Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Θεμελιώδη Θέματα Επιστήμης Υπολογιστών, 2019-20

1η Σειρά Γραπτών Ασκήσεων

(αυτόματα – τυπικές γλώσσες – γραμματικές - λογική – υπολογισιμότητα – πολυπλοκότητα)

Ονοματεπώνυμο: *Θοδωρής Αράπης*

(el18028, theodoraraps2000@gmail.com)

Άσκηση 1.

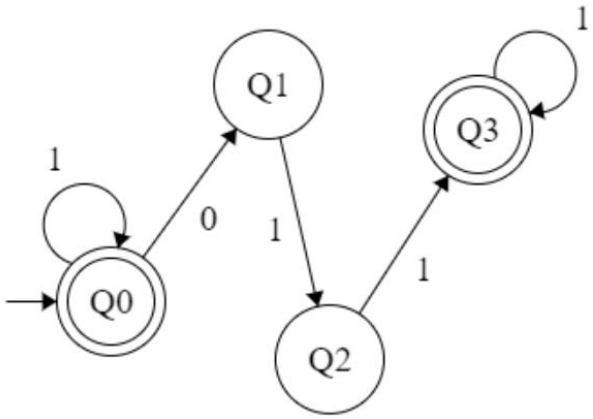
Q	0	1	2	3	4	5	6	7	8	9
Q_0	Q_0	Q_1	Q_2	Q_0	Q_1	Q_3	Q_0	Q_1	Q_2	Q_0
Q_1	Q_1	Q_3	Q_0	Q_1	Q_2	Q_0	Q_1	Q_3	Q_0	Q_1
Q_2	Q_2	Q_0	Q_1	Q_3	Q_0	Q_1	Q_2	Q_0	Q_1	Q_3
Q_3	Q_2	Q_0	Q_1	Q_3	Q_0	Q_1	Q_2	Q_0	Q_1	Q_3

Άσκηση 2.

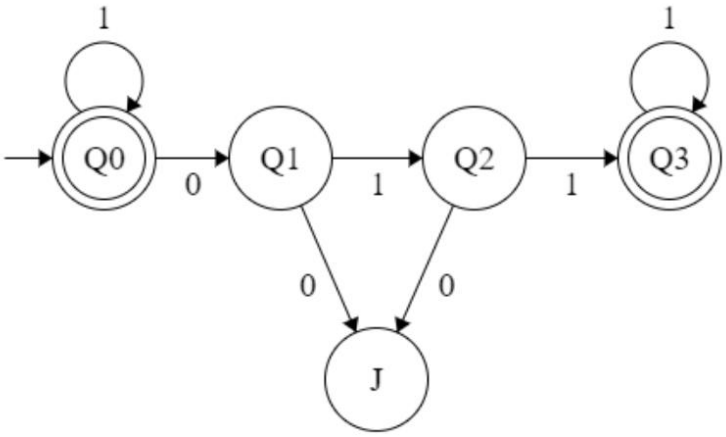
L_1:

(i)

Q	0	1
Q_0	Q_1	Q_0
Q_1	Ø	Q_2
Q_2	Ø	Q_3
Q_3	Q_1	Q_3

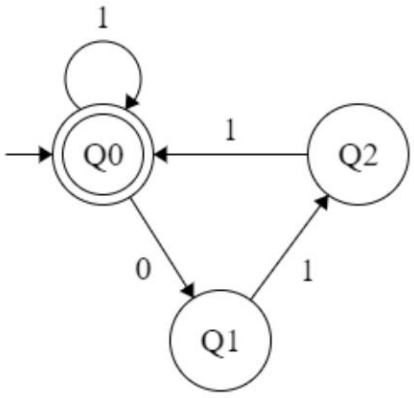


(ii) Ισοδύναμο DFA:



Ελαχιστοποίηση DFA:

