Εκτελούμε στην αρχική γραμματική μια σειρά από μετασχηματισμούς, ώστε να αποκτήσουμε μια γραμματική στην παραπάνω κανονική μορφή, που παράγει την ίδια γλώσσα. Λεπτομέρειες στο [16]. □

4.1.2 Μηχανές Turing

Στρεφόμαστε τώρα σε ένα μοντέλο πολύ πιο ισχυρό από το πεπερασμένο αυτόματο, του οποίου η γέννηση το 1936 από τον Alan Turing έδωσε τα θεμέλια για τη Θεωρία Υπολογισμού, καθώς απέφερε μία κοινώς αποδεκτή τυποποίηση για τη διαισθητική έννοια του «αλγορίθμου». Η μηχανή Turing TM είναι αρκετά όμοια με ένα πεπερασμένο αυτόματο· έχοντας, όμως, απεριόριστη μνήμη

με τη μορφή μίας *άπειρης ταινίας*, στην οποία μπορεί να κινείται και προς τις δύο κατευθύνσεις και, όχι μόνο να διαβάζει, αλλά επιπλέον να γράφει, αποτελεί ένα πολύ πιο ακριβές μοντέλο υπολογιστή.

Στην πραγματικότητα, η μηχανή Turing, όσο πρωτόγονη κι αν φαίνεται, μπορεί να κάνει οτιδήποτε μπορεί να κάνει ένας ηλεκτρονικός υπολογιστής και δεν αντιπροσωπεύει απλώς ακόμα μία κλάση αυτομάτων που δύναται να εξελιχθεί σε ένα πιο ισχυρό είδος. Έτσι, οποιαδήποτε προσπάθεια να την «ενισχύσουμε» δεν έχει κανένα αποτέλεσμα και τα ποικίλα είδη «επαυξημένης» μηχανής Turing που έχουν κατά καιρούς προταθεί[1] μπορούν σε κάθε περίπτωση να προσομοιωθούν από το βασικό, πρότυπο μοντέλο. Με λίγα λόγια,η μηχανή Turing δείχνει να είναι η απόλυτη υπολογιστική μηχανή, το τέλος της πορείας μας προς όλο και πιο ισχυρά αυτόματα.

Τυπικά, μία **μηχανή Turing** (Turing machine) είναι μία διατεταγμένη επτάδα $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, όπου

- Q είναι ένα μη κενό, πεπερασμένο σύνολο **καταστάσεων**,
- Σ είναι ένα αλφάβητο (το **αλφάβητο εισόδου**),
- Γ είναι ένα αλφάβητο (το **αλφάβητο ταινίας**), τέτοιο ώστε $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$ είναι μία *μερική* συνάρτηση, η **συνάρτηση μετάβασης**,
- $q_0 \in Q$ είναι η **αρχική κατάσταση**,
- $B \in \Gamma - \Sigma$ είναι το **κενό σύμβολο**,
- $F \subseteq Q$ είναι το σύνολο των **τελικών καταστάσεων**.

Αρχικά η είσοδος εισάγεται στην ταινία, και, καθώς η είσοδος είναι πεπερασμένου μήκους, ενώ η ταινία άπειρου, το υπόλοιπο της ταινίας δεξιά και αριστερά της εισόδου περιέχει το κενό σύμβολο B. Θεωρούμε ότι η ανάγνωση των περιεχομένων της ταινίας γίνεται μέσω μίας *κεφαλής*, που στην αρχή βρίσκεται στο αριστερό άκρο της εισόδου. Σε κάθε βήμα η M ανιχνεύει με την κεφαλή της ένα σύμβολο της ταινίας και, σύμφωνα με τις επιβολές της συνάρτησης μετάβασης, γράφει ένα σύμβολο από το αλφάβητο ταινίας στη θέση του (που δεν αποκλείεται να είναι και το ίδιο), κινεί την κεφαλή μία θέση αριστερά ή δεξιά[2] και περνά στην επόμενη κατάσταση.

[1] Ενδεικτικά μόνο, αναφέρουμε τις TMs με πολλαπλές ή πολυδιάστατες ταινίες και τις TMs με πολλαπλές κεφαλές.

[2] Το L συμβολίζει «αριστερά» και το R «δεξιά».

Εφόσον η ταινία της είναι άπειρη, είναι αδύνατο να περιγράψουμε την κατάσταση της λειτουργίας της M ακριβώς. Έτσι, στην περιγραφή μας περιλαμβάνουμε μόνο το κομμάτι της ταινίας που βρίσκεται ανάμεσα στα άπειρα κενά (φυσικά συμπεριλαμβανομένου, ενδεχομένως, και του κομματιού ανάμεσα στην κεφαλή και το πρώτο μη κενό σύμβολο). Αυτό σίγουρα είναι πεπερασμένο, αφού σε πεπερασμένα βήματα η κεφαλή μπορεί να ανιχνεύσει μόνο πεπεραμένο πλήθος συμβόλων. Επομένως ορίζουμε ως **συνολική κατάσταση** της TM M ένα κατάλληλο στοιχείο του $Q \times \Gamma^* \times \Gamma^*$ και θεωρούμε ότι στη συνολική κατάσταση $(q, \alpha, X\beta)$, το σύμβολο στο οποίο βρίσκεται η κεφαλή είναι το X.

Οι σχέσεις της **παραγωγής σε ένα βήμα** $\vdash_M$ και **παραγωγής** $\vdash_M^*$ ορίζονται με τον προφανή τρόπο και δε θα μπούμε στις λεπτομέρειες. Χαρακτηριστικά αναφέρουμε ότι αν $\delta(q, X) = (q', X', L)$ τότε

$$(q, \varepsilon, X\alpha) \vdash_M (q', \varepsilon, BX'\alpha),$$

και αν $X' = B$, τότε

$$(q, \alpha Y, X) \vdash_M (q', \alpha, Y).$$

Αποδοχή και γλώσσες

Όπως και με τα πεπερασμένα αυτόματα, θα θεωρούμε ότι η αποδοχή μίας λέξης από μια TM λαμβάνει χώρα όταν η μηχανή οδηγηθεί σε κάποια τελική κατάσταση. Αυτό όμως με τη διαφορά ότι η μηχανή Turing δεν είναι υποχρεωμένη, όπως το πεπερασμένο αυτόματο, να διαβάσει ολόκληρη την είσοδο προτού τη δεχτεί. Έτσι, λέμε ότι η **συμβολοσειρά** $w \in \Sigma^*$ **γίνεται δεκτή** από τη M, αν υπάρχουν $q \in F$ και $\alpha, \beta \in \Gamma^*$, έτσι ώστε

$$(q_0, \varepsilon, w) \vdash_M^* (q, \alpha, \beta).$$

Ως **γλώσσα** της TM M ορίζουμε την

$$L(M) = \{w \in \Sigma^* \mid (q_0, \varepsilon, w) \vdash_M^* (q, \alpha, \beta) \text{ για κάποια } q \in F,\ \alpha, \beta \in \Gamma^*\}.$$

Παράδειγμα 4.1.1. Η $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$, με τη συνάρτηση μετάβασης δ που φαίνεται στον Πίνακα 4.1, είναι μία μηχανή Turing που δέχεται τη γλώσσα $L(M) = \{0^n 1^n \mid n \geq 1\}$.

Οι γλώσσες που γίνονται δεκτές από κάποια μηχανή Turing καλούνται **αναδρομικά απαριθμητές** (recursively enumerable) γλώσσες.

Μία άλλη σημαντική έννοια σχετική με τις μηχανές Turing είναι αυτή του τερματισμού. Λέμε ότι η M **τερματίζει** (halts), όταν κατά τον υπολογισμό

	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Πίνακας 4.1: Πίνακας καταστάσεων για την TM του Παραδείγματος 4.1.1.

της έρθει αντιμέτωπη με ένα ζεύγος κατάστασης και συμβόλου (q, X), για το οποίο η (μερική) συνάρτηση μετάβασης δ δεν είναι ορισμένη.

Δεδομένης μίας TM, μπορούμε να υποθέτουμε χωρίς βλάβη της γενικότητας ότι η TM τερματίζει όποτε δέχεται την είσοδό της. Αντίθετα, δεν είναι πάντοτε δυνατό να απαιτήσουμε ότι μια TM τερματίζει, ακόμα και όταν δε δέχεται την είσοδο, καθώς για κάποιες εισόδους η TM μπορεί να μην τερματίζει ποτέ. Οι γλώσσες εκείνες, για τις οποίες υπάρχει μηχανή Turing που τερματίζει για κάθε είσοδο—ανεξάρτητα από το εάν τη δέχεται ή όχι—λέγονται **αναδρομικές** (recursive) γλώσσες. Όπως είναι γνωστό, το σύνολο των αναδρομικών γλωσσών περιέχεται γνήσια σε εκείνο των αναδρομικά απαριθμητών.

4.1.3 Αυτόματα με δύο στοίβες

Έχει ενδιαφέρον στο σημείο αυτό να παρατηρήσουμε ότι η μηχανή Turing δεν είναι το μόνο είδος μηχανής που αυξάνει την ισχύ ενός αυτομάτου στοίβας. Η πρώτη ιδέα που έρχεται στο νου για κάτι τέτοιο είναι μάλλον η ενίσχυση του αυτομάτου στοίβας με μία ακόμα στοίβα. Έτσι γεννιέται το αυτόματο δύο στοιβών, που, κατά ένα θαυμαστό τρόπο, είναι με δύο μόλις στοίβες ήδη τόσο ισχυρό, ώστε να μην μπορούμε να αυξήσουμε κι άλλο την ισχύ του, και προκύπτει υπολογιστικά ισοδύναμο με τη μηχανή Turing.

Ένα **αυτόματο με δύο στοίβες**[3] (two-pushdown automaton 2-PDA) είναι μία διατεταγμένη επτάδα $T = (Q, \Sigma, \Gamma, \delta, Z_0, F)$, όπου

- Q είναι ένα μη κενό, πεπερασμένο σύνολο **καταστάσεων**,
- Σ είναι ένα αλφάβητο (το **αλφάβητο εισόδου**),

[3]Θεωρούμε, όπως και στην περίπτωση του απλού αυτομάτου στοίβας, τη μη ντετερμινιστική εκδοχή. Αντίθετα, όμως, από το αυτόματο στοίβας, και σε συμφωνία με τη μηχανή Turing, ο μη ντετερμινισμός στο αυτόματο με δύο στοίβες δεν είναι παρά απλώς ένα τεχνικό χαρακτηριστικό. Δεν έχει επιρροή στην υπολογιστική του ισχύ και ως προς αυτή δεν του παρέχει υπεροχή από το μη ντετερμινιστικό.

- Γ είναι ένα αλφάβητο (το **αλφάβητο στοίβας**), τέτοιο ώστε $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, N, E, I\})$ είναι η **συνάρτηση μετάβασης**,
- $q_0 \in Q$ είναι η **αρχική κατάσταση**,
- $Z_0 \in \Gamma - \Sigma$ είναι το **αρχικό σύμβολο στοίβας**,
- $F \subseteq Q$ είναι το σύνολο των **τελικών καταστάσεων**.

Το T ξεκινά τον υπολογισμό της εισόδου $w = x_1 x_2 \dots x_n \in \Sigma^*$ τοποθετώντας στην πρώτη στοίβα μόνο το αρχικό σύμβολο Z_0 και στη δεύτερη την είσοδο w αντίστροφα, δηλαδή έτσι ώστε το x_n να βρίσκεται στον πάτο της δεύτερης στοίβας και το x_1 στην κορυφή. Σε αντίθεση με το αυτόματο μίας στοίβας, εδώ δεν υπάρχει ξεχωριστή «ταινία» από την οποία διαβάζεται η είσοδος.

Οι κινήσεις του T σε κάθε βήμα καθορίζονται μέσω της δ από την κατάσταση στην οποία αυτό βρίσκεται, καθώς και από το σύμβολο της κορυφής κάθε μίας από τις δύο στοίβες. Κάθε τέτοια κίνηση περιλαμβάνει απαραιτήτως την εισαγωγή ενός συμβόλου στην κορυφή κάποιας στοίβας. Ο συγκεκριμένος τρόπος με τον οποίο θα γίνει αυτό υπαγορεύεται από τα πέντε σύμβολα L, R, N, E, I που «εξάγει» η δ και που αντιστοιχούν σε ισάριθμους διαφορετικούς τύπους κινήσεων: L για «αριστερά» (left-shift), R για «δεξιά» (right-shift), N για «στασιμότητα» (no-shift), E για «διαγραφή» (erase) και I για «εισαγωγή» (insert).

