Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

1η Σειρά Γραπτών Ασκήσεων

Ιωάννης Τσαντήλας, Α.Μ.: 03120883

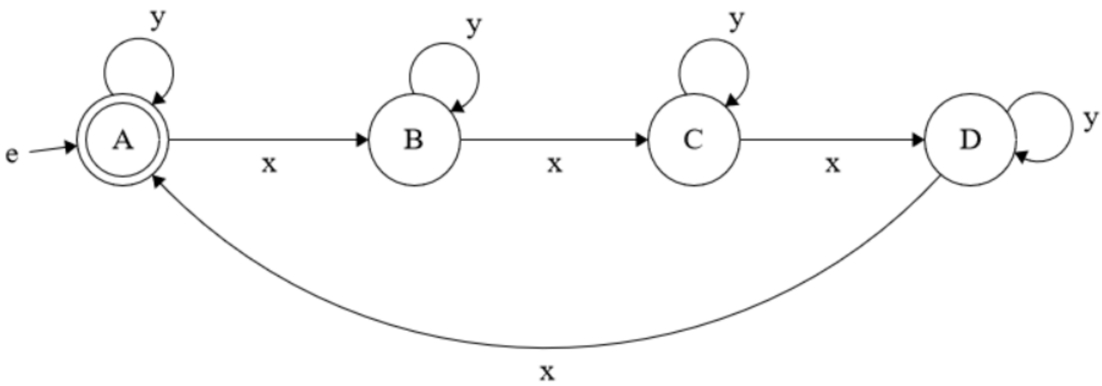
(Αυτόματα – Τυπικές Γλώσσες – Γραμματικές - Λογική – Υπολογισιμότητα – Πολυπλοκότητα)

Άσκηση 1

Κατασκευάστε DFA, κανονικές παραστάσεις και κανονικές γραμματικές για κάθε μία από τις παρακάτω γλώσσες:

α) $Σ_1$ = {x, y} των οποίων το πλήθος των 'x' είναι πολλαπλάσιο του 4.

- Η συμβολοσειρά θα έχει τη δομή y_n x y_m x y_z x y_k x y_l, με n,m,k,l φυσικούς αριθμούς – δηλαδή πλήθος τετράδων x όπου ανάμεσα τους μπορούμε να βάλουμε όσα y θέλουμε. Το DFA λοιπόν είναι:



Εικόνα 1.1α: DFA Άσκησης 1α.

- Όσον αφορά την παράσταση, έχουμε:

  A = e + Ay + Dx (1)

  B = By + Ax (2)

  C = Cy + Bx (3)

  D = Dy + Cx (4)

  Χρησιμοποιώντας στις (2), (3), (4) την ταυτότητα R = Q + RL = QL* έχουμε:

  B = Axy*

  C = Bxy*

  D = Cxy*

  Αντικαθιστώντας την (3) στην (4): D = Bxy*xy* και την (2) στην νέα (4): D = Axy*xy*xy*. Εάν βάλουμε την νέα (4) στην (1):

  A = e + Ay + Axy*xy*xy*x = e + A(y + Axy*xy*xy*x) = e(y + xy*xy*xy*x)* και αφού εA = A τότε:

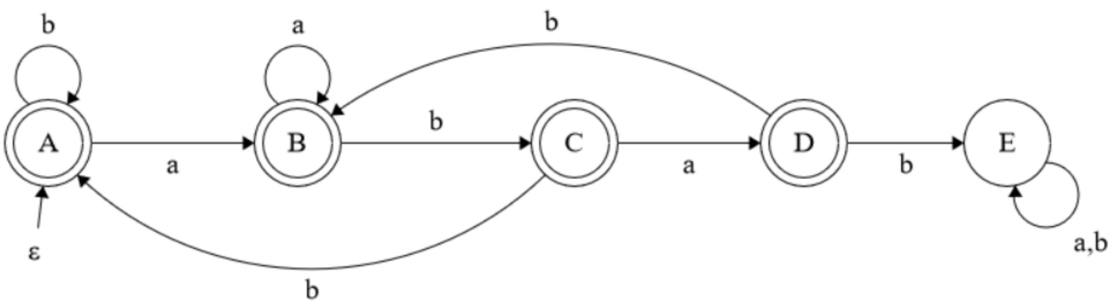
A = (y + xy*xy*xy*x)* =>

(y + xy*xy*xy*x)*

- Τέλος, η γραμματική του DFA είναι μια τετράδα G = (V,T,P,S) με:
  1. V τις καταστάσεις: V = {A, B, C, D}
  2. T το αλφάβητο: T = {x,y}
  3. S την αρχική κατάσταση: S = {A}
  4. P τους κανόνες παραγωγής:
     a. A -> yA | xB | e
     b. B -> xC | yB
     c. C -> xD | yC
     d. D -> xA | yD

β) Σ_2 = {a, b} που δεν περιέχουν δύο συνεχόμενα 'ab'.