Q_0			
Q_1	**X_0**		
Q_2	**X_0**	**X_1**	
Q_3		**X_0**	**X_0**
	Q_0	**Q_1**	**Q_2** **Q_3**



Ελαχιστοποιημένο DFA

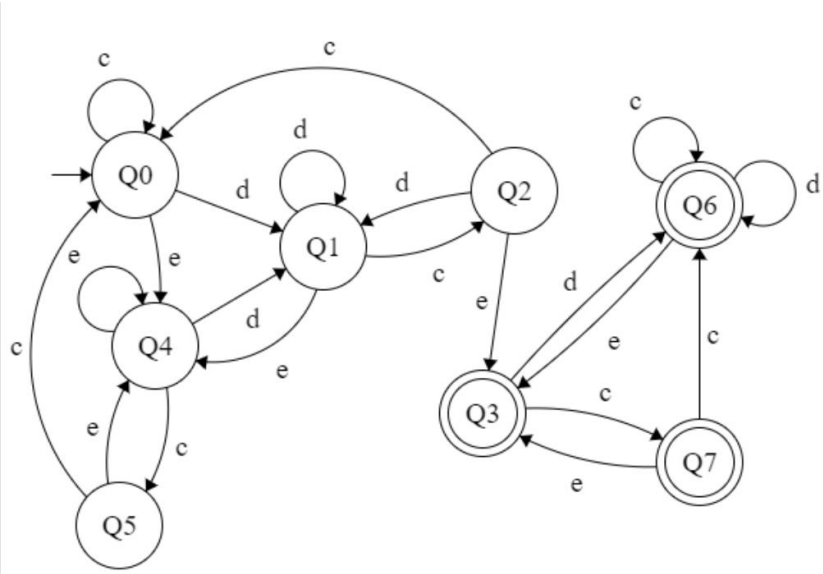
Με εφαρμογή του θεωρήματος Myhill-Nerode, παραπάνω, αποδεικνύεται ότι το DFA στο (i) δεν είναι ελάχιστο.

(iii) Κανονική παράσταση της γλώσσας L_1: ***1*(011⁺)****

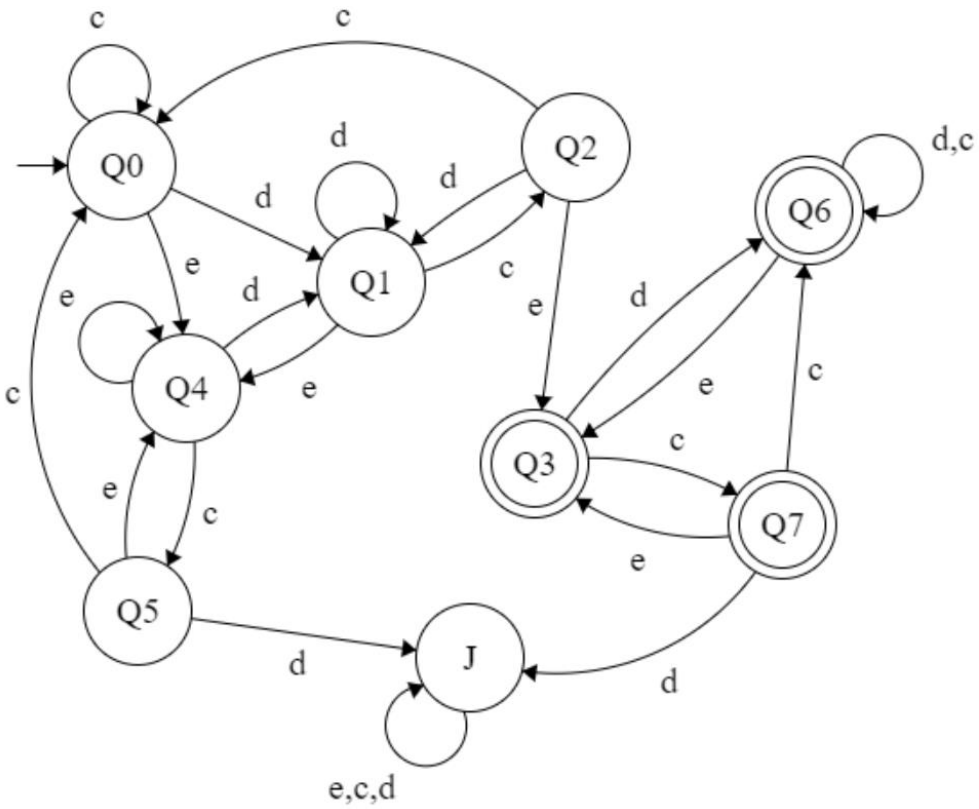
L_2:

(i)

Q	a	d	c
Q_0	Q_0	Q_1	Q_4
Q_1	Q_2	Q_1	Q_4
Q_2	Q_0	Q_1	Q_3
Q_3	Q_7	Q_6	Q_3
Q_4	Q_5	Q_1	Q_4
Q_5	Q_0	Ø	Q_4
Q_6	Q_6	Q_6	Q_3
Q_7	Q_6	Ø	Q_3



(ii) <u>Ισοδύναμο DFA:</u>



Ελαχιστοποίηση DFA:

Q_0							
Q_1	X_3						
Q_2	X_2	X_2					
Q_3	X_0	X_0	X_0				
Q_4	X_2	X_3	X_2	X_0			
Q_5	X_1	X_3	X_2	X_0	X_1		
Q_6	X_0	X_0	X_0	X_2	X_0	X_0	
Q_7	X_0	X_0	X_0	X_0	X_0	X_0	X_0
	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6 Q_7

Με εφαρμογή του θεωρήματος Myhill-Nerode, παραπάνω, αποδεικνύεται ότι το DFA στο (i) είναι ελάχιστο (αφού δεν υπάρχουν καταστάσεις που να μπορούν να συγχονευτούν).

(iii) Κανονική παράσταση της γλώσσας L_2:

$$[c^*[(ee^*c(e^+c)^*c)^*+d+ed(e^+d)^*]d^*c(d^+c)^*(c^+[(ee^*c(e^+c)^*c)^*+d+ed(e^+d)^*]d^*c(d^+c)^*)^*e^+c(e^+c)^*c+c^*[(ee^*c(e^+c)^*c)^*+d+ed(e^+d)^*]d^*c(d^+c)^*(c^+[(ee^*c(e^+c)^*c)^*+d+ed(e^+d)^*]d^*c(d^+c)^*)^*e^+d(e^+d)^*]c^*d^*$$

Πιο απλά:

Για $\lambda = c^*[(ee^*c(e^+c)^*c)^*+d+ed(e^+d)^*]d^*c(d^+c)^*$

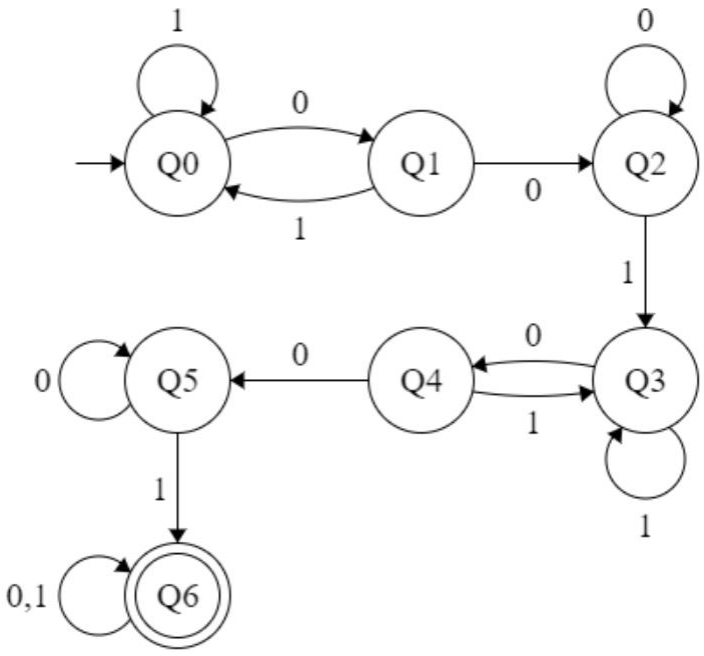
θα έχω:

$$[\lambda(c\lambda)^*e^+c(e^+c)^*c+\lambda(c\lambda)^*e^+d(e^+d)^*]c^*d^*$$

L_3:

(i)

Q	0	1
Q_0	Q_1	Q_0
Q_1	Q_2	Q_0
Q_2	Q_2	Q_3
Q_3	Q_4	Q_3
Q_4	Q_5	Q_3
Q_5	Q_5	Q_6
Q_6	Q_6	Q_6



(ii) Ισοδύναμο DFA:

Στην πραγματικότητα το παραπάνω είναι DFA. Προκειμένου να ήταν αισθητή η διαφόρα DFA-NFA θα μπορούσαμε να προσθέσουμε στο προηγούμενο ορισμένες μεταβάσεις (π.χ. $\delta(Q_2, 0) \rightarrow Q_0$), ώστε να έχουμε πολλαπλές μεταβάσεις από μία κατάσταση για μία δεδομένη είσοδο.