Έτσι, ως **συνολική κατάσταση** του T ορίζουμε ένα στοιχείο του $Q \times \Gamma^* \times \Gamma^*$. Αν (q, α, β) είναι μια συνολική κατάσταση, θεωρούμε ότι α και β είναι τα περιεχόμενα της πρώτης και δεύτερης στοίβας, αντίστοιχα, και ότι το T διαβάζει το τελευταίο σύμβολο του α και το πρώτο του β. Τώρα μπορούμε να ορίσουμε αυστηρά τον τρόπο με τον οποίο γίνονται οι κινήσεις του T. Αν $(q, \alpha X, Y\beta)$, (q', α', β') είναι δύο συνολικές καταστάσεις, λέμε ότι η $(q, \alpha X, Y\beta)$ **παράγει σε ένα βήμα** τη (q', α', β'),

$$(q, \alpha X, Y\beta) \vdash_T (q', \alpha', \beta'),$$

αν και μόνο αν $(q', Z, \mathbf{D}) \in \delta(q, X, Y)$ και ισχύει ένα από τα παρακάτω:

1. $\mathbf{D} = L$, $\alpha' = \alpha$, $\beta' = ZY\beta$
2. $\mathbf{D} = R$, $\alpha' = \alpha XZ$, $\beta' = \beta$
3. $\mathbf{D} = N$, $\alpha' = \alpha X$, $\beta' = Z\beta$
4. $\mathbf{D} = E$, $\alpha' = \alpha$, $\beta' = Z\beta$
5. $\mathbf{D} = I$, $\alpha' = \alpha X$, $\beta' = ZY\beta$

Η ανακλαστική, μεταβατική κλειστότητα της $\vdash_T$ συμβολίζεται, όπως συνήθως, με $\vdash_T^*$. Όπως συνέβη και με το PDA, υπάρχουν δύο τρόποι ορισμού της έννοιας της αποδοχής από ένα 2-PDA. Έτσι, **η γλώσσα που γίνεται δεκτή σε τελική κατάσταση** από το T είναι η

$$L(T) = \{w \in \Sigma^* \mid (q_0, Z_0, w) \vdash_T^* (q, \alpha, \varepsilon) \text{ για κάποια } q \in F,\ \alpha \in \Gamma^+\},$$

ενώ **η γλώσσα που γίνεται δεκτή με κενή στοίβα** από το T είναι η

$$N(T) = \{w \in \Sigma^* \mid (q_0, Z_0, w) \vdash_T^* (q, \varepsilon, Z) \text{ για κάποια } q \in Q,\ Z \in \Gamma\}.$$

Και πάλι οι δύο ορισμοί είναι ισοδύναμοι και γι' αυτό θα μιλάμε γενικά για τη **γλώσσα** του T.

Θεώρημα 4.1.2. *$L = L(T_F)$ για κάποιο 2-PDA T_F αν και μόνο αν $L = N(T_N)$ για κάποιο 2-PDA T_N.*

Ιδέα απόδειξης. Κατά βάση προσομοιώνοντας τη λειτουργία της μιας μηχανής από την άλλη, με μικρές αλλαγές. Περισσότερα μπορεί να δει κανείς στο [16]. □

Όπως είπαμε, το 2-PDA στην ουσία δεν είναι παρά ακόμα μία εκδοχή της κλασικής TM.

Θεώρημα 4.1.3. *Για κάθε TM M υπάρχει ένα 2-PDA T τέτοιο ώστε L(T)= L(M) και αντίστροφα, για κάθε 2-PDA T υπάρχει μία TM M τέτοια ώστε L(M)=L(T).*

Ιδέα απόδειξης. Είναι φανερή διαισθητικά η ομοιότητα των αυτομάτων με δύο στοίβες με τις μηχανές Turing: Μπορούμε να φανταστούμε ότι η ταινία της μηχανής Turing κόβεται στο σημείο όπου βρίσκεται η κεφαλή, και τα μη κενά τμήματα των δύο κομματιών τοποθετούνται στις δύο στοίβες. Στην πρώτη εισάγεται το τμήμα αριστερά της κεφαλής, ενώ στη δεύτερη εκείνο από το σημείο της κεφαλής και δεξιά, με τρόπο ώστε τα σύμβολα που είναι πλησιέστερα στην κεφαλή να βρίσκονται προς την κορυφή κάθε στοίβας. Οι κινήσεις της κεφαλής της TM αντιστοιχίζονται χωρίς δυσκολία στις κινήσεις του 2-PDA, που μπορεί να κάνει ακριβώς ό,τι και η TM. Για τις τεχνικές λεπτομέρειες μπορεί να ανατρέξει κανείς στο [16]. □

4.1.4 Μηχανές Turing και γραμματικές χωρίς περιορισμούς

Είναι εντυπωσιακό—και προσδίδει ακόμα περισσότερο στη βαρύτητά τους—ότι οι μηχανές Turing είναι ισοδύναμες σε παραγωγική ισχύ όχι μόνο με τα αυτόματα δύο στοιβών, αλλά και με τις γραμματικές χωρίς περιορισμούς.

Η ισοδυναμία αυτή δείχνεται είτε ευθέως, ανάμεσα στις γραμματικές χωρίς περιορισμούς και τις μηχανές Turing, είτε ανάμεσα στις γραμματικές χωρίς περιορισμούς και τα αυτόματα με δύο στοίβες, με χρήση του Θεωρήματος 4.1.3. Μία απόδειξη με τον πρώτο τρόπο υπάρχει στο [7], ενώ μία με τον δεύτερο στο [16].

Πρόταση 4.1.1. *Αν η G είναι μία γραμματική χωρίς περιορισμούς, τότε η L(G) είναι μία αναδρομικά απαριθμητή γλώσσα.*

Πρόταση 4.1.2. *Αν η L είναι μία αναδρομικά απαριθμητή γλώσσα, τότε L=L(G) για κάποια γραμματική χωρίς περιορισμούς G.*

Έτσι προκύπτει το βασικό Θεώρημα ισοδυναμίας γραμματικών χωρίς περιορισμούς και μηχανών Turing, χάρη στο οποίο μπορούμε να χρησιμοποιούμε γνωστά αποτελέσματα για τις αναδρομικά απαριθμητές γλώσσες με στόχο να εξάγουμε συμπεράσματα για τις γραμματικές χωρίς περιορισμούς, και αντίστροφα.

Θεώρημα 4.1.4. *Μία γλώσσα είναι γλώσσα χωρίς περιορισμούς αν και μόνο αν είναι αναδρομικά απαριθμητή.*

Απόδειξη. Άμεση από τις Προτάσεις 4.1.1 και 4.1.2. □

Πρέπει να παρατηρήσουμε βέβαια, ότι ο συγκεκριμένος προσδιορισμός των γλωσσών χωρίς περιορισμούς δεν είναι και τόσο χρήσιμος. Αυτό οφείλεται στο γεγονός ότι ο τρόπος με τον οποίο γίνεται αποδοχή από μία μηχανή Turing είναι μάλλον μονομερής, με την έννοια ότι, εφόσον ο τερματισμός δεν είναι εξασφαλισμένος, δεν παρέχεται καμια πληροφορία για το πότε η είσοδος *δεν* ανήκει στη γλώσσα. Το συγκεκριμένο χαρακτηριστικό όμως δεν είναι ασυνεπές με τη φύση των γραμματικών χωρίς περιορισμούς: Αν αφενός μία συμβολοσειρά μπορεί να παραχθεί από μία γραμματική χωρίς περιορισμούς, ξεκινώντας να σχηματίζουμε μία-μία όλες τις δυνατές παραγωγές, από τις μικρότερου προς τις μεγαλύτερου μήκους, κάποια στιγμή θα βρούμε μία «σωστή» και θα σταματήσουμε. Αν αφετέρου δεν υπάρχει καμία τέτοια παραγωγή, η διαδικασία αυτή θα συνεχιστεί ατέρμονα χωρίς να είμαστε σε θέση να συμπεράνουμε αν υπάρχει αυτό που ψάχνουμε.

Σημειώνουμε ότι, αν και δεν υπάγονται στην ιεραρχία του Chomsky, υπάρχουν και γραμματικές που μπορούν να προσδιοριστούν με πιο χρήσιμα είδη υπολογισμού βασιζόμενα στις μηχανές Turing, και που σχετίζονται με τις αναδρομικές γλώσσες.

4.2 Γλώσσες με συμφραζόμενα

Στην ενότητα αυτή θα εξετάσουμε την κλάση $\mathcal{L}_1$ των γλωσσών με συμφραζόμενα, που περιέχει γνήσια την κλάση των γλωσσών χωρίς συμφραζόμενα, αλλά περιέχεται γνήσια σε εκείνη των γλωσσών χωρίς περιορισμούς.[4] Αν και η χρησιμότητα των γλωσσών με συμφραζόμενα είναι αρκετά περιορισμένη σε σχέση με αυτή των χωρίς, έχουν και αυτές κάποιον ρόλο στον ορισμό των γλωσσών προγραμματισμού, όταν μοντελοποιούνται στοιχεία ευαίσθητα στα συμφραζόμενα. Πέραν αυτού, είναι διαισθητικά εμφανές ότι οι σχετιζόμενες με τα συμφραζόμενα πληροφορίες είναι δυνητικά χρήσιμες κατά την επεξεργασία οποιασδήποτε γλώσσας.

Θα δούμε ότι το υπολογιστικό μοντέλο που συνδέεται με τις γλώσσες χωρίς συμφραζόμενα είναι ένα είδος μηχανής Turing με περιορισμένες δυνατότητες, το γραμμικά φραγμένο αυτόματο.

4.2.1 Μονοτονικές γραμματικές

Μία τυπική γραμματική $G = (V, T, P, S)$ ονομάζεται **μονοτονική** (monotonic) ή **μη φθίνουσα** (noncontracting, nondecreasing) αν για κάθε κανόνα $\alpha \to \beta \in P$, είναι $|\alpha| \leq |\beta|$.

Δηλαδή σε μία μονοτονική γραμματική δεν υπάρχει κανένας περιορισμός ως προς τη μορφή του αριστερού μέλους ενός κανόνα, αλλά το μήκος του δεξιού του μέλους οφείλει να είναι τουλάχιστον όσο και αυτό του αριστερού. Έτσι, μία μονοτονική γραμματική είναι εκ φύσεως αδύνατο να παράγει την κενή συμβολοσειρά ε, που έχει μήκος 0.

Δείχνεται ότι η μονοτονικότητα των κανόνων είναι ισοδύναμη με την ιδιότητα της ευαισθησίας στα συμφραζόμενα, με μοναδική εξαίρεση τον κανόνα $S \to \varepsilon$, που είναι επιτρεπτός στις γραμματικές με συμφραζόμενα προκειμένου να παραχθεί το ε.

Θεώρημα 4.2.1. *Κάθε γραμματική με συμφραζόμενα που δεν παράγει το ε είναι μονοτονική και κάθε μονοτονική γραμματική είναι και γραμματική με συμφραζόμενα.*

Ιδέα απόδειξης. Είναι φανερό από τους ορισμούς ότι μία γραμματική με συμφραζόμενα που δεν παράγει το ε είναι μονοτονική. Στο [16] περιγράφεται ένας απλός αλγόριθμος μετατροπής των μονοτονικών κανόνων σε κανόνες ευαίσθητους στα συμφραζόμενα, χωρίς να μεταβληθεί η παραγόμενη γλώσσα. □

[4] Για την ακρίβεια, η $\mathcal{L}_1$ περιέχεται γνήσια ακόμα και στην κλάση των αναδρομικών γλωσσών. Παρ' όλα αυτά, σχεδόν κάθε γλώσσα που μπορεί κανείς να σκεφτεί ανήκει στην $\mathcal{L}_1$ και η απόδειξη ύπαρξης γλωσσών που δεν ανήκουν σε αυτή γίνεται κυρίως με τη μέθοδο της διαγωνιοποίησης.

Επομένως, αν και οι παραγωγές των γραμματικών χωρίς συμφραζόμενα είναι γενικά αρκετά περίπλοκες, τουλάχιστον σε θεωρητικό επίπεδο μπορούμε να τις «διαχειριστούμε» εύκολα χάρη στην ιδιότητα της μονοτονικότητας. Έτσι, αν θέλουμε να αποφανθούμε για το εάν μια συγκεκριμένη λέξη w παράγεται από μία γραμματική με συμφραζόμενα, αρκεί να ελέγξουμε συστηματικά όλες τις παραγωγές λέξεων μήκους *το πολύ όσο και η* w. Αν καμία από αυτές δεν αποδίδει τη w, τότε η w ξεκάθαρα δεν είναι στη γλώσσα της γραμματικής, καθώς κάθε άλλη παραγωγή θα αποδίδει λέξεις μεγαλύτερου μήκους. Άλλωστε, αφού κάθε αλφάβητο είναι πεπερασμένο, το πλήθος των παραγωγών που χρειάζεται να ελέγξουμε είναι πεπερασμένο.

Από την άλλη, αυτή η διαδικασία δεν είναι καθόλου πρακτική. Γενικά απαιτούνται πάρα πολλά βήματα για να εφαρμοστεί, με αποτέλεσμα να είναι συχνά αναγκαίο να βρεθούν πιο αποδοτικοί αλγόριθμοι, που, τουλάχιστον για κάποιες ειδικές περιπτώσεις, είναι γρηγορότεροι. Τέτοιες τεχνικές περιγράψαμε στην Ενότητα 3.5, αλλά δυστυχώς δουλεύουν μόνο για την περίπτωση των γραμματικών χωρίς συμφραζόμενα.

Τέλος, είναι φανερό ότι η παραπάνω διαδικασία δεν είναι αποτελεσματική για μία τυχούσα γραμματική χωρίς περιορισμούς, καθώς στις παραγωγές μιας τέτοιας γραμματικής είναι δυνατό βήματα που μειώνουν το μήκος της λέξης να διαδέχονται εκείνα που το αυξάνουν. Αυτό έχει ως συνέπεια να μην μπορούμε, όπως στην περίπτωση της μονοτονικής γραμματικής, να καθορίσουμε κάποιο όριο για το που θα τερματιστεί ο αλγόριθμος. Το γεγονός αυτό δείχνει και τη δυσκολία του προβλήματος να αποφασίσουμε αν μία λέξη ανήκει στη γλώσσα μίας γραμματικής γενικά.

4.2.2 Κανονική μορφή Kuroda και γραμματικές με μονομερή συμφραζόμενα

Η κανονική μορφή που ορίσαμε για τις γλώσσες χωρίς περιορισμούς δεν ήταν παρά ένα είδος γενίκευσης της κανονικής μορφής Kuroda, που σχετίζεται με τις γλώσσες με συμφραζόμενα. Μία μονοτονική γραμματική $G = (V, T, P, S)$ είναι σε **κανονική μορφή Kuroda** (Kuroda normal form), αν κάθε κανόνας της είναι σε μία από τις παρακάτω μορφές:

- $A \to a$
- $A \to B$
- $A \to BC$
- $AB \to CD$

όπου $A, B, C, D \in V$ και $a \in T$.