- Το DFA μας θα πρέπει να ελέγχει, στην περίπτωση που δημιουργηθεί 'aba', να μην προστεθεί ένα επιπλέον b και, σε αυτή τη περίπτωση, να οδηγείται σε junk state. Το DFA είναι:



Εικόνα 1.1β: DFA Άσκησης 1β.

- Όσον αφορά την παράσταση, επειδή έχουμε 4 τελικές καταστάσεις, η παράσταση θα προκύψει από την ένωση αυτών των τεσσάρων. Επομένως, πρώτα θα βρούμε ξεχωριστά για κάθε μία και τέλος θα τις προσθέσουμε. Άρα, έχουμε:

  A = ε + Ab + Cb (1)

  B = Ba + Da + Aa (2)

  C = Bb (3)

  D = Ca (4)

  E = Ea + Eb + Db (5)

  (3), (4) => D = Bba

  (2), (4) => B = Ba + Bbaa + Aa = B (a + baa) + Aa = Aa(a + baa)*

(2), (3) => $C = Bb = Aa(a + baa)^*b$

(1), (3) => $A = \varepsilon + Ab + Cb = \varepsilon + Ab + Aa(a + baa)^*bb = \varepsilon + A(b + a(a + baa)^*bb) =$

$= \varepsilon(b + a(a + baa)^*bb)^*$ =>

$A = (b + a(a + baa)^*bb)^*$

$B = Aa(a + baa)^* = (b + a(a + baa)^*bb)^*a(a + baa)^*$

$C = Aa(a + baa)^*b = (b + a(a + baa)^*bb)^*a(a + baa)^*b$

$D = Bba = (b + a(a + baa)^*bb)^*a(a + baa)^*ba$

Άρα, εάν A +B + C + D:

$(b + a(a + baa)^*bb)^* + (b + a(a + baa)^*bb)^*a(a + baa)^* + (b + a(a + baa)^*bb)^*a(a + baa)^*b +$

$(b + a(a + baa)^*bb)^*a(a + baa)^*ba$ =>

$(b + a(a + baa)^*bb)^* [\ \varepsilon + a(a + baa)^* + a(a + baa)^*b + a(a + baa)^*ba\ ]$ =>

$$\mathbf{(b + a(a + baa)^*bb)^* \{\ \varepsilon + a(a + baa)^* [\varepsilon + b + baa]\ \}}$$

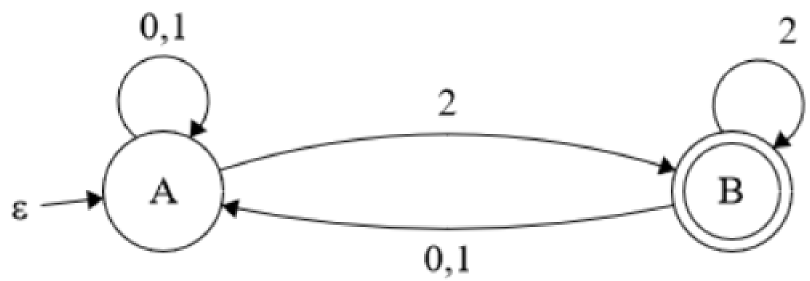
- Τέλος, η γραμματική του DFA είναι μια τετράδα G = (V,T,P,S) με:
  5. V τις καταστάσεις: V = {A, B, C, D, E}
  6. T το αλφάβητο: T = {a,b}
  7. S την αρχική κατάσταση: S = {A}
  8. P τους κανόνες παραγωγής:
     a. A -> aB | bA | e
     b. B -> aB | bC
     c. C -> aD | bA
     d. D -> aB | bE
     e. E -> aE | bE

Άσκηση 2

Κατασκευάστε DFA που υλοποιούν τις παρακάτω γλώσσες και δώστε επίσης τις αντίστοιχες κανονικές καταστάσεις:

α) $\Sigma_1 = \{0, 1, 2\}$ ώστε n mod 3 = 2.