Ελαχιστοποίηση DFA:

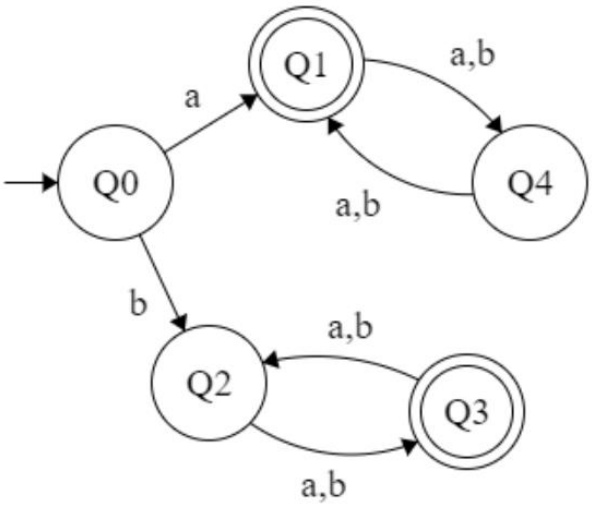
Q_0						
Q_1	X_5					
Q_2	X_4	X_4				
Q_3	X_3	X_3	X_3			
Q_4	X_2	X_2	X_2	X_2		
Q_5	X_1	X_1	X_1	X_1	X_1	
Q_6	X_0	X_0	X_0	X_0	X_0	X_0
	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5 Q_6

Με εφαρμογή του θεωρήματος Myhill-Nerode, παραπάνω, αποδεικνύεται ότι το DFA στο (i) είναι ελάχιστο (αφού δεν υπάρχουν καταστάσεις που να μπορούν να συγχονευτούν).

(iii) Κανονική παράσταση της γλώσσας L_3: ***(1+0)*001(1+0)*001(1+0)****

L_4:

Q	a	b
Q_0	Q_1	Q_2
Q_1	Q_4	Q_4
Q_2	Q_3	Q_3
Q_3	Q_2	Q_2
Q_4	Q_1	Q_1

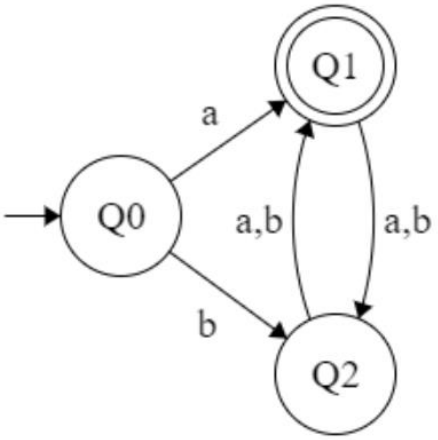


(ii) Ισοδύναμο DFA:

Το παραπάνω είναι και DFA.

Ελαχιστοποίηση DFA:

Q_0				
Q_1	X_0			
Q_2	X_1	X_0		
Q_3	X_0		X_0	
Q_4	X_1	X_0		X_0
	Q_0	Q_1	Q_2	Q_3



Ελαχιστοποιημένο DFA

Με εφαρμογή του θεωρήματος Myhill-Nerode, παραπάνω, αποδεικνύεται ότι το DFA στο (i) δεν είναι ελάχιστο.

(iii) Κανονική παράσταση της γλώσσας L_4: ***[a+b(a+b)](aa+ab+ba+bb)****

Άσκηση 3.

α) Η γλώσσα δεν είναι κανονική γιατί δεν υπάρχει αυτόματο που να μπορεί να "μετράει" το πλήθος των '0' ή των '1'. Επομένως δεν υπάρχει κάποιος τρόπος να ελέγξει αν όντως υπάρχουν διπλάσσια '0' από '1'.