Θεώρημα 4.2.2. *Κάθε μονοτονική γραμματική είναι ισοδύναμη με μία γραμματική σε κανονική μορφή Kuroda.*

Η απόδειξη περιγράφεται στο [16]. Έτσι πηγάζει άμεσα η σύνδεση των γραμματικών με συμφραζόμενα με την κανονική μορφή Kuroda.

Πόρισμα 4.2.1. *Κάθε γραμματική με συμφραζόμενα που δεν παράγει το ε είναι ισοδύναμη με μία γραμματική σε κανονική μορφή Kuroda.*

Απόδειξη. Άμεση από τα Θεωρήματα 4.2.1 και 4.2.2. □

Ένα ερώτημα που παρέμεινε αναπάντητο για αρκετό καιρό είναι εάν μπορούμε να φέρουμε μία τυχούσα γραμματική με συμφραζόμενα σε τέτοια μορφή, ώστε οι κανόνες της να έχουν συμφραζόμενα αποκλειστικά από τη μία πλευρά (την ίδια για όλους)—είτε μόνο αριστερά, είτε μόνο δεξιά της μεταβλητής που αντικαθίσταται. Αυτού του είδους οι γραμματικές με συμφραζόμενα ονομάζονται **γραμματικές με μονομερή συμφραζόμενα** (one-sided context-sensitive grammars) και χρησιμοποιούνται ως φυσικά εργαλεία για την επεξεργασία λέξεων από αριστερά προς τα δεξιά, ακόμα και όταν η γλώσσα είναι στην ουσία χωρίς συμφραζόμενα.

Το ερώτημα απαντήθηκε τελικά καταφατικά, αλλά με έναν μάλλον περίπλοκο τρόπο και η σημασία του παρέμεινε θεωρητικής φύσης, καθότι είναι υπερβολικά δύσκολο να φτάσουμε σε αυτού του είδους την κανονική μορφή για μία τυχούσα γραμματική με συμφραζόμενα.

Θεώρημα 4.2.3. *Κάθε γραμματική με συμφραζόμενα είναι ισοδύναμη με μία γραμματική με μονομερή συμφραζόμενα.*

Η απόδειξη έγινε από τον M. Pentonnen το 1974 με αρκετά περίπλοκο τρόπο, χρησιμοποιώντας ιδέες των Haines και Gladkij. Πολύ πιο εύκολο είναι να δειχτεί το κάπως ασθενέστερο αποτέλεσμα, που δηλώνει ότι κανόνες *με ευαισθησία στα αριστερά συμφραζόμενα* (left-context sensitive), σε συνδυασμό με *αναστρέφοντες* (inverting) κανόνες της μορφής $AB \to BA$, επαρκούν για την παραγωγή οποιασδήποτε γλώσσας με συμφραζόμενα.

Θεώρημα 4.2.4. *Κάθε γραμματική με συμφραζόμενα που δεν παράγει το ε είναι ισοδύναμη με μία γραμματική $G' = (V', T', P', S')$, της οποίας οι κανόνες έχουν τις μορφές:*

- $A \to a$
- $A \to B$

- $A \to BC$
- $AB \to AC$
- $AB \to BA$

όπου $A, B, C \in V'$, $a \in T'$.

Ιδέα απόδειξης. Χρησιμοποιώντας το Πόρισμα 4.2.1, απομένει να απαλλαχθούμε από τους κανόνες της μορφής $AB \to CD$. Αυτό γίνεται με την εισαγωγή μιας σειράς από νέους κανόνες της μορφής $EF \to EG$ και $EF \to FE$, που περιγράφεται στο [16]. □

4.2.3 Γραμμικά φραγμένα αυτόματα

Όπως ήδη αναφέραμε, υπάρχει πληθώρα παραλλαγών της πρότυπης μηχανής Turing, η καθεμία από τις οποίες παραμένει πεισματικά ισοδύναμη με την αρχική. Εάν, όμως, επιλέξουμε να περιορίσουμε το κομμάτι της ταινίας που διατίθεται για τους υπολογισμούς, θα οδηγηθούμε σε ένα μοντέλο με αυστηρά μικρότερη ισχύ. Αυτό είναι το γραμμικά φραγμένο αυτόματο.

Ένα **γραμμικά φραγμένο αυτόματο** (linear bounded automaton LBA) ή **επί τόπου αποδέκτης** (in-place acceptor) είναι μία διατεταγμένη οκτάδα

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \mathfrak{c}, \$, F),$$

όπου τα $Q, \Sigma, \Gamma, \delta, q_0, F$ είναι όπως ορίστηκαν για τη μη ντετερμινιστική μηχανή Turing. Τα $\mathfrak{c}, \$$ είναι ειδικά σύμβολα του Γ: ο αριστερός και ο δεξιός *δείκτης τέλους* (endmarker), αντίστοιχα.

Οι δείκτες τέλους βρίσκονται στην ταινία αρχικά και η είσοδος εισάγεται ανάμεσά τους, χωρίς οι ίδιοι να θεωρούνται μέρος της εισόδου. Η κεφαλή του LBA δεν μπορεί να κινηθεί αριστερά του $\mathfrak{c}$ ή δεξιά του \$, ούτε και να γράψει άλλα σύμβολα στη θέση τους, διαγράφοντάς τους. Έτσι, ένα LBA είναι κατά βάση μία μηχανή Turing, η οποία, αντί να έχει άπειρη ταινία για τους υπολογισμούς της, είναι περιορισμένη στο κομμάτι της ταινίας που αντιστοιχεί στην είσοδο, συν τα δύο τετράγωνα που περιέχουν τους δείκτες τέλους και βρίσκονται αμέσως αριστερά και δεξιά της εισόδου.

Αφού η κεφαλή ενός LBA δεν μπορεί να κινηθεί πέρα από τους δείκτες τέλους, δεν υπάρχει λόγος να υποθέσουμε, όπως στη γενική περίπτωση της μηχανής Turing, την ύπαρξη άπειρων κενών συμβόλων αριστερά και δεξιά της εισόδου. Έτσι, ορίζουμε ως **συνολική κατάσταση** του LBA M ένα στοιχείο του $Q \times \Gamma^* \times \Gamma^*$ και οι σχέσεις της **παραγωγής σε ένα βήμα** $\vdash_M$ και **παραγωγής** $\vdash_M^*$ ανάμεσα σε δύο συνολικές καταστάσεις είναι όπως και

στη μηχανή Turing, με τα δύο ιδιαίτερα χαρακτηριστικά που προαναφέραμε: Οι δείκτες τέλους δεν επιτρέπεται να διαγραφούν, και όταν η κεφαλή διαβάζει ¢ απαγορεύεται να κινηθεί προς τα αριστερά, ενώ όταν διαβάζει $ απαγορεύεται να κινηθεί προς τα δεξιά.

Η **συμβολοσειρά** $w \in (\Sigma - \{¢, \$\})^*$ **γίνεται δεκτή** από το M, αν υπάρχει υπολογισμός, που με είσοδο $¢w\$$ φτάνει σε κάποια τελική κατάσταση. Δηλαδή αν υπάρχουν $q \in F$ και $\alpha, \beta \in \Gamma^*$, έτσι ώστε

$$(q_0, \varepsilon, ¢w\$) \vdash_M^* (q, \alpha, \beta).$$

Επομένως, ως **γλώσσα** του M ορίζουμε την

$$L(M) = \{w \in (\Sigma - \{¢, \$\})^* \mid (q_0, \varepsilon, ¢w\$) \vdash_M^* (q, \alpha, \beta) \text{ για κάποια } q \in F,\ \alpha, \beta \in \Gamma^*\}.$$

Ο όρος «γραμμικά φραγμένο» αυτόματο στην πραγματικότητα κατάγεται από μία άλλη παραλλαγή της μηχανής Turing, στην οποία η κεφαλή της μηχανής περιορίζεται σε κομμάτι της ταινίας μήκους όχι όσο και της εισόδου (συν δύο), αλλά φραγμένου από κάποια γραμμική συνάρτηση του μήκους της εισόδου. Τα δύο μοντέλα προκύπτουν ακριβώς της ίδιας υπολογιστικής ισχύος, κι έτσι ο χαρακτηρισμός επικράτησε και στα δύο.

Αντίθετα, δεν είναι γνωστό εάν η μη ντετερμινιστική εκδοχή των γραμμικά φραγμένων αυτομάτων είναι αυστηρά πιο ισχυρή από την ντετερμινιστική, ή εάν, όπως και στην περίπτωση των μηχανών Turing, δεν υπάρχει διαφορά ανάμεσα στην υπολογιστική ισχύ των δύο. Αυτό αποτελεί το λεγόμενο *πρόβλημα του γραμμικού αυτομάτου* (LBA problem) και είναι μάλλον το πιο διάσημο ανοικτό πρόβλημα της θεωρίας αυτομάτων.

Γραμμικά φραγμένα αυτόματα και αυτόματα με δύο στοίβες

Εξαιρετικά ενδιαφέρουσα είναι η σχέση των γραμμικά φραγμένων αυτομάτων με τα αυτόματα με δύο στοίβες, που είδαμε ότι είναι ισοδύναμα με τις μηχανές Turing. Εύκολα υποψιαζόμαστε ότι μπορούμε να περιορίσουμε τα 2-PDAs κατά τέτοιο τρόπο, ώστε οι γλώσσες που δέχονται να είναι οι ίδιες με αυτές των LBAs. Το καίριο ερώτημα είναι με ποιον τρόπο.

Προκύπτει ότι το επιθυμητό αποτέλεσμα επιτυγχάνεται εάν περιορίσουμε τον αποθηκευτικό χώρο των 2-PDAs, δηλαδή τη χωρητικότητα των στοιβών, στο μέγεθος της εισόδου. Συγκεκριμένα, στο [16] αποδεικνύεται ότι τα γραμμικά φραγμένα αυτόματα έχουν την ίδια ισχύ με αυτόματα με δύο στοίβες, στα οποία δεν υπάρχουν κινήσεις που εισάγουν σύμβολα στις στοίβες.

Θεώρημα 4.2.5. *Κάθε LBA είναι ισοδύναμο με ένα 2-PDA* $T = (Q, \Sigma, \Gamma, \delta, Z_0, F)$ *χωρίς παραγωγές της μορφής*

$$(q, \alpha, \beta) \vdash_T (p, \alpha, Z\beta)$$

για $q, p \in Q,\ \alpha, \beta \in \Gamma^*,\ Z \in \Gamma$.

Ένα από τα συμπεράσματα της επόμενης υποενότητας είναι ότι τα γραμμικά φραγμένα αυτόματα δέχονται ευρύτερη κλάση γλωσσών από εκείνη των αυτομάτων στοίβας. Δεδομένου του παραπάνω θεωρήματος, αυτό είναι αρκετά εντυπωσιακό, καθώς σημαίνει ότι μία στοίβα άπειρου μεγέθους είναι λιγότερο ισχυρή από δύο στοίβες πεπερασμένου μεγέθους.

4.2.4 Γραμμικά φραγμένα αυτόματα και γραμματικές με συμφραζόμενα

Τα γραμμικά φραγμένα αυτόματα φυσικά δεν είναι παρά ό,τι ακριβώς χρειάζεται για να γίνουν δεκτές οι γλώσσες με συμφραζόμενα.

Πρόταση 4.2.1. *Αν η G είναι μία γραμματική με συμφραζόμενα, τότε υπάρχει LBA M τέτοιο ώστε L=L(M).*

Ιδέα απόδειξης. Δεδομένης μίας γραμματικής με συμφραζόμενα $G = (V, T, P, S)$, κατασκευάζεται ένα γραμμικά φραγμένο αυτόματο που προσομοιώνει τις παραγωγές της. Εξαιτίας του περιορισμένου χώρου της ταινίας του, το LBA τερματίζει αυτόματα όταν σε κάποιο ενδιάμεσο βήμα για την παραγωγή της εισόδου $w \in T^*$ παράγεται μία λέξη με μήκος μεγαλύτερο από της w. Έτσι, το LBA δέχεται την w αν υπάρχει παραγωγή $S \Rightarrow_G^* w$, τέτοια ώστε καμία ενδιάμεση προτασιακή μορφή να μην είναι μακρύτερη της w.

Αυτό συμβαδίζει με τη μονοτονικότητα των γραμματικών με συμφραζόμενα, σύμφωνα με την οποία το δεξί μέλος οποιουδήποτε κανόνα έχει μήκος τουλάχιστον όσο και το αριστερό, και κατά συνέπεια δεν μπορεί να υπάρχει παραγωγή $S \Rightarrow^* \alpha \Rightarrow^* w$ με $|\alpha| > |w|$. Το LBA που κατασκευάζεται λοιπόν δέχεται όλες και μόνο εκείνες τις λέξεις που παράγει η G. Λεπτομέρειες στο [7]. □

Αντίστοιχα, δεδομένου ενός LBA, κατασκευάζεται μία γραμματική, που, χρησιμοποιώντας μόνο μονοτονικούς κανόνες, προσομοιώνει τις κινήσεις του LBA και παράγει τη λέξη w αν και μόνο αν το LBA δέχεται την w.

Πρόταση 4.2.2. *Αν το M είναι ένα LBA, τότε υπάρχει γραμματική με συμφραζόμενα G τέτοια ώστε L(G)=L(M).*

Ιδέα απόδειξης. Η απόδειξη είναι παράλληλη με αυτή της κατασκευής γραμματικής χωρίς περιορισμούς από TM της Πρότασης 4.1.2, με μικρές διαφορές, που έχουν να κάνουν κυρίως με τους δείκτες τέλους. Λεπτομέρειες στο [7]. □

Οι παραπάνω προτάσεις οδηγούν στο Θεώρημα ισοδυναμίας γραμματικών με συμφραζόμενα και γραμμικά φραγμένων αυτομάτων.