- Οποιοσδήποτε αριθμός που θα διαιρεθεί με το 3 μπορεί να αφήσει υπόλοιπο 0, 1 ή 2. Επομένως, θα έχουμε 3 καταστάσεις, A (υπόλοιπο 0), B (υπόλοιπο 1), C (υπόλοιπο 2), όπου μόνο η τρίτη θα είναι αποδοχής. Παρατηρούμε ωστόσο, ότι οι A, B μπορούν να συγχωνευτούν, επομένως το DFA είναι:



Εικόνα 1.2α: DFA Άσκησης 2α.

- Όσον αφορά την παράσταση:

  $A = \varepsilon + A0 + A1 + B0 + B1 = \varepsilon + A(0 + 1) + B(0+1) = [\varepsilon + B(0 + 1)]\,(0 + 1)^*$ (1)

  $B = A2 + B2 = A22^*$ (2)

  Εάν αντικαταστήσουμε την (2) στην (1):

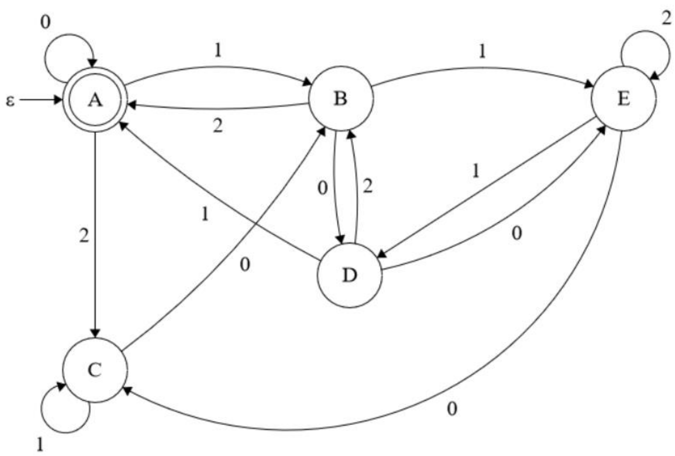
  $A = [\varepsilon + A22^*(0 + 1)]\,(0 + 1)^* = \varepsilon\,(0 + 1)^* + A22^*(0 + 1)\,(0 + 1)^* \Rightarrow$

  $A = \varepsilon\,(0 + 1)^*\,[\,22^*(0 + 1)\,(0 + 1)^*\,]^* \Rightarrow$

  $$\mathbf{(0 + 1)^*\,[\,22^*\,(0 + 1)\,(0 + 1)^*\,]^*}$$

β) $\Sigma_1 = \{0, 1, 2\}$ ώστε n mod 5 = 0.

- Οποιοσδήποτε αριθμός διαιρεθεί με το 5, θα αφήσει υπόλοιπο 0, 1, 2, 3 ή 4. Επομένως θα έχουμε 5 καταστάσεις A (υπόλοιπο 0), B (υπόλοιπο 1), C (υπόλοιπο 2), D (υπόλοιπο 3), E (υπόλοιπο 4). Η A θα είναι η κατάσταση υποδοχής. Το DFA θα είναι:



Εικόνα 1.2β: DFA Άσκησης 2β.

Όσον αφορά την παράσταση:

A = ε + A0 + B2 +D1 (1)

B = A1 + C0 + D0 (2)

C = A2 + C1 + E0 (3)

D = C2 + B2 + E1 (4)

E = B1 + E2 + D0 (5)

(5) => E = B1 + E2 + D0 = (B1 + D0)2*

(3) => C = A2 + C1 + E0 = (A2 + E0)1* => (5) => A21* + (B1 + D0)2*01*

(4) => (3), (5) => D = A21*2 + B12*01* + D02*01*2 + B2 + B12*1 + D02*1 =>

D = A21*2(02*101*2 + 02*1)* + B(12*01* + 2 + 12*1)(02*01*2 + 02*1)*

(2) => (3), (4) => B = A1 + A21*0 + B10*0 + D02*0 =

= A(1 + 21*0) + B101*0 + A21*2(02*101*2 + 02*1)*02*0 +

B(12*01* + 2 + 12*1)(02*01*2 + 02*1)*02*0 =>

=> B = A[1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 + (12*01* + 2 + 12*1)