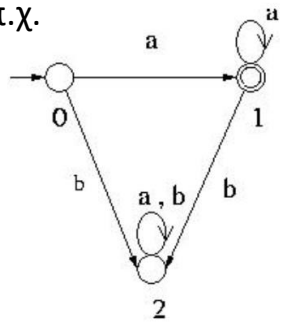
Απόδειξη:

Θα χρησιμοποιήσουμε το Pumping Lemma.

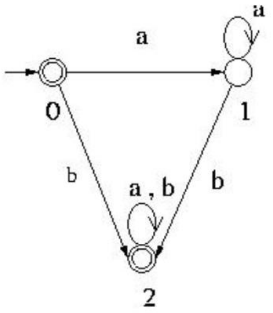
Έστω ότι η γλώσσα είναι κανονική. Θέτω $n \in Z$ το Pumping length και αποδεικνύω για κάθε n. Διαλέγω $z \in L$ τέτοιο ώστε $|z| = 3n \geq n$ (2n το πλήθος των '0' και n το πλήθος των '1'). Η z μπορεί να γραφεί: $z = uvw$ με $|uv| \leq n$ και $|v| \geq 1$. Αρκέι να μελετήσουμε τα strings για τα οποία ισχύει: $z = 0^{2n}1^n \in L$. Τότε αναγκαστικά $v = 0^k$, $k \geq 1$. Διαλέγω i = 2: $uvvw = 0^{2n+k}1^n \in L$. Άτοπο. Άρα δεν είναι κανονική γλώσσα.

***Απόδειξη για το ότι αν μια γλώσσα δεν είναι κανονική τότε ούτε η συμπληρωματική της θα είναι κανονική:
Έστω ένα DFA που αποδέχεται μία γλώσσα L. Ισχύει ότι αν αντιστραφούν οι καταστάσεις που αποδέχεται με τις καταστάσεις που δεν αποδέχεται, τότε δημιουργείται ένα DFA που αποδέχεται την συμπληρωματική της L. Αν είχαμε NFA τότε θα έπρεπε πρώτα να μετατραπεί στο ισοδύναμό του DFA και μετά να παίρναμε το αντεστραμμένο αυτού. Συνεπώς αν μία γλωσσα είναι κανονική και επομένως υπάρχει αυτόματο που να την αποδέχεται, τότε (πάντα) και η συμπληρωματική της θα είναι κανονική (αφού θα υπάρχει αυτόματο που να την αποδέχεται, άμα κάνουμε την προαναφερθείσα αντιστροφή). Επομένως εύκολα συμπεραίνουμε πως αν μία γλώσσα δεν είναι κανονική τότε ούτε και η συμπληρωματική της θα είναι κανονική.

π.χ.



accepts a^+



accepts the complement of a^+

β) Η γλώσσα δεν είναι κανονική. Αρκεί να δείξω ότι η γλώσσα που αποδέχεται παλινδρομικές συμβολοσειρές (δηλαδή η συμπληρωματική της) δεν είναι κανονική. Οπότε σύμφωνα με την απόδειξη (***) ούτε αυτή θα είναι κανονική.

Η L = {w ∈ {0, 1}* : η w είναι παλινδρομική} δεν είναι κανονική.

<u>Απόδειξη:</u>

Θα χρησιμοποιήσουμε το Pumping Lemma.

Έστω ότι η γλώσσα είναι κανονική. Θέτω nϵZ το Pumping length και αποδεικνύω για κάθε n. Διαλέγω zϵL τέτοιο ώστε |z| = 2n ≥ n (2n το μήκος της z). Η z μπορεί να γραφεί: z = uvw με |uv| ≤ n και |v| ≥ 1. Τότε:

z = $\overbrace{\underbrace{10100\ldots0}_{u}\underbrace{1}_{v}}^{n}\overbrace{\underbrace{10\ldots00101}_{w}}^{n}$ για i = 2: uvvw = $\overbrace{\underbrace{10100\ldots0}_{u}\underbrace{1}_{v}\underbrace{1}_{v}}^{n+1}\overbrace{\underbrace{10\ldots00101}_{w}}^{n}$,

η οποία δεν ανήκει στην L διότι δεν είναι παλινδρομική. Άρα έχουμε άτοπο. Επομένως δεν είναι κανονική γλώσσα. Άρα από (***) ούτε η {w ∈ {0, 1}* : η w δεν είναι παλινδρομική} είναι κανονική.