Θεώρημα 4.2.6. *Μία γλώσσα είναι γλώσσα με συμφραζόμενα αν και μόνο αν γίνεται δεκτή από ένα γραμμικά φραγμένο αυτόματο.*

Απόδειξη. Άμεση από τις Προτάσεις 4.2.1 και 4.2.2. □

4.3 Η ιεραρχία συγκεντρωτικά

Στα προηγούμενα μελετήσαμε τις τέσσερις κλάσεις γλωσσών της ιεραρχίας Chomsky και είδαμε πως καθεμία από αυτές χαρακτηρίζεται ως η κλάση των γλωσσών που παράγεται από μία οικογένεια γραμματικών και γίνεται δεκτή από έναν τύπο μηχανών. Οι σχέσεις που αναπτύσσονται και συνδέουν τις διαφορετικές έννοιες επισκοπούνται στον Πίνακα 4.2.

Τύπος	**Γραμματική**	**Γλώσσα**	**Μηχανή**
0	Χωρίς περιορισμούς	Αναδρομικά απαριθμητή/ Χωρίς περιορισμούς	Ντετερμινιστική/Μη ντετερμινιστική μηχανή Turing/Αυτόματο με δύο στοίβες
1	Μονοτονική/Με συμφραζόμενα	Με συμφραζόμενα	Μη ντετερμινιστικό γραμμικά φραγμένο αυτόματο
2	Χωρίς συμφραζόμενα	Χωρίς συμφραζόμενα	Μη ντετερμινιστικό αυτόματο στοίβας
3	Αριστερογραμμική/ Δεξιογραμμική/ Κανονική	Κανονική	Ντετερμινιστικό/Μη ντετερμινιστικό πεπερασμένο αυτόματο

Πίνακας 4.2: Η ιεραρχία Chomsky.

Όπως έχουμε τονίσει, οι συμπεριλήψεις των γλωσσών τύπου i σε εκείνες του τύπου $i-1$, $i = 1, 2, 3$, είναι όλες γνήσιες. Γενική μέθοδος, όμως, να απαντήσουμε αν μία δεδομένη γλώσσα ανήκει σε κάποια κλάση γλωσσών

δεν υπάρχει.[5] Έτσι, η απόδειξη ότι μία δεδομένη γλώσσα δεν ανήκει σε κάποια κλάση γίνεται με διάφορες στρατηγικές, όπως με την επίδειξη αδυναμίας των μηχανών του αντίστοιχου τύπου να δεχτούν τη γλώσσα, με χρήση των *Λημμάτων Άντλησης* (Pumping Lemmas), με εκμετάλλευση των ιδιοτήτων κλειστότητας, ή με την τεχνική της διαγωνιοποίησης.

Δε θα επεκταθούμε στις παραπάνω τεχνικές. Θα αναφέρουμε μόνο πως μία γλώσσα χωρίς συμφραζόμενα που δεν είναι κανονική είναι η $\{a^n b^n \mid n > 0\}$, ενώ μία γλώσσα με συμφραζόμενα που δεν είναι γλώσσα χωρίς συμφραζόμενα είναι η $\{a^n b^n c^n \mid n > 0\}$. Με διαγωνιοποίηση εξασφαλίζεται η ύπαρξη αναδρομικής γλώσσας που δεν είναι γλώσσα με συμφραζόμενα.

[5] Σημειώνουμε ότι για την ακρίβεια, τα περισσότερα ενδιαφέροντα προβλήματα που αφορούν γλώσσες είναι αναποκρίσιμα, καθώς πολλά από αυτά ανάγονται στο πρόβλημα τερματισμού (halting problem) της μηχανής Turing. Η μόνη κλάση γλωσσών για την οποία υπάρχουν αλγόριθμοι για αρκετά από τα προβλήματα που μας απασχολούν είναι η $\mathcal{L}_3$.

Κεφάλαιο 5

Πολυπλοκότητα υπολογιστικών μοντέλων με περιορισμένους πόρους

Στην πορεία που ακολουθήσαμε γνωρίσαμε διαφόρων ειδών υπολογιστικά μοντέλα, κάποια σαφώς ισχυρότερα από άλλα, και κάποια με φαινομενικές μόνο διαφορές, αλλά στην ουσία ισοδύναμα. Η εικόνα αυτή όμως είναι κάπως μονόπλευρη, καθώς περιγράφει την κατάσταση μόνο από την οπτική γωνία της υπολογισιμότητας. Μια άλλη πλευρά, τουλάχιστον εξίσου σημαντική, που τα τελευταία τριάντα χρόνια έχει μελετηθεί εκτενώς και υπόσχεται πολλά για τη μελλοντική επίλυση σπουδαίων πρακτικών ζητημάτων, όπως αυτό της αξιοπιστίας λογισμικού (software reliability), αλλά και του ελέγχου λογισμικού και μηχανικού εξοπλισμού (software and hardware testing), είναι εκείνη της πολυπλοκότητας.

5.1 Βασικές ιδέες και στόχος

Ας δείξουμε τι εννοούμε με ένα παράδειγμα. Είδαμε ότι η υπολογιστική ισχύς των ντετερμινιστικών πεπερασμένων αυτομάτων (1DFAs) δεν αυξάνεται εάν επιχειρήσουμε να τα ενισχύσουμε με την επιπρόσθετη δυνατότητα του μη ντετερμινισμού και/ή της κίνησης σε δύο κατευθύνσεις. Είτε πρόκειται για μη ντετερμινιστικά μίας κατεύθυνσης (1NFAs), ντετερμινιστικά δύο κατευθύνσεων (2DFAs) ή ακόμα και μη ντετερμινιστικά δύο κατευθύνσεων (2NFAs), τα πεπερασμένα αυτόματα παραμένουν ανήμπορα μπροστά σε μία μη κανονική γλώσσα. Παρά το γεγονός αυτό όμως, οι επιπλέον ιδιότητες όντως βελτιώνουν τα 1DFAs κατά μία έννοια: Δεδομένου ενός προβλήμα-

τος, τα ενισχυμένα αυτόματα κάποιες φορές κατορθώνουν να μείνουν αισθητά μικρότερα, δηλαδή με αρκετά μικρότερο πλήθος καταστάσεων, από τα 1DFAs.

Η παραπάνω παρατήρηση έδωσε το έναυσμα για μια πολύ πιο γενική και συστηματική έρευνα: Όταν μετατρέπουμε μια μηχανή κάποιου τύπου σε μία ισοδύναμη μηχανή ενός άλλου τύπου, πόσο πιο «μεγάλη»[1] πρέπει να είναι η νέα μηχανή; Το ερώτημα δεν είναι καθόλου απλό, αφού ακόμα και ο μικρός κόσμος των κανονικών γλωσσών επιφυλάσσει εκπλήξεις και ανοικτά προβλήματα.

Στο κεφάλαιο αυτό λοιπόν ασχολούμαστε με την ευρεία περιοχή της **πολυπλοκότητας** (descriptional complexity), ή, όπως αλλιώς καλείται, **περιεκτικότητας** (succinctness, conciseness) ή **οικονομίας στην περιγραφή** (economy of description). Όλοι οι παραπάνω όροι αναφέρονται στην επιστήμη—ή τέχνη, ίσως—του να περιγράφεται κάτι με όσο πιο απλό τρόπο γίνεται, και μόνο τόσο περίπλοκα, όσο είναι απαραίτητο. Ειδικότερα, ανάλογα με τις περιστάσεις, οι υπολογιστικοί πόροι για την περιγραφή, όπως ο μη ντετερμινισμός, η δυνατότητα κίνησης της κεφαλής σε δύο κατευθύνσεις (bidirectionality), η τυχαιότητα (randomness) στους υπολογισμούς, η αμφισημία (ambiguity)[2] ή η πρόβλεψη (lookahead), μπορούν να χρησιμοποιηθούν κάποιες φορές σε πλήρη έκταση, κάποιες άλλες καθόλου, και τις περισσότερες μόνο σε ένα βαθμό. Αυτό μας φέρνει στην περιοχή της **πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους** (descriptional complexity of machines with limited resources).

Και για να περιγράψουμε το θέμα μας με ακόμη περισσότερη ακρίβεια, το επίκεντρό μας θα είναι οι κλιμακωτές ανταλλαγές ή διαφορές (trade-offs) ανάμεσα σε διάφορους υπολογιστικούς πόρους, που επέρχονται όταν κάποιοι από τους πόρους αυτούς περιορίζονται. Σε τελική ανάλυση, τα ερωτήματα που μας απασχολούν είναι του τύπου: *Τι κόστος έχει ο περιορισμός του A υπολογιστικού πόρου σε σχέση με τον B, και, συγκεκριμένα, ποιο είναι το άνω και κάτω φράγμα του κόστους;*

Έτσι, δεδομένων δύο διαφορετικών συστημάτων περιγραφής[3] μιας κλάσης γλωσσών, και δεδομένου ενός μέτρου πολυπλοκότητας[4], ερευνούμε το άνω και κάτω φράγμα για την αύξηση (blow-up) ή μείωση (savings) της πολυπλοκότητας, όταν περνούμε από το ένα σύστημα περιγραφής στο άλλο. Ασφαλώς

[1] Αναφερόμαστε πάντα στο μέγεθος του συνόλου καταστάσεων.

[2] Δεν πρέπει να γίνει σύγχυση με την αμφισημία γραμματικών. Η έννοια της αμφισημίας εδώ αναφέρεται σε αυτόματα και θα μελετηθεί στη συνέχεια.

[3] Τέτοια μπορεί να είναι τυπικές γραμματικές, κανονικές εκφράσεις, συστήματα επανεγγραφής (rewriting systems), συστήματα γραμματικών (grammar systems) και πολλά άλλα, όμως εδώ ενδιαφερόμαστε αποκλειστικά για *κλάσεις μηχανών*.

[4] Συγκεκριμένα ενδιαφερόμαστε για *το μέγεθος της μηχανής*, που θεωρούμε πάντα ταυτόσημο με το πλήθος των καταστάσεών της.

είναι πιθανό το ενδεχόμενο να μην υπάρχει αναδρομική συνάρτηση που να υπηρετεί ως φράγμα για τη διαφορά στην πολυπλοκότητα ανάμεσα σε δύο συστήματα περιγραφής, και τότε λέμε ότι η διαφορά είναι μη αναδρομική. Καθώς κάθε λογικό μέτρο πολυπλοκότητας μπορεί να υποτεθεί ότι φράσσεται από μία αναδρομική συνάρτηση, οι μη αναδρομικές διαφορές παρουσιάζουν το ενδιαφέρον χαρακτηριστικό ότι δεν εξαρτώνται από συγκεκριμένα (λογικά) μέτρα πολυπλοκότητας.

Στις επόμενες ενότητες συγκεντρώνουμε ορισμένα από τα σημαντικότερα γνωστά αποτελέσματα του πολύπτυχου χώρου της πολυπλοκότητας μηχανών με περιορισμένους πόρους.

5.2 Πεπερασμένα αυτόματα και περιορισμένος μη ντετερμινισμός

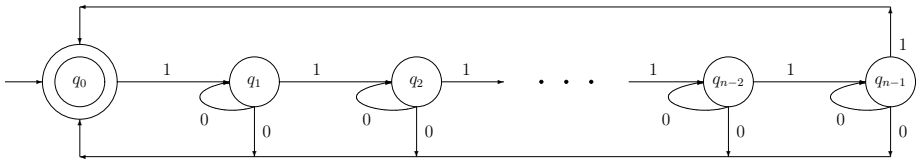
Στην ενότητα αυτή θα ερευνήσουμε πώς η πολυπλοκότητα των πεπερασμένων αυτομάτων ποικίλλει, ανάλογα με το βαθμό στον οποίο αυτά χρησιμοποιούν το μη ντετερμινισμό. Θα εισάγουμε για το σκοπό αυτό την έννοια του *φάσματος* μιας κανονικής γλώσσας, που λειτουργεί ως ένδειξη για το πόσο «εγγενώς μη ντετερμινιστική» είναι η γλώσσα. Θα ασχοληθούμε, ακόμα, με δύο ειδικές περιπτώσεις πεπερασμένων αυτομάτων: τα πεπερασμένα αυτόματα με πολλαπλές αρχικές καταστάσεις και τα πεπερασμένα αυτόματα Las Vegas.

5.2.1 Προς πεπερασμένο μη ντετερμινισμό

Όπως έχουμε ήδη αναφέρει, η πολυπλοκότητα, δηλαδή το μέγεθος, ενός ελαχίστου ντετερμινιστικού πεπερασμένου αυτομάτου είναι δυνατόν να απέχει δραματικά από εκείνη ενός ελαχίστου μη ντετερμινιστικού που δέχεται την ίδια γλώσσα. Για την ακρίβεια, η χρήση του μη ντετερμινισμού στα πεπερασμένα αυτόματα κάποιες φορές επιτρέπει εκθετικά συνοπτικότερες αναπαραστάσεις γλωσσών.

Στο δημοσίευμα–ορόσημο [14] του 1971, με το οποίο γεννιέται το ενδιαφέρον για την περιοχή που αποτελεί το όλο θέμα αυτού του κεφαλαίου, αποδεικνύεται, μεταξύ άλλων, η ύπαρξη μιας άπειρης ακολουθίας γλωσσών $(L_n)_{n\geq 1}$, τέτοιας ώστε για κάθε $n \geq 1$ υπάρχει ένα NFA M_n μεγέθους n που δέχεται την L_n, ενώ κάθε ισοδύναμο DFA έχει μέγεθος τουλάχιστον 2^n. Το NFA M_n που δέχεται την L_n απεικονίζεται στο Σχήμα 5.1.

Με άλλα λόγια, δείχνεται ότι η μέθοδος κατασκευής υποσυνόλου που σκιαγραφήσαμε στην Ενότητα 2.2.2 και μετατρέπει ένα NFA μεγέθους n σε ισοδύναμο DFA μεγέθους 2^n, δεν επιδέχεται βελτίωση. Παρά το παραπάνω γεγονός



Σχήμα 5.1: Το NFA M_n με n καταστάσεις. Ένα ισοδύναμο DFA χρειάζεται τουλάχιστον 2^n καταστάσεις.