(02*01*2 + 02*1)*02*0]*

Τελικά, εάν αντικαταστήσουμε στην (1):

A = ε + A0 + B2 +D1 = ε + A0 + A[1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 +

(12*01* + 2 + 12*1) (02*01*2 + 02*1)*02*0]*2 + A21*2(02*101*2 + 02*1)* +

B(12*01* + 2 + 12*1)(02*01*2 + 02*1)* =>

A = ε + A0 + A[1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 +

(12*01* + 2 + 12*1) (02*01*2 + 02*1)*02*0]*2 + A21*2(02*101*2 + 02*1)* +

A[1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 + (12*01* + 2 + 12*1)

(02*01*2 + 02*1)*02*0]* (12*01* + 2 + 12*1)(02*01*2 + 02*1)* =>

A = { 0 + 1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 + (12*01* + 2 + 12*1) (02*01*2 +

02*1)*02*0]*2 + 21*2(02*101*2 + 02*1)* + [1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0]

[101*0 + (12*01* + 2 + 12*1) (02*01*2 + 02*1)*02*0]* (12*01* + 2 + 12*1)(02*01*2 +

02*1)* } * =>

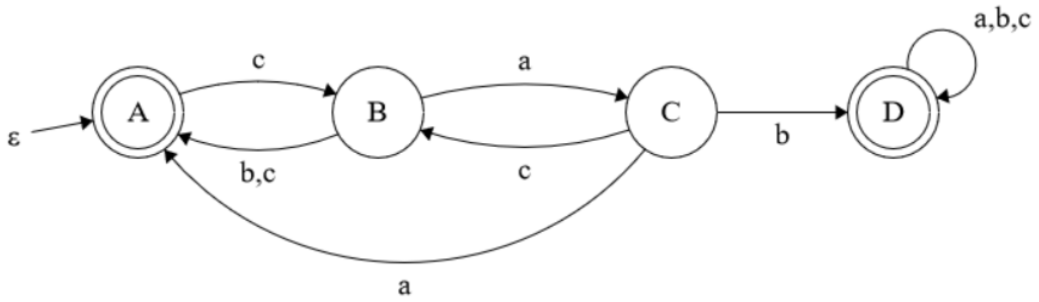
{ [0 + 1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 + (12*01* + 2 + 12*1) (02*01*2 + 02*1)*02*0]*2 + 21*2(02*101*2 + 02*1)* + [1 + 21*0 + 21*2(02*101*2 + 02*1)*02*0] [101*0 + (12*01* + 2 + 12*1) (02*01*2 + 02*1)*02*0]* (12*01* + 2 + 12*1)(02*01*2 + 02*1)* } *

Άσκηση 3

Κατασκευάστε τα ελάχιστα DFA που υλοποιούν τις παρακάτω γλώσσες:

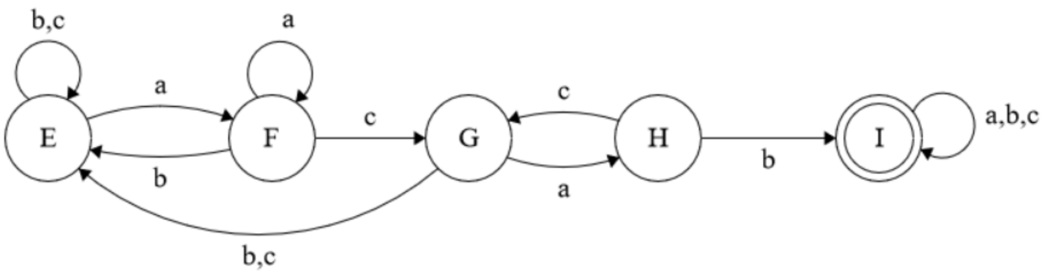
α) L_1 = { ω є {a, b, c}* | cab є ω, acab !є ω}.