γ) Η γλώσσα είναι κανονική αφού οι συμβολοσειρές της γλώσσας έχουν πεπερασμένο μήκος. Συνεπώς, υπάρχει αυτόματο που να μπορεί να περιγράψει όλες τις συμβολοσειρές της γλώσσας και να τις αποδέχεται.

δ) Η γλώσσα δεν είναι κανονική γιατί δεν υπάρχει αυτόματο που να μπορεί να “μετράει” το πλήθος των ‘0’ ή των ‘1’. Επομένως δεν υπάρχει κάποιος τρόπος να ελέγξει αν όντως υπάρχει διαφορετικό πλήθος ‘0’ και ‘1’. Αρκεί να δείξω ότι η συμπληρωματική της δεν είναι κανονική. Οπότε σύμφωνα με την απόδειξη (***) ούτε αυτή θα είναι κανονική.

Η L = {w ∈ {0, 1}* : 0^n1^m όπου n = m} δεν είναι κανονική.

Απόδειξη:

Θα χρησιμοποιήσουμε το Pumping Lemma.

Έστω ότι η γλώσσα είναι κανονική. Θέτω n∈Z το Pumping length και αποδεικνύω για κάθε n. Διαλέγω z∈L τέτοιο ώστε |z| = 2n ≥ n (n το πλήθος των ‘0’ και n το πλήθος των ‘1’). Η z μπορεί να γραφεί: z = uvw με |uv| ≤ n και |v| ≥ 1. Τότε:

z = $\overbrace{\underbrace{00\dots0}_{u}\underbrace{00}_{v}}^{n}\overbrace{\underbrace{11\dots11}_{w}}^{n}$ για i = 2: uvvw = $\overbrace{\underbrace{00\dots0}_{u}\underbrace{00}_{v}\underbrace{00}_{v}}^{n+2}\overbrace{\underbrace{11\dots11}_{w}}^{n}$,

η οποία δεν ανήκει στην L διότι δεν είναι παλινδρομική. Άρα έχουμε άτοπο. Επομένως δεν είναι κανονική γλώσσα. Άρα από (***) ούτε η {w ∈ {0, 1}* : 0^n1^m όπου n ≠ m} είναι κανονική.

Άσκηση 4.

α) Η γλώσσα που παράγει η G είναι η: {w ∈ {a, b}* : το πλήθος των ‘a’ είναι μεγαλύτερο ή ισο από το πλήθος των ‘b’}, όπου το b μπορεί να έχεις και μηδενικό πλήθος.

β) Η γραμματική για τη γλώσσα θα είναι γραμματική χωρίς συμφραζόμενα

και θα είναι η εξής: G: S → aaaAbb, A → aaaAbb|ε.

Θα χρειαστούμε ένα αυτόματο στοίβας για να αποδεχτούμε αυτήν την γλώσσα. Το αυτόματο αυτό θα λειτουργεί ως εξής: Θα μετράει το πλήθος των ‘a’ που εισέρχονται (χρησιμοποιώντας την πράξη modulo3) και κάθε φορά που διαβάζει τρία ‘a’ θα κάνει push δύο ‘b’ στην στοίβα. Έτσι, όταν

στην είσοδο αρχίσει να διαβάζει ‘b’, για κάθε ένα που διαβάζει θα αφαιρεί και ένα ‘b’ από την στοίβα μέχρι αυτή να αδειάσει.

γ) Η γραμματική για τη γλώσσα θα είναι γραμματική με συμφραζόμενα και θα είναι η εξής:

G: $S \rightarrow A111B$, $A111B \rightarrow 0A111B0|0A111B0|1A111B1|1A111B0|111$

Άσκηση 5.