όμως, δε φαίνεται να κάνουν όλα τα μη ντετερμινιστικά πεπερασμένα αυτόματα ίδια χρήση του μη ντετερμινισμού. Στην ουσία, ο μη ντετερμινισμός στα πεπερασμένα αυτόματα είναι ένας πόρος που μπορεί να χρησιμοποιηθεί σε διαφορετικά ποσά, καθώς και να προσδιοριστεί ποσοτικά με ακρίβεια.

Ένας προφανής τρόπος να μετρήσουμε την ποσότητα του μη ντετερμινισμού που ένα αυτόματο χρησιμοποιεί *για έναν υπολογισμό* είναι υπολογίζοντας το πλήθος των μη ντετερμινιστικών κινήσεων που εκτελούνται κατά τον υπολογισμό. Θεωρώντας την «καλύτερη προσπάθεια» του αυτομάτου, η ποσότητα του μη ντετερμινισμού που χρησιμοποιείται *για την αποδοχή μιας εισόδου* w μετριέται με την ελάχιστη ποσότητα μη ντετερμινισμού που χρησιμοποιείται σε έναν υπολογισμό που οδηγεί στην αποδοχή της w. Τέλος, ο μη ντετερμινισμός που χρησιμοποιεί το NFA M θεωρείται περιορισμένος (limited), ή πεπερασμένος (finite), εάν υπάρχει ακέραιος c, τέτοιος ώστε για κάθε λέξη w που γίνεται αποδεκτή από το M, υπάρχει υπολογισμός του M που οδηγεί στην αποδοχή της w με το πολύ c μη ντετερμινιστικές κινήσεις.

Το χρήσιμο να εξεταστεί ερώτημα είναι εάν η χρήση—έστω και βαριά—περιορισμένων ποσοτήτων μη ντετερμινισμού μπορεί επίσης να μειώσει το μέγεθος των DFAs. Το βασικό αποτέλεσμα είναι ότι υπάρχουν περιπτώσεις, στις οποίες ακόμα και εξαιρετικά περιορισμένος μη ντετερμινισμός πράγματι βοηθά. Όμως ο βαθμός στον οποίο μπορεί να συμβεί κάτι τέτοιο είναι φραγμένος, καθώς υπάρχουν γλώσσες για τις οποίες κάθε NFA περιορισμένου μη ντετερμινισμού που τις δέχεται έχει το ίδιο μέγεθος με ένα ισοδύναμο DFA.

5.2.2 Φάσματα κανονικών γλωσσών

Ο τρόπος μέτρησης του μη ντετερμινισμού που εισήχθη στην προηγούμενη υποενότητα δεν είναι και τόσο εκλεπτυσμένος, καθώς αδιαφορεί παντελώς για την πληθικότητα του συνόλου των καταστάσεων που διαδέχονται άμεσα κάθε κατάσταση. Αρκεί να αναλογιστούμε ότι στην πραγματικότητα δεν υπάρχει διαφορά στην ποσότητα του μη ντετερμινισμού ενός υπολογισμού με δύο μη

ντετερμινιστικές κινήσεις, καθεμία από τις οποίες οδηγεί σε κατάσταση με δύο δυνατές επόμενες καταστάσεις, και ενός υπολογισμού με μία μόνο μη ντετερμινιστική κίνηση, που όμως οδηγεί σε κατάσταση με τέσσερις δυνατές επόμενες καταστάσεις. Έτσι κρίνεται απαραίτητη η υιοθέτηση κάπως πιο σύνθετων μέτρων, όπως η διακλάδωση, το μάντεμα και το φάσμα, που όλα εισάγονται στο [3].

Έστω $M = (Q, \Sigma, \delta, q_0, F)$ ένα πεπερασμένο αυτόματο. Η **διακλάδωση** (branching) και το **μάντεμα** (guessing) μιας *κίνησης* (p, a, q) είναι $\beta_M(p, a, q) = |\delta(p, a)|$ και $\gamma_M(p, a, q) = \log_2(|\delta(p, a)|)$, αντίστοιχα. Η διακλάδωση και το μάντεμα ενός *υπολογισμού* $\mu_1 \dots \mu_r$ του M θα είναι $\beta_M(\mu_1 \dots \mu_r) = \beta_M(\mu_1) \cdot \dots \cdot \beta_M(\mu_r)$ και $\gamma_M(\mu_1 \dots \mu_r) = \gamma_M(\mu_1) + \dots + \gamma_M(\mu_r)$, αντίστοιχα. Ακόμα, η διακλάδωση και το μάντεμα μιας *λέξης* $w \in L(M) - \{\varepsilon\}$ ορίζονται ως $\beta_M(w) = min\{\beta_M(\mu_1 \dots \mu_r) \mid \mu_1 \dots \mu_r$ υπολογισμός που οδηγεί σε αποδοχή της $w\}$ και $\gamma_M(w) = min\{\gamma_M(\mu_1 \dots \mu_r) \mid \mu_1 \dots \mu_r$ υπολογισμός που οδηγεί σε αποδοχή της $w\}$, αντίστοιχα, ενώ $\beta_M(\varepsilon) = 1$ και $\gamma_M(\varepsilon) = 0$, αν $\varepsilon \in L(M)$. Τέλος, ως διακλάδωση και μάντεμα του *αυτομάτου* M ορίζουμε $\beta_M = sup\{\beta_M(w) \mid w \in L(M)\}$ και $\gamma_M = sup\{\gamma_M(w) \mid w \in L(M)\}$, αντίστοιχα, ενώ $\beta_M = 1$ και $\gamma_M = 0$ στην περίπτωση που $L(M) = \varnothing$.

Έτσι, για μια λέξη w και έναν υπολογισμό $\mu_1 \dots \mu_r$ για την w, το μέγεθος της διακλάδωσης $\beta_M(\mu_1 \dots \mu_r)$ κατά κάποιο τρόπο αντανακλά το ποσό παραλληλισμού (parallelism) που απαιτείται για μια προσομοίωση του $\mu_1 \dots \mu_r$ σε πραγματικό χρόνο—δηλαδή το πλήθος των διαδικασιών που θα εκτελούνται ταυτόχρονα—ενώ $1/\beta_M(\mu_1 \dots \mu_r)$ είναι η πιθανότητα να επιλέξει το M το συγκεκριμένο υπολογισμό $\mu_1 \dots \mu_r$ για την w. Συνεπώς, το μέγεθος της διακλάδωσης του αυτομάτου β_M διαισθητικά μας δίνει την ελάχιστη ποσότητα μη ντετερμινισμού που χρειάζεται το M για να δεχτεί όλες τις λέξεις της $L(M)$.

Από την άλλη, για μια κίνηση μ, το μάντεμα $\gamma_M(\mu)$ είναι ίσο με το πλήθος των bits πληροφορίας που απαιτούνται για να ξεχωρίσουμε και να καταγράψουμε τη μ ανάμεσα στις άλλες κινήσεις που το M θα μπορούσε να έχει επιλέξει σε κάθε στάδιο ενός υπολογισμού όπου η μ εκτελείται. Αν $|\delta(p, a)| \leq 2$ για κάθε $p \in Q$ και $a \in \Sigma$, το $\gamma_M(\mu_1 \dots \mu_r)$ ενός υπολογισμού $\mu_1 \dots \mu_r$ μετρά ακριβώς πόσες από τις $\mu_1, \dots, \mu_r$ είναι μη ντετερμινιστικές κινήσεις. Καθώς όμως το μάντεμα, αν και το πιο φυσικό μέτρο, δεν προκύπτει πάντοτε ακέραιος αριθμός, φαίνεται τεχνικά πιο βολική η χρήση της διακλάδωσης.

Κτίζοντας πάνω στα προηγούμενα, μπορούμε τώρα να προχωρήσουμε στην έννοια του φάσματος μιας κανονικής γλώσσας. Έστω L μια κανονική γλώσσα. Ως **φάσμα** (spectrum) της L ορίζεται η άπειρη ακολουθία $\sigma(L) = (\sigma_1(L), \sigma_2(L), \dots)$, όπου $\sigma_i(L)$ είναι το ελάχιστο μέγεθος ενός NFA που δέχεται την L με διακλάδωση το πολύ i. Έτσι, το φάσμα $\sigma(L)$ δείχνει πόσο «εγγενώς μη ντετερμινιστική» η L είναι, με την έννοια ότι δίνει το μέγεθος των ελαχίστων

NFAs που δέχονται την L για κάθε δυνατό ποσό μη ντετερμινισμού· πράγματι, το $\sigma_1(L)$ δίνει το ελάχιστο μέγεθος ενός DFA που τη δέχεται, ενώ, στο άλλο άκρο, το $\sigma_\infty(L)$ αντανακλά το ελάχιστο μέγεθος που απαιτείται σε ένα NFA, ώστε να τη δεχτεί με τυχαία ποσότητα μη ντετερμινισμού.

Το ενδιαφέρον φυσικά εστιάζεται στο ερώτημα του *πόσο χαμηλές μπορεί να είναι οι μεσαίες τιμές ενός φάσματος*. Αποδεικνύεται ότι, αν και άπειρη ακολουθία ακεραίων, το φάσμα είναι υπολογίσιμο, μονοτονικό, καθώς και φραγμένο. Έτσι, όπως ήδη αναφέραμε, βρίσκεται ότι υπάρχουν γλώσσες για τις οποίες ο περιορισμένος μη ντετερμινισμός δεν οδηγεί σε καμία μείωση του πλήθους των καταστάσεων που απαιτούνται για την περιγραφή της γλώσσας. Μάλιστα υπάρχουν περιπτώσεις όπου τα NFAs με περιορισμένο μόνο μη ντετερμινισμό είναι τόσο μεγάλα, όσο και τα ισοδύναμα DFAs, ενώ, αντίθετα, η χρήση τυχαίας ποσότητας μη ντετερμινισμού οδηγεί σε εκθετικά μικρότερα αυτόματα.

Αντιδιαμετρικά, εξίσου υπαρκτές είναι και οι περιπτώσεις στις οποίες πολύ μικρές ποσότητες μη ντετερμινισμού έχουν ως αποτέλεσμα ουσιαστική μείωση του μεγέθους. Μάλιστα, με τη βοήθεια τεχνικών από την επικοινωνιακή πολυπλοκότητα (communication complexity), δείχθηκε πρόσφατα η ύπαρξη γλωσσών, στις οποίες ο πεπερασμένος μη ντετερμινισμός βοηθά σταδιακά, αλλά μόνο μετά από ένα συγκεκριμένο σημείο, δηλαδή αφού έχει εξασφαλιστεί ένα συγκεκριμένο ποσό μη ντετερμινισμού.

5.2.3 Πεπερασμένα αυτόματα με πολλαπλές αρχικές καταστάσεις

Ένας άλλος αποδοτικός τρόπος να περιορίσουμε το μη ντετερμινισμό σε ένα αυτόματο είναι να μειώσουμε τα σημεία στα οποία ο μη ντετερμινισμός υπεισέρχεται κατά τη διάρκεια ενός υπολογισμού. Στην περίπτωση που επιτρέψουμε μόνο *μία* μη ντετερμινιστική κίνηση, και αυτή προτού καν ξεκινήσει ουσιαστικά ο υπολογισμός, μιλάμε για τα MDFAs, τα πεπερασμένα αυτόματα με πολλαπλές αρχικές καταστάσεις και ντετερμινιστική συνάρτηση μετάβασης. Σε ένα τέτοιο αυτόματο η διακλάδωση υπολογίζεται από το πλήθος των αρχικών καταστάσεων κι έτσι είναι πάντα πεπερασμένη. Άλλωστε κάθε MDFA μπορεί να μετασχηματιστεί σε ισοδύναμο NFA με πεπερασμένο μη ντετερμινισμό.

Αποδεικνύεται μέσω της μεθόδου κατασκευής υποσυνόλου ότι για κάθε MDFA με n καταστάσεις και k αρχικές καταστάσεις υπάρχει ισοδύναμο DFA με το πολύ $\sum_{1\leq i\leq k}\binom{n}{i}$ καταστάσεις. Ακόμα, για κάθε NFA με n καταστάσεις και διακλάδωση k, υπάρχει ισοδύναμο MDFA με $kn+1$ καταστάσεις και k αρχικές καταστάσεις. Με άλλα λόγια, ο μη ντετερμινισμός σε ένα NFA με

πεπερασμένη διακλάδωση μπορεί να μετατοπιστεί από τη συνάρτηση μετάβασης στις αρχικές καταστάσεις.

5.2.4 Πεπερασμένα αυτόματα Las Vegas

Ένας αρκετά ενδιαφέρων τύπος αυτομάτων, στον οποίο ο μη ντετερμινισμός χρησιμοποιείται με έναν ιδιαίτερο τρόπο, είναι τα **αυτοπιστοποιούμενα μη ντετερμινιστικά πεπερασμένα αυτόματα** (self-verifying nondeterministic finite automata) SNFAs. Αυτά είναι NFAs, των οποίων το σύνολο των καταστάσεων διαμερίζεται σε τρία ξένα υποσύνολα: τις καταστάσεις αποδοχής (accepting states), τις καταστάσεις απόρριψης (rejecting states) και τις ουδέτερες καταστάσεις (neutral states). Έτσι, μία είσοδος θα γίνεται αποδεκτή ή θα απορρίπτεται από ένα SNFA, εάν υπάρχει υπολογισμός που οδηγεί σε κατάσταση αποδοχής ή απόρριψης, αντίστοιχα. Επιπλέον, για κάθε είσοδο υπάρχει τουλάχιστον ένας υπολογισμός που να οδηγεί είτε σε αποδοχή είτε σε απόρριψη, ενώ δεν είναι δυνατόν για κάποια είσοδο να υπάρχει και υπολογισμός που οδηγεί σε αποδοχή και υπολογισμός που οδηγεί σε απόρριψη.