- Αρχικά, θα δημιουργήσουμε τα δύο ξεχωριστά DFA, για τη δημιουργία της cab και της acab. Το DFA για το cab είναι:



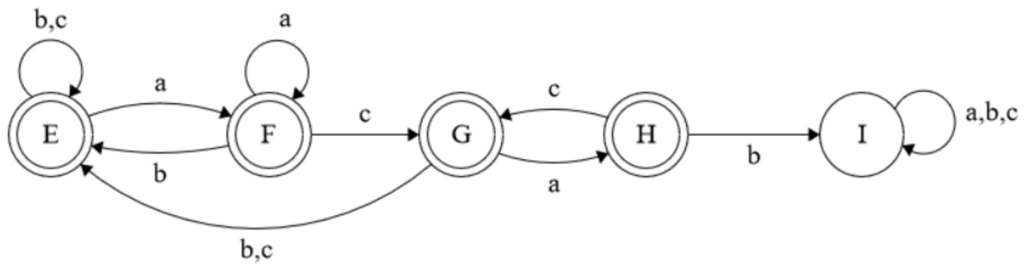
Εικόνα 1.3α: DFA που αποδέχεται συμβολοσειρές που περιέχουν το 'cab'.

Ενώ για το acab είναι:



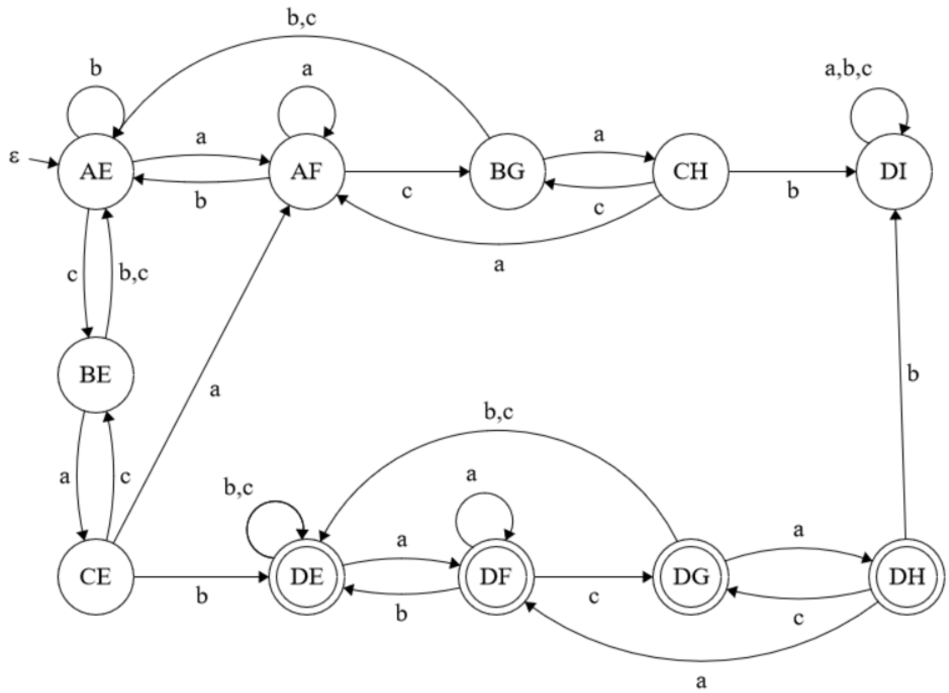
Εικόνα 1.3α': DFA που αποδέχεται συμβολοσειρές που περιέχουν το 'acab'.

Βέβαια, θέλουμε η acab να μην περιέχεται στην ω, άρα θα αλλάξουμε την τελική κατάσταση σε μη τελική και τις μη τελικές, σε τελικές:



Εικόνα 1.3α'': DFA που δεν αποδέχεται συμβολοσειρές που περιέχουν το 'acab'.

Το ζητούμενο DFA είναι ο πολλαπλασιασμός των DFA που εμφανίζονται στις Εικόνες 1.3α, 1.3α'', με τις τελικές καταστάσεις να είναι οι «κοινές» των δύο, δηλαδή οι DE, DF, DG, DI:



Εικόνα 1.3α''': DFA Άσκησης 1.3α.

Ας αποδείξουμε τώρα ότι είναι το ελάχιστο δυνατό. Γράφουμε τον πίνακα καταστάσεων:

	a	b	c
->AE	AF	AE	BE
AF	AF	AE	BG
BG	CH	AE	AE
CH	AF	DI	BG
BE	CE	AE	AE
CE	AF	DE	BE
DE*	DF	DE	DE
DF*	DF	DE	DG
DG*	DH	DE	DE
DH*	DE	DI	DG
DI	DI	DI	DI

Πίνακας 1.1: Πίνακας καταστάσεων DFA 1.3α''''.