Έστω L μία γλώσσα χωρίς συμφραζόμενα και G η γραμματική της. Θα εξετάσουμε αν η L^R είναι και αυτή γλώσσα χωρίς συμφραζόμενα. Έστω G^R η γραμματική της L^R, προκειμένου να ισχύει:

Αν G: $A \rightarrow \alpha$, τότε G^R: $A \rightarrow \alpha^R$

Αν G: $A \rightarrow w_1Bw_2$, $B \rightarrow u$, τότε G^R: $A \rightarrow w_1^R Bw_2^R$, $B \rightarrow u^R$

Όπου στην πρώτη έχουμε w_1uw_2 και στην δευτερη $w_1^R u^R w_2^R = (w_1uw_2)^R$

Αν G: $A \rightarrow \varepsilon$, τότε G^R: $A \rightarrow \varepsilon$

Παρατηρούμε λοιπόν οτι σε κάθε περίπτωση προκύπτει πράγματι η ανάστροφη συμβολοσειρά. Άρα, οι γλώσσες χωρίς συμφραζόμενα είναι κλειστές ως προς την πράξη αναστροφή.

Άσκηση 6.

Halting Problem

Έστω ότι το πρόβλημα είναι επιλύσιμο. Είναι γνωστό ότι το σύνολο όλων των προγραμμάτων είναι αριθμήσιμο. Ορίζουμε μία μηχανιστική απαρίθμηση ($\Pi_1, \Pi_2, \Pi_3, \ldots, \Pi_n$) όλων των προγραμμάτων. Δημιουργούμε ένα πρόγραμμα Π το οποίο να ελέγχει αν ένα πρόγραμμα τερματίζει ή όχι με τέτοιον τρόπο ώστε αν το Π_v, $v \in N$, με είσοδο v τερματίζει τότε και το Π να τερματίζει. Το Π, όμως, θα πρέπει να ανήκει στην παραπάνω αρίθμηση. Έστω k, $\Pi = \Pi_k$, ο δείκτης του Π. Δίνοντας, λοιπόν, ως είσοδο το k, παρατηρούμε ότι το $\Pi_k(k)$ θα τερματίζει ανν το Π_k τερματίζει, το οποίο

συμβαίνει όταν το $\Pi_k(k)$ δεν τερματίζει ποτέ. Άτοπο. Άρα το Halting Problem είναι μη αποκρίσιμο.

Totality Problem

Αρκεί να δείξουμε ότι το Halting Problem ανάγεται στο Totality Problem.

Αφού κανένας αλγόριθμος δεν μπορεί να επιλύσει το HP, τότε κανένας αλγόριθμος δεν θα μπορεί να επιλύσει το TP. Για κάθε πρόγραμμα Π_v με είσοδο ν κατασκευάζουμε ένα πρόγραμμα Π_v' , το οποίο λαμβάνει μία αφηρημένη είσοδο, την αγνοεί και τρέχει το $\Pi_v(v)$. Το Π_v' θα τερματίζει για κάθε είσοδο αν και μόνο αν το Π_v τερματίζει για είσοδο ν. Συνεπώς, ένας αλγόριθμος που μας πληροφορεί για το εάν το Π_v' τερματίζει για κάθε είσοδο, μας πληροφορεί και για το εάν τερματίζει για είσοδο ν. Έχουμε λοιπόν μία λύση για το HP.

Halting Problem $\leq^p$ Totality Problem

Άσκηση 7.

Ο αλγόριθμος είναι ο εξής:

Αρχικά, εφόσον γνωρίζουμε το πλήθος των literals (αλλιώς διασχίζουμε μία φορά τον τύπο και το βρίσκουμε), δημιουργούμε έναν πινακα α[N] (integer), όπου N το πλήθος των literals και θέτουμε σε κάθε θέση την τιμή '1'. Ύστερα διατρέχουμε ξανά τον τύπο και εξετάζουμε τις φράσεις ανάμεσα στις παρενθέσεις. Τοποθετούμε σε έναν πίνακα μεταβλητού μεγέθους τις μεταβλητές που διαβάζονται κατά αύξουσα σειρά δείκτη. Ελέγχουμε αν υπάρχουν σε γειτονικές θέσεις μεταβλητές και το συμπλήρωμά τους. Αν ναι τότε προχωράμε στις επόμενες παρενθέσεις και επαναλαμβάνουμε.