Το **πεπερασμένο αυτόματο Las Vegas** (Las Vegas finite automaton) LVFA, τώρα, ορίζεται ως ένα SNFA A, στο οποίο για κάθε $x \in L(A)$ υπάρχει υπολογισμός που οδηγεί σε αποδοχή με πιθανότητα τουλάχιστον $\frac{1}{2}$, και για κάθε $x \notin L(A)$ υπάρχει υπολογισμός που οδηγεί σε απόρριψη με πιθανότητα τουλάχιστον $\frac{1}{2}$. Η πιθανότητες αυτές υπολογίζονται ως εξής: Για κάθε κατάσταση q του A και κάθε σύμβολο a του αλφαβήτου εισόδου, θεωρούμε μία τυχαία κατανομή πιθανοτήτων επί του συνόλου των κινήσεων που ξεκινούν από την q με ανάγνωση του a. Η πιθανότητα ενός υπολογισμού του A είναι τώρα το γινόμενο των πιθανοτήτων όλων των κινήσεων που έχουν συμπεριληφθεί στον υπολογισμό.

Στο [8], τα αποτελέσματα του οποίου έχουν γενικευτεί και για πεπερασμένα αυτόματα Las Vegas *δύο κατευθύνσεων*, αλλά και στο χώρο της *κβαντοεπικοινωνίας* (quantum communication), δείχνεται ότι η πολυπλοκότητα των LVFAs που δέχονται μία γλώσσα L είναι τουλάχιστον ίση με την τετραγωνική ρίζα της πολυπλοκότητας των ελαχίστων DFAs που δέχονται την L, ενώ μέσω μιας συγκεκριμένης κανονικής γλώσσας πιστοποιείται ότι το κάτω αυτό φράγμα είναι βέλτιστο. Γενικότερα, η επικρατούσα εικασία είναι ότι το κόστος των υπολογισμών Las Vegas είναι πλησιέστερα σε εκείνο των ντετερμινιστικών, παρά των μη ντετερμινιστικών υπολογισμών.

5.3 Πεπερασμένα αυτόματα και περιορισμένη αμφισημία

Στην προηγούμενη ενότητα μελετήσαμε τις διαφορές στην πολυπλοκότητα ανάμεσα σε NFAs με διαφορετικά ποσά μη ντετερμινισμού. Για να μετρήσουμε αυτά τα ποσά, λάβαμε υπόψη μεταξύ των υπολογισμών κάθε αυτομάτου που οδηγούν σε αποδοχή, μόνο εκείνους με το μικρότερο πλήθος μη ντετερμινιστικών κινήσεων. Ένα άλλο μέτρο, η **αμφισημία** (ambiguity), λειτουργεί θεωρώντας όλους τους υπολογισμούς που οδηγούν σε αποδοχή.

Ένα NFA καλείται **k-διφορούμενο** (k-ambiguous) αν δέχεται κάθε λέξη της γλώσσας του με το πολύ k διαφορετικούς υπολογισμούς. Τα 1-διφορούμενα NFAs καλούνται **μη διφορούμενα** (unambiguous, UFAs) και ναι μεν επιτρέπεται να χρησιμοποιήσουν μη ντετερμινισμό, αλλά, όπως και τα DFAs, δέχονται λέξεις κατά μοναδικό τρόπο. **Πεπερασμένα διφορούμενο** (finitely ambiguous, FNA) καλείται ένα NFA εάν είναι k-διφορούμενο για κάποιο θετικό ακέραιο k, ενώ **πολυωνυμικά διφορούμενο** (polynomially ambiguous, PNA) εάν υπάρχει ένα πολυώνυμο p, τέτοιο ώστε για κάθε λέξη x της γλώσσας, υπάρχουν το πολύ $p(|x|)$ υπολογισμοί που οδηγούν σε αποδοχή. Είναι φανερό ότι, αφού, δεδομένου ενός NFA $M = (Q, \Sigma, \delta, q_0, F)$, για μία λέξη μήκους n μπορεί να υπάρχουν το πολύ $|Q|^n$ διαφορετικοί υπολογισμοί που οδηγούν στην αποδοχή της, κάθε NFA είναι εκθετικά διφορούμενο (exponentially ambiguous).

Αντικείμενο εκτεταμένης μελέτης έχουν αποτελέσει οι διαφορές στην πολυπλοκότητα ανάμεσα στα DFAs, UFAs, NFAs, FNAs και PNAs, πολλές από τις οποίες είναι εκθετικές, όπως για πρώτη φορά δείχνεται στο [17]. Κάποιες άλλες όμως, όπως η διαφορά στην πολυπλοκότητα ανάμεσα στα FNAs και PNAs, παραμένουν ανοικτά προβλήματα ή έχουν εξιχνιαστεί εν μέρει. Θέμα σύγχρονης έρευνας αποτελεί, σε αντιστοιχία με την έννοια του φάσματος που αναπτύχθηκε, το ερώτημα του πώς μεταβάλλεται η πολυπλοκότητα των NFAs καθώς αυξάνεται το μέγεθος, όχι του μη ντετερμινισμού αυτή τη φορά, αλλά, της αμφισημίας.

5.4 Πεπερασμένα αυτόματα δύο κατευθύνσεων και περιορισμένοι πόροι

Σε αυτή την ενότητα θα ασχοληθούμε με τα πεπερασμένα αυτόματα δύο κατευθύνσεων, που παρουσιάσαμε στην Ενότητα 2.2.3. Εκτός από τις διαφορές στην πολυπλοκότητα ανάμεσα στα κλασικά 2NFAs, 2DFAs, NFAs και DFAs, θα ερευνήσουμε και κάποια μοντέλα περιορισμένων δυνατοτήτων. Το

ζήτημα της πολυπλοκότητας των πεπερασμένων αυτομάτων δύο κατευθύνσεων αποτελεί σήμερα πολύ σημαντικό πεδίο έρευνας, και, όπως όλα δείχνουν, θα αποτελέσει και στο μέλλον, καθώς μεταξύ άλλων σχετίζεται και με το ανοικτό πρόβλημα της υπολογιστικής πολυπλοκότητας, εάν ο λογαριθμικός χώρος (logarithmic space, L) περιέχεται γνήσια στο μη ντετερμινιστικό λογαριθμικό χώρο (nondeterministic logarithmic space, NL).

Έχει ήδη γίνει σαφές από την Ενότητα 2.2.3 ότι τόσο τα 2DFAs, όσο και τα 2NFAs δε δέχονται παρά τις κανονικές γλώσσες. Υπάρχει πληθώρα αποτελεσμάτων *προσομοίωσης*, που δείχνουν πώς ένα μοντέλο πεπερασμένου αυτομάτου δύο κατευθύνσεων μπορεί να προσομοιωθεί από άλλα μοντέλα πεπερασμένων αυτομάτων· αυτά συμπεριλαμβάνουν τις μετατροπές από 2DFA σε DFA, από 2DFA σε NFA, από 2NFA σε DFA και από 2NFA σε NFA. Παρ' όλα αυτά, υπάρχουν ελάχιστα μόνο αποτελέσματα *διαχωρισμού*.

Κατά τη σύγκριση δύο διαφορετικών μοντέλων ως προς την πολυπλοκότητα, θεωρούμε τα δύο μοντέλα **διαχωρισμένα** (separated), όταν για την αποδοχή της ίδιας ακολουθίας γλωσσών, το ένα παραμένει εκθετικά μικρότερου μεγέθους από το άλλο. Σχετικά με την πολυπλοκότητα διαφορετικών μοντέλων πεπερασμένων αυτομάτων δύο κατευθύνσεων, το βασικό πρόβλημα είναι εάν τα 2NFAs μπορούν να διαχωριστούν από τα 2DFAs, εάν, δηλαδή, υπάρχει οικογένεια κανονικών γλωσσών L_n, τέτοια ώστε για να την περιγράψει ένα 2DFA χρειάζεται εκθετικό πλήθος καταστάσεων, ενώ ένα 2NFA αρκείται σε πολυωνυμικό.

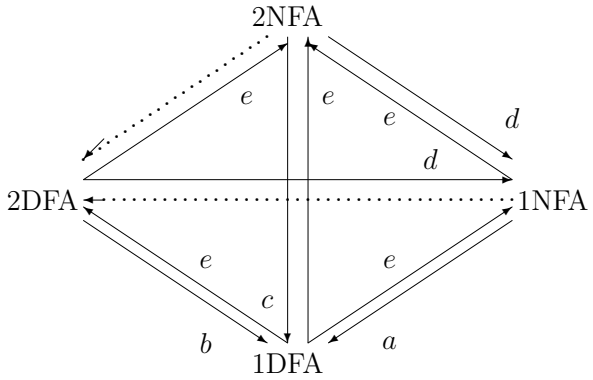
Το διάσημο αυτό πρόβλημα έχει μείνει ανοικτό εδώ και πολύ καιρό, και, αν και πιστεύεται ευρέως ότι η διαφορά στην πολυπλοκότητα ανάμεσα στα 2NFAs και 2DFAs είναι όντως εκθετική, αυτό φαίνεται πολύ δύσκολο να τεκμηριωθεί. Στην πραγματικότητα προκαλεί έκπληξη το πόσα λίγα ξέρουμε για το μετασχηματισμό από 2NFA σε 2DFA. Όχι μόνο δε γνωρίζουμε την ακριβή τιμή της διαφοράς στην πολυπλοκότητα, αλλά ούτε καν μπορούμε να αποφανθούμε εάν είναι πολυωνυμική. Το καλύτερο γνωστό άνω φράγμα είναι εκθετικό και το καλύτερο γνωστό κάτω φράγμα κυβικό. Η επικρατούσα εικασία είναι ότι η διαφορά στην πολυπλοκότητα ανάμεσα στα 2NFAs και 2DFAs είναι ίση με το καλύτερο γνωστό άνω φράγμα, και συγκεκριμένα, ότι ένα 2DFA απαιτεί μέγεθος τουλάχιστον $2^n - 1$ για να δεχτεί μία γλώσσα που ένα 2NFA δέχεται με μέγεθος n.

Αρκετή προσπάθεια έχει γίνει για την απόδειξη της εικασίας για διάφορες παραλλαγές των 2DFAs με κάποιους περιορισμούς στους πόρους. Ανάμεσα στα γνωστά αποτελέσματα είναι ότι η εικασία ισχύει για τα 2DFAs που είναι *single-pass* (τερματίζουν μόλις φτάσουν σε κάποιο άκρο της εισόδου), *sweeping* (μπορούν να αλλάξουν κατεύθυνση μόνο όταν φτάσουν σε κάποιο άκρο της εισόδου) και *oblivious* (λειτουργούν πανομοιότυπα για όλες τις εισόδους

του ίδιου μήκους).

Ο Χρήστος Καπούτσης στο [10] εξετάζει (και επαληθεύει) την εικασία εστιάζοντας στην ειδική περίπτωση των 2DFAs που είναι *moles*. Η κλάση των moles είναι μία περιορισμένη κλάση αυτομάτων που συμπεριλαμβάνει στα μέλη της τα NFAs πολυωνυμικού μεγέθους, που δέχονται τις γλώσσες που ανήκουν σε μία ειδική οικογένεια γλωσσών, τη *liveness*. Η liveness είναι η δυσκολότερη ανάμεσα στις οικογένειες γλωσσών που αναγνωρίζονται από πολυωνυμικές οικογένειες NFAs, κι έτσι η εικασία αποκτά την ισοδύναμη μορφή, ότι καμία οικογένεια 2DFAs πολυωνυμικού μεγέθους δεν μπορεί να αναγνωρίσει τη liveness. Στην εργασία του Χ. Καπούτση δείχνεται αντ' αυτού ότι καμία οικογένεια από 2DFAs που είναι moles, ανεξαρτήτως μεγέθους, δεν μπορεί να αναγνωρίσει τη liveness.

Στο Σχήμα 5.2 παρουσιάζουμε τους δώδεκα μετασχηματισμούς που ορίζονται ανάμεσα στους τέσσερις βασικούς τύπους αυτομάτων που μελετούμε, καθώς και τις ακριβείς τιμές της διαφοράς ανάμεσα στις πολυπλοκότητές τους, όπου είναι γνωστές. Τα διακεκομμένα βέλη υποδεικνύουν τα ανοικτά προβλήματα.



Σχήμα 5.2: Οι δώδεκα μετασχηματισμοί που ορίζονται από το μη ντετερμινισμό και τη διπλή κατεύθυνση, καθώς και οι γνωστές διαφορές στην πολυπλοκότητα: $a = 2^n - 1$, $b = n\big(n^n - (n-1)^n\big)$, $c = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \binom{n}{i}\binom{n}{j}(2^i-1)^j$, $d = \binom{2n}{n+1}$, $e = n$.

5.5 Περιορίζοντας το μη ντετερμινισμό και την αμφισημία στα αυτόματα στοίβας

Εδώ θα στραφούμε προς τα αυτόματα στοίβας και τα αποτελέσματα που επιφέρει στην πολυπλοκότητά τους ο περιορισμός του μη ντετερμινισμού και της αμφισημίας. Βέβαια, προκειμένου να μειώσουμε το μη ντετερμινισμό, πρέπει πάλι πρώτα να συμφωνήσουμε ως προς ένα τρόπο να τον μετρούμε. Θα εξετάσουμε διάφορους τέτοιους τρόπους. Ακόμα, θα ερευνήσουμε την αυξομείωση της αμφισημίας και θα δούμε ότι μπορεί να έχει αξιοθαύμαστες επιπτώσεις στην πολυπλοκότητα των αυτομάτων στοίβας. Όπως αποδεικνύεται, η αύξηση της αμφισημίας ενός PDA (ή, όπως θα δούμε, της αντίστοιχης CFG) κατά ένα μόνο βαθμό είναι δυνατό να επιφέρει μη αναδρομική μείωση στην πολυπλοκότητά του.