- Αν σε κάποιο ζεύγος παρενθέσεων δεν βρούμε συμπληρωματικές μεταβλητές στις γειτονικές θέσεις, τότε για κάθε συμπληρωματική μεταβλητή ($\neg x_i$), στον μεταβλητού μεγεθους πίνακα, βάζουμε '0' στην θέση α[i]. Τέλος, εκττυπώνουμε «NAI» και για κάθε i από 0 εώς και N-1 εκτυπώνουμε την ανάθεση [π.χ. (TRUE, FALSE, FALSE, …)], όπου η πρώτη τιμή καθορίζεται από το α[0] (αν α[0] = 0 τοτε «TRUE» αλλιώς «FALSE»), η δεύτερη από το α[1] κ.ο.κ.

- Αν σε κάθε ζεύγος παρενθέσεων βρούμε συμπληρωματικές μεταβλητές, τοτε εκτυπώνουμε «ΟΧΙ».
- Αν η είσοδος είναι κενή τότε ο αλγόριθμος θα εκτυπώνει «ΟΧΙ».

Άσκηση 8.

α) Αρχικά παρατηρούμε ότι αν ένα υπογράφημα του G είναι κλίκα, τότε, αν αντιστρέψουμε τον G, η κλίκα αυτή θα αποτελεί ένα ανεξάρτητο σύνολο κορυφών για τον G' και αντίστροφα. Συνεπώς, κάθε αλγόριθμος που εντοπίζει ένα ανεξάρτητο σύνολο κορυφών (μεγέθους τουλάχιστον k) σε έναν γράφο G, μπορεί να χρησιμοποιηθεί για την εύρεση κλίκας (μεγέθους τουλάχιστον k) στον ίδιο γράφο. Αυτό συμβαίνει διότι τα σύνολα των κορυφών που δεν είναι ανεξάρτητα, είναι κλίκες. Πιο αναλυτικά, έστω S ένα ανεξάρτητο σύνολο για έναν δεδομένο γράφο G = (V, E). Τότε, για κάθε ακμή e = (x, y), e∈E και x, y∈V, το πολύ μία από τις δύο κορυφές της x, y θα ανήκει στο S. Ως εκ τούτου, τουλάχιστον μία από αυτές θα ανήκει στο S^R (S^R = V – S). Συνεπώς, κάθε ακμή έχει τουλάχιστον μία κορυφή της στο S^R. Άρα, το S^R είναι κλίκα.

Είναι, λοιπόν, προφανές ότι αν ένας αποδοτικός αλγόριθμος επιλύει το πρόβλημα Clique, μπορεί να χρησιμοποιηθεί για να επιλύσει το πρόβλημα Independent Set για τουλάχιστον κάποιο μέγεθος k, αν τον βάλουμε να αναζητήσει αν ο G έχει μία κλίκα μεγέθους το πολύ n-k (όπου n το μέγεθος του γράφου). Συνεπώς, λόγω αναγωγής, αφού το Independent Set είναι NP-complete, τότε και το Clique θα είναι NP-complete.

Independent Set $\leq^p$ Clique

β) Έχοντας αποδείξει παραπάνω ότι το Clique είναι NP-complete, μπορούμε να δείξουμε ότι αυτό ανάγεται στο Dense Subgraph. Θα ισχύει:

Έστω μία είσοδος (G, k) του προβλήματος Clique. Εφόσον ο G έχει τουλάχιστον μία κλίκα C μεγέθους k και πάνω, τότε είναι προφανές πως θα υπάρχει μία κλίκα C_k μεγέθους ακριβώς k ($C_k \subseteq C$). Είναι γνωστό ότι ένας πλήρης γράφος (και συνεπώς εδώ η C_k) έχει πλήθος ακμών: $\frac{k(k-1)}{2}$, όπου k το πλήθος των κορυφών του. Επομένως, γνωρίζοντας αυτά, μπορούμε να

λύσουμε το πρόβλημα Clique για (G, k), λύνοντας το πρόβλημα Dense Subgraph για γράφο G, a = k και b = $\frac{k(k-1)}{2}$. Συνεπώς, το Clique ανάγεται στο Dense Subgraph. Άρα και το Dense Subgraph είναι NP-complete.

Clique $\leq^p$ Dense Subgraph