5.5.1 Μετρώντας το μη ντετερμινισμό στα αυτόματα στοίβας

Κατ' αρχάς πρέπει να παρατηρήσουμε ότι, σε αντίθεση με τα πεπερασμένα αυτόματα και τις μηχανές Turing, δεν είναι πάντοτε δυνατό να αφαιρέσουμε το μη ντετερμινισμό από ένα PDA χωρίς να μειώσουμε την υπολογιστική ισχύ του. Έτσι, ένα εύστοχο ερώτημα είναι «τι αποτέλεσμα έχει στο μέγεθος ενός PDA η εξάλειψη του μη ντετερμινισμού, *όταν αυτό είναι δυνατό*», και η απάντηση που δόθηκε ήδη από το 1976 είναι ότι η αύξηση του μεγέθους συχνά είναι μη φραγμένη αναδρομικά (recursively unbounded), δηλαδή μεγαλώνει γρηγορότερα από κάθε αναδρομική συνάρτηση. Είναι λοιπόν φανερό ότι το καίριο ερώτημα δε θα πρέπει να αφορά την πλήρη εξάλειψη του μη ντετερμινισμού, αλλά τον περιορισμό του, και γι' αυτό χρειαζόμαστε τρόπους μέτρησής του.

Πολλοί τρόποι μέτρησης του μη ντετερμινισμού στα PDAs έχουν κατά καιρούς προταθεί, αλλά τρεις είναι οι πιο σημαντικοί, εκ των οποίων περισσότερο έχει καθιερωθεί ένας. Δύο από τα μέτρα είναι δυναμικά, δηλαδή βασίζονται στη συμπεριφορά του αυτομάτου, ενώ το άλλο είναι στατικό και βασίζεται μόνο στη δομή του αυτομάτου. Τα δύο δυναμικά μέτρα είναι γνωστά ως *maxmax* και *minmax*, ενώ το στατικό ως *βάθος* του PDA, και θα αναφερθούμε σε όλα αναλυτικά.

Με βάση το **maxmax** μέτρο, το PDA χρεώνεται με το μέγιστο ποσό μη ντετερμινισμού που χρησιμοποιεί κατά την αποδοχή μιας εισόδου. «Μη επιτυχής» χρήση μη ντετερμινισμού, δηλαδή χρήση μη ντετερμινισμού κατά τη διάρκεια υπολογισμών που δεν οδηγούν σε αποδοχή, δε λαμβάνεται υπόψη. Το μέτρο maxmax, αν και τεχνικά ιδιαίτερα εύχρηστο, έχει αρκετά μειονεκτήματα. Ένα από αυτά είναι και ότι με χρήση ενός τέτοιου μέτρου όλες οι

πεπερασμένες ποσότητες μη ντετερμινισμού προκύπτουν ισοδύναμες, με αποτέλεσμα να αποδίδονται μόνο τρεις μη ισοδύναμες τιμές μη ντετερμινισμού: μηδενική (χαρακτηρίζει τις ντετερμινιστικές γλώσσες χωρίς συμφραζόμενα), πεπερασμένη (χαρακτηρίζει πεπερασμένες ενώσεις ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα) και γραμμική (χαρακτηρίζει όλες τις γλώσσες χωρίς συμφραζόμενα).

Με στόχο να ξεπεραστούν τα προβλήματα που σχετίζονται με το δυναμικό maxmax μέτρο, προτάθηκε το στατικό μέτρο του μη ντετερμινιστικού **βάθους** (depth) ενός PDA, που, σε αντίθεση με το maxmax, παράγει μια άπειρη ιεραρχία από κλάσεις πολυπλοκότητας των γλωσσών χωρίς συμφραζόμενα. Η χρησιμότητα του μέτρου αυτού γίνεται κατανοητή αν θεωρήσουμε τις γλώσσες χωρίς συμφραζόμενα

$$L_1 = \{ww^R \mid w \in \{0,1\}^*\},$$
$$L_2 = \{0^i 1^j 2 \mid j = i \quad \acute{\eta} \quad j = 2i\},$$

που κατά το maxmax μέτρο απαιτούν και οι δύο γραμμική ποσότητα μη ντετερμινισμού. Ωστόσο, κατά μία έννοια, η L_1 δείχνει να εμπλέκεται σε λιγότερο μη ντετερμινισμό από την L_2, αφού η L_1 μπορεί να γίνει δεκτή από ένα PDA που ουσιαστικά αποτελείται από δύο μη ντετερμινιστικά κομμάτια (ένα για εισαγωγή και ένα για εξαγωγή των συμβόλων της w από τη στοίβα), με ένα μη ντετερμινιστικό *άλμα* (jump) ανάμεσά τους. Με το μέτρο βάθους αυτό το άλμα μετριέται μόνο μία φορά και γι' αυτό λέμε ότι η L_1 έχει μη ντετερμινιστικό βάθος 1. Αντίθετα, ένα PDA που δέχεται την L_2 δεν μπορεί να αποσυντεθεί σε ντετερμινιστικά μέρη, κι έτσι λέμε ότι η L_2 έχει άπειρο μη ντετερμινιστικό βάθος.

Γενικά λοιπόν, ως το βάθος ενός PDA ορίζουμε το μέγιστο πλήθος «αλμάτων» που γίνονται κατά τη μη αναστρέψιμη κίνηση από ένα ντετερμινιστικό κομμάτι του σε ένα άλλο, σε μία βέλτιστη αποσύνθεσή του σε ντετερμινιστικά κομμάτια. Ισοδύναμα, μπορούμε να πούμε ότι το βάθος του PDA είναι ίσο με το πλήθος των δεικτών (markers) που θα πρέπει να προσθέσουμε στις συμβολοσειρές της γλώσσας του για να την κάνουμε ντετερμινιστική. Πράγματι, παρατηρούμε ότι ένας τέτοιος δείκτης στο μέσο κάθε λέξης της L_1 είναι αρκετός για το σκοπό αυτό.

Ένας άλλος τρόπος βελτίωσης του maxmax μέτρου είναι η χρήση του **minmax**, κατά το οποίο, σε αντίθεση με το maxmax, το PDA χρεώνεται μόνο με το ποσό μη ντετερμινισμού που χρησιμοποιεί κατά τον «καλύτερο» υπολογισμό του που οδηγεί στην αποδοχή μιας εισόδου. «Μη επιτυχής» χρήση μη ντετερμινισμού ομοίως αγνοείται.

Είναι γεγονός ότι σε κάποιες περιπτώσεις, όπως στα PDAs με πεπερασμένο μη ντετερμινισμό (δηλαδή αυτά που, ανεξάρτητα από το μήκος της

εισόδου τους, εκτελούν πάντα φραγμένο πλήθος μη ντετερμινιστικών κινήσεων), η διαφορά ανάμεσα στη χρήση του maxmax και του minmax μέτρου είναι ασήμαντη. Όμως γενικά το minmax μέτρο, αν και σαφώς δυσκολότερο στο χειρισμό, αφού απαιτεί την ανίχνευση των καλύτερων προσπαθειών του PDA, είναι προτιμότερο από το maxmax και είναι μάλλον το πιο καθιερωμένο και από τα τρία μέτρα. Εξάλλου, ο τρόπος με τον οποίο λειτουργεί, θεωρώντας την καλύτερη αντί για τη χειρότερη προσπάθεια του αυτομάτου, προσεγγίζει πολύ περισσότερο τον τρόπο της βασικής λειτουργίας του αυτομάτου, δηλαδή της αναγνώρισης μιας γλώσσας. Εκτός αυτού, το minmax μέτρο ανταναλά την ποσότητα παραλληλισμού που απαιτείται για την προσομοίωση του μη ντετερμινισμού. Το μέτρο minmax εξετάστηκε εκτενώς στο [4], όπου και έγινε γνωστό ότι, όπως και το μέτρο βάθους, παράγει μία άπειρη ιεραρχία από κλάσεις πολυπλοκότητας των γλωσσών χωρίς συμφραζόμενα.

5.5.2 Αυτόματα στοίβας και περιορισμένη αμφισημία

Αφού εισηγάγαμε τους τρόπους μέτρησης του μη ντετερμινισμού στα PDAs, θα δώσουμε περιληπτικά κάποια από τα βασικότερα αποτελέσματα σχετικά με την πολυπλοκότητα των PDAs με περιορισμούς στο μη ντετερμινισμό και την αμφισημία. Μέσω των κλασικών τεχνικών προσομοίωσης των PDAs από CFGs και αντίστροφα, προκύπτει ότι τα αποτελέσματα που παρουσιάζονται εδώ για PDAs με περιορισμένη αμφισημία έχουν ισχύ και για CFGs με περιορισμένη αμφισημία.

Όπως και ένα NFA, ένα **PDA** καλείται **k-διφορούμενο** (k-ambiguous) εάν κάθε λέξη της γλώσσας του γίνεται δεκτή με το πολύ k διαφορετικούς τρόπους. **Μη διφορούμενα** (unambiguous, UPDAs) ονομάζονται τα 1-διφορούμενα PDAs. Όμοια με τα NFAs ορίζονται επίσης τα πεπερασμένα διφορούμενα και τα πολυωνυμικά διφορούμενα PDAs. Ονομάζοντας τώρα μία **CFG k-διφορούμενη**, αν κάθε λέξη της γλώσσας που παράγει έχει το πολύ k συντακτικά δέντρα, έχουμε ότι για κάθε k-διφορούμενο PDA υπάρχει ισοδύναμη k-διφορούμενη CFG, και αντίστροφα.

Εστιάζοντας για λίγο στον περιορισμό του μη ντετερμινισμού, γνωρίζουμε ότι δεν υπάρχει αναδρομική συνάρτηση που να φράσσει τη μείωση της πολυπλοκότητας όταν χρησιμοποιούμε μη ντετερμινιστικά αντί για ντετερμινιστικά PDAs. Με άλλα λόγια, για κάθε αναδρομική συνάρτηση είναι δυνατό να βρεθεί μία ντετερμινιστική γλώσσα χωρίς συμφραζόμενα (και για την ακρίβεια, μία άπειρη ακολουθία τέτοιων γλωσσών), τέτοια ώστε η διαφορά ανάμεσα στα μεγέθη του μικρότερου PDA και του μικρότερου DPDA που τη δέχονται να μην μπορεί να φραχθεί από τη συνάρτηση αυτή. Μάλιστα, τα μη ντετερμινιστικά PDAs που χρησιμοποιήθηκαν για να αποδειχθεί το παραπάνω είναι και

μη διφορούμενα.

Σχετικά με τον πόρο της αμφισημίας στα PDAs, οφείλουμε κατ' αρχάς να πούμε ότι παρά την εμφανή σύνδεσή της με τον πόρο του μη ντετερμινισμού, οι δύο έννοιες είναι αρκετά διαφορετικές και η σχέση ανάμεσά τους για την περίπτωση των PDAs δεν είναι τόσο απλή όσο ήταν για εκείνη των NFAs. Έχει αποδειχθεί ότι η μείωση της πολυπλοκότητας όταν περνούμε από μη διφορούμενο σε τυχαίο PDA δεν είναι αναδρομική. Επίσης, όπως ήδη αναφέραμε, για τυχόντα ακέραιο k, αν επιτρέψουμε σε ένα k-διφορούμενο PDA να έχει αμφισημία βαθμού $k+1$, είναι δυνατό να επέλθει μη αναδρομική μείωση στην πολυπλοκότητα.

Τέλος, στο [5], όπου και μελετάται λεπτομερώς η σχέση μη ντετερμινισμού και αμφισημίας από πλευράς πολυπλοκότητας για τα PDAs, τεκμηριώνεται η ύπαρξη ενός ενδιαφέροντα τύπου διπλής άπειρης ιεραρχίας γλωσσών. Συγκεκριμένα, για κάθε k, $k' \in \mathbb{N} \cup \{\infty\}$ με $k \leq k'$, δείχνεται ότι υπάρχει γλώσσα που είναι εγγενώς διφορούμενη με βαθμό k και εγγενώς μη ντετερμινιστική με βαθμό k'.

5.6 Πολυπλοκότητα συντακτικών αναλυτών

Θα ολοκληρώσουμε αυτή την επιλεκτική παρουσίαση αποτελεσμάτων πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους, εξετάζοντας την περίπτωση των συντακτικών αναλυτών, και συγκεκριμένα, τους ορθούς συντακτικούς αναλυτές προθέματος και τους συντακτικούς αναλυτές LL(k) και LR(k). Θα δούμε με ποιους τρόπους μπορούμε να μειώσουμε την πολυπλοκότητά τους.

Όπως έχουμε ήδη πει στην Ενότητα 3.5, οι συντακτικοί αναλυτές διαδραματίζουν αναντικατάστατο ρόλο στη διαδικασία μετατροπής λογισμικού ή πηγαίου κώδικα (source code) σε εκτελέσιμο κώδικα (execution code). Ιδιαίτερα οι **ορθοί συντακτικοί αναλυτές προθέματος** (correct prefix parsers), που είναι συντακτικοί αναλυτές με την ικανότητα να ανιχνεύουν τα λάθη όσο το δυνατόν νωρίτερα, έχουν ερευνηθεί εντατικά λόγω της μεγάλης πρακτικής τους αξίας. Εντούτοις, η δυνατότητα γρήγορης ανίχνευσης των λαθών πληρώνεται συχνά σε πολυπλοκότητα του συντακτικού αναλυτή—κι αυτή είναι επίσης μιά πρακτική πλευρά που πρέπει να ληφθεί υπόψη.

Αποδεικνύεται με χρήση της έννοιας των *scanning* PDAs, στην οποία δε θα επεκταθούμε, η ύπαρξη άπειρης ακολουθίας γλωσσών, για την οποία οι ορθοί συντακτικοί αναλυτές προθέματος είναι εκθετικά μεγαλύτερου μεγέθους από τους μικρότερους απλούς ντετερμινιστικούς συντακτικούς αναλυτές. Αυτή η δραματική διαφορά στην πολυπλοκότητα προωθεί το ερώτημα του πώς θα με-

ταβληθεί το μέγεθος του μικρότερου συντακτικού αναλυτή, αν η ανίχνευση του λάθους καθυστερήσει «λίγο». Ακριβέστερα, *είναι δυνατό να μειώσουμε την καθυστέρηση στην ανίχνευση λάθους βαθμιαία, ενώ ταυτόχρονα να αυξήσουμε το μέγεθος του συντακτικού αναλυτή επίσης μόνο βαθμιαία;*

Για να δοθεί απάντηση, προτάθηκε, βασιζόμενη στην ιδέα του φάσματος κανονικής γλώσσας, η έννοια του **φάσματος χρονοκαθυστέρησης** (time-delay spectrum). Διαισθητικά, για μία δεδομένη γλώσσα, η k-οστή είσοδος στο φάσμα χρονοκαθυστέρησης περιέχει το μέγεθος του μικρότερου ντετερμινιστικού συντακτικού αναλυτή που ανιχνεύει ένα λάθος με χρονοκαθυστέρηση το πολύ k. Εν συντομία, το αποτέλεσμα είναι ότι, πράγματι, υπάρχουν γλώσσες, για τις οποίες η βαθμιαία αύξηση της χρονοκαθυστέρησης επιτρέπει βαθμιαία μείωση της πολυπλοκότητας του συντακτικού αναλυτή.

Πέρα από τους ορθούς συντακτικούς αναλυτές προθέματος, πολλοί συντακτικοί αναλυτές στην πράξη κατασκευάζονται, όπως έχουμε ήδη τονίσει, με βάση τις LL(k) και LR(k) γραμματικές. Επειδή οι συντακτικοί αναλυτές αυτού του είδους προκύπτουν γενικά αρκετά μεγάλοι σε μέγεθος, πολλή προσοχή έχει δοθεί στην εύρεση τρόπων μείωσης του μεγέθους τους. Όπως αναφέραμε και στην Ενότητα 3.5 (Θεώρημα 3.5.6), οι γραμματικές LR(1) αρκούν για την περιγραφή όλων των ντετερμινιστικών γλωσσών χωρίς συμφραζόμενα, και κατά συνέπεια, οποιαδήποτε αύξηση επιχειρήσουμε στο μήκος της πρόβλεψης των LR(k) γραμματικών δε θα αυξήσει και την εκφραστική δύναμή τους. Έτσι, τουλάχιστον θεωρητικά, δεν υπάρχει λόγος να χρησιμοποιήσουμε για τις γραμματικές LR(k) πρόβλεψη μήκους μεγαλύτερου από 1. Από την άλλη, η κλάση των γλωσσών LL(k) περιέχεται γνήσια στην κλάση των LL($k+1$), που σημαίνει ότι περισσότερες γλώσσες μπορούν να περιγραφούν με χρήση πρόβλεψης μεγαλύτερου μήκους για τις LL γραμματικές.

Στο [15] του 1996 όμως μελετήθηκε διεξοδικά η χρήση των LL και LR γραμματικών στην πράξη και μέσω παραδειγμάτων γλωσσών πρακτικού ενδιαφέροντος φάνηκε για πρώτη φορά η ανάγκη, παρά τα προαναφερθέντα, να χρησιμοποιηθούν όχι μόνο LL, αλλά και LR γραμματικές με μήκος πρόβλεψης μεγαλύτερο του 1. Τα έτη 2000 και 2001 ακολούθησαν και άλλες μελέτες, από περισσότερο θεωρητική τώρα σκοπιά, και το πόρισμα πιστοποίησε πως η χρήση μεγαλύτερης πρόβλεψης στις LL και LR γραμματικές μπορεί πράγματι να αποβεί εξαιρετικά καρποφόρα για τη συντακτική ανάλυση ορισμένων γλωσσών. Το γεγονός αυτό έχει μεγάλο ενδιαφέρον ιδιαίτερα για την LR περίπτωση, καθώς δείχνει ότι, αν και όχι απαραίτητο, είναι πιθανότατα σκόπιμο να αυξήσουμε το μήκος της πρόβλεψης μιας LR γραμματικής, καθαρά για λόγους πολυπλοκότητας.

Επίλογος

Στα κεφάλαια που προηγήθηκαν επιχειρήσαμε μία μεστή, αλλά χωρίς επιμονή στην αυστηρή, μαθηματική θεμελίωση—η οποία, εξάλλου, βρίσκεται εύκολα σε πολλά αναγνώσματα, για τα οποία και υπάρχουν παραπομπές—παρουσίαση των βασικών στοιχείων της θεωρίας τυπικών γλωσσών.

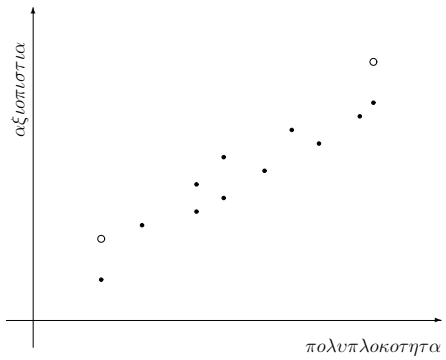
Οι στόχοι μας ήταν δύο. Από τη μία το ζητούμενο ήταν μία σφαιρική μελέτη του αντικειμένου και η εξέτασή του τόσο από την πλευρά της θεωρίας τυπικών γραμματικών, όσο και από την πλευρά της θεωρίας αυτομάτων, που, αν και φαινομενικά διαφορετικές, προσπαθήσαμε να αναδείξουμε τα θαυμαστά πολλά σημεία επαφής τους. Φροντίσαμε να μην παραλείψουμε ιδιαίτερες αναφορές στις τόσο πλούσιες πρακτικές εφαρμογές της θεωρίας.

Δεδομένου, από την άλλη, ότι σε πολλά σημεία της κλασικής θεωρίας τυπικών γλωσσών η έρευνα έχει ήδη φτάσει στην κορύφωσή της και σήμερα θεωρείται κάπως παρωχημένη, μη έχοντας πια να προσφέρει πολλά καινούρια πράγματα, η δεύτερη επιδίωξή μας ήταν να δώσουμε μία, έστω και μερική, εικόνα των περισσότερο επίκαιρων ζητημάτων, στα οποία είναι στραμμένη η σύγχρονη έρευνα. Για το σκοπό αυτό επιλέξαμε να παρουσιάσουμε το γενικό πλαίσιο, αλλά και αρκετά από τα θεμελιώδη αποτελέσματα, της θεωρίας πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους. Ο παραπάνω είναι ένας κλάδος που κερδίζει συνεχώς έδαφος, καθώς δείχνει ικανός να αποδώσει λύσεις σε πολλά πρακτικά ζητήματα που απασχολούν τους ερευνητές σήμερα.

Θα θέλαμε να ολοκληρώσουμε την παρουσίασή μας σκιαγραφώντας εν συντομία τη σύνδεση της πολυπλοκότητας υπολογιστικών μοντέλων με περιορισμένους πόρους με ένα τέτοιο, επίκαιρο ζήτημα, και συγκεκριμένα με αυτό της αξιοπιστίας λογισμικού. Αν και είναι γεγονός ότι ακόμα δεν έχουν δημοσιευθεί αποτελέσματα που συνδέουν σθεναρά τις δύο περιοχές, σοβαρές ενδείξεις για κάτι τέτοιο υπάρχουν ήδη από το 2000. Άλλωστε είναι και διαισθητικά εμφανές ότι η «πολυπλοκότητα» ενός συστήματος λογισμικού και η αξιοπιστία του σχετίζονται. Προφανώς όσο πιο απλά μπορούμε να εκφράσουμε ένα αντικείμενο, τόσο πιο λίγα λάθη ενδεχομένως θα κάνουμε, και τόσο πιο εύκολο

θα είναι να το εκφράσουμε σωστά.

Επομένως, η επιλογή ενός κατάλληλου συστήματος περιγραφής, που επιτρέπει την εύκολη αναπαράσταση του λογισμικού που μας ενδιαφέρει, μπορεί να βοηθήσει σημαντικά στη βελτίωση της αξιοπιστίας του. Βέβαια αυτό δε σημαίνει σε καμία περίπτωση ότι η μικρότερη δυνατή περιγραφή ενός συστήματος λογισμικού αποτελεί πάντα τη βέλτιστη λύση, καθώς απλότητα δεν ισοδυναμεί πάντα με περιεκτικότητα. Ακόμα, προκειμένου να τυποποιηθούν οι έννοιες που μας απασχολούν, είναι απαραίτητο να βρεθούν μέτρα και για την «απλότητα» και για την «αξιοπιστία» των συστημάτων λογισμικού, που φυσικά μπορεί να είναι διαφορετικά. Στο Σχήμα 5.3 φαίνεται μια περίπτωση, στην οποία, βάσει των συγκεκριμένων μέτρων που θεωρήθηκαν, οι πιο περίπλοκες περιγραφές τείνουν να είναι και πιο αξιόπιστες.



Σχήμα 5.3: Περιγραφές ενός αντικειμένου σε συγκεκριμένο σύστημα. Το ενδιαφέρον εστιάζεται στη σχέση ανάμεσα στην απλούστερη περιγραφή με την υψηλότερη αξιοπιστία και την πιο αξιόπιστη περιγραφή με τη μεγαλύτερη απλότητα, καθώς επίσης και στο «μονοπάτι» μεταξύ των δύο.

Κλείνοντας, προς αποφυγή εσφαλμένων εντυπώσεων οφείλουμε, ασφαλώς, να τονίσουμε πως το σημερινό επίπεδο της επιστημονικής γνώσης απέχει ακόμη πολύ από το να είναι σε θέση να αποδείξει συγκεκριμένες ιδιότητες σύνθετων μηχανολογικών συστημάτων, όπως ενός αεροσκάφους[5] ή άλλων

[5]Για την ακρίβεια, είναι πολύ δύσκολο ακόμα και να δημιουργηθούν τα αντίστοιχα συστήματα περιγραφών που μοντελοποιούν ένα τόσο περίπλοκο σύστημα. Ένα μοντέρνο αεροσκάφος έχει υδραυλικά τμήματα, ηλεκτρονικούς δείκτες και συστήματα ελέγχου λογισμικού, όλα συνδεόμενα μεταξύ τους με έναν τόσο σύνθετο τρόπο, που απαιτούνται στην κυριολεξία χιλιάδες σελίδων λεπτομερών περιγραφών για να δειχτεί π.χ. ότι η ισχύς του συστήματος θα πέσει αυτόματα σε περίπτωση χαμηλής ένδειξης καυσίμων σε μία από τις μηχανές.

τεχνολογικών συσκευών, βασιζόμενες στην πολυπλοκότητα των περιγραφών τους. Έχουμε, ωστόσο, βάσεις για να αισιοδοξούμε, πως μια μέρα θα είναι.

Βιβλιογραφία

[1] M. D. Davis, R. Sigal, E. J. Weyuker. *Computability, Complexity, and Languages.* Academic Press, 1994.

[2] J. Goldstine, M. Kappes, C. M. R. Kintala, H. Leung, A. Malcher, D. Wotschke. *Descriptional Complexity of Machines with Limited Resources.* Journal of Universal Computer Science, vol. 8, no. 2, 2002.

[3] J. Goldstine, C. M. R. Kintala, D. Wotschke. *On measuring nondeterminism in regular languages.* Inform. and Comput., 86(2):179–194, 1990.

[4] J. Goldstine, H. Leung, D. Wotschke. *Measuring nondeterminism in pushdown automata.* In STACS 97 (Lübeck), volume 1200 of LNCS, pages 295–306. Springer-Verlag, Berlin, 1997.

[5] C. Herzog. *Pushdown automata with bounded nondeterminism and bounded ambiguity.* Theoret. Comput. Sci., 181(1):141–157, 1997.

[6] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 2nd ed., 2001.

[7] J. E. Hopcroft, J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[8] J. Hromkovič, G. Schnitger. *On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata.* Inform. and Comput., 169(2):284–296, 2001.

[9] J. Hromkovič, G. Schnitger. *On the power of Las Vegas II: Two-way finite automata.* Theoret. Comput. Sci., 262(1-2):1–24, 2001.

[10] C. Kapoutsis. *Deterministic moles cannot solve liveness.* In: Proceedings of the 7th Workshop on Descriptional Complexity of Formal Systems, 2005.

[11] C. Kapoutsis. *Removing Bidirectionality from Nondeterministic Finite Automata.* In: Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science, 2005.

[12] D. E. Knuth. *On the translation of languages from left to right.* Information & Control, 8, 1965.

[13] H. R. Lewis, X. X. Παπαδημητρίου. *Στοιχεία Θεωρίας Υπολογισμού.* Εκδόσεις Κριτική, 2005.

[14] A. R. Meyer, M. J. Fischer. *Economy of description by automata, grammars, and formal systems.* IEEE Twelfth Annual Symposium on Switching and Automata Theory, 1971.

[15] T. J. Parr, R. W. Quong. *LL and LR translators need $k > 1$ lookahead.* ACM SIGPLAN Notices, 31(2):27–34, 1996.

[16] G. E. Révész. *Introduction to Formal Languages.* McGraw-Hill Computer Science Series, 1985.

[17] E. M. Schmidt. *Succinctness of descriptions of context-free, regular and finite languages.* PhD thesis, Cornell University, Ithaca, NY, 1978.

[18] S. Sippu, E. Soisalon-Soininen. *Parsing Theory.* Springer-Verlag, 1990.

[19] M. Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.

[20] T. A. Sudkamp. *Languages and Machines.* Addison-Wesley, 1988.

[21] Σ. Ζάχος. *Αυτόματα και Τυπικές Γραμματικές.* Σημειώσεις μαθήματος, 2004.

Ευρετήριο