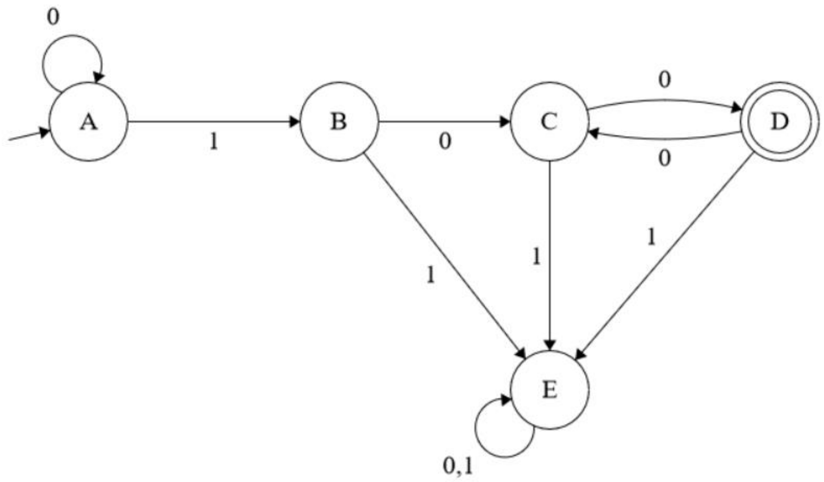
Στην συνέχεια, αναζητούμε την ισοδυναμία των καταστάσεων:

- 0-Equivalence: {AE, AF, BG, CH, BE, CE, DI} {DE, DF, DG, DH}
- 1-Equivalence: {AE, AF, BG, CH, BE, DI} {CE} {DE, DF, DG} {DH}
- 2-Equivalence: {AE, AF, BG, CH, DI} {CE} {BE} {DE, DF} {DG} {DH}
- 3-Equivalence: {AF, BG, CH, DI} {AE} {CE} {BE} {DE} {DF} {DG} {DH}
- 4-Equivalence: {CH, DI} {AF} {BG} {AE} {CE} {BE} {DE} {DF} {DG} {DH}
- 5-Equivalence: {CH} {DI} {AF} {BG} {AE} {CE} {BE} {DE} {DF} {DG} {DH}

Βλέπουμε πως καμία κατάσταση δεν κατέληξε στο ίδιο σύνολο, άρα όλες είναι ανεξάρτητες μεταξύ τους. Επομένως το εν λόγω DFA είναι πράγματι το ελάχιστο.

β) L_2 = { ω ϵ {0,1}* | ω δυαδικός τιμής 4^k, k ≥ 1}.

- Η συμβολοσειρά θα έχει τη μορφή 1 ακολουθούμενο από k ζευγάρια μηδενικών, αφού: 4 = 100, 16 = 10000, 64 = 1000000, 256 = 100000000. Επομένως, το ζητούμενο DFA είναι:



Εικόνα 1.3β: DFA Άσκησης 1.3β.

Ας αποδείξουμε πως είναι το ελάχιστο. Αρχικά, γράφουμε τον πίνακα καταστάσεων:

	0	1
->A	A	B
B	C	E
C	D	E
D*	C	E
E	E	E

Πίνακας 1.2: Πίνακας καταστάσεων DFA 1.3β.

Στην συνέχεια, αναζητούμε την ισοδυναμία των καταστάσεων:

- 0-Equivalence: {A, B, C, E} {D}
- 1-Equivalence: {A, B, E} {C} {D}
- 2-Equivalence: {A, E} {B} {C} {D}
- 3-Equivalence: {A} {E} {B} {C} {D}

Καμία κατάσταση δεν κατέληξε στο ίδιο σύνολο, άρα όλες οι καταστάσεις είναι ανεξάρτητες. Επομένως, πράγματι είναι το ελάχιστο.

Άσκηση 4

Είναι οι παρακάτω γλώσσες κανονικές;

α) $L_1 = \{ \omega \in \{0, 1\}^* \mid n_0 \neq 2n_1 \}$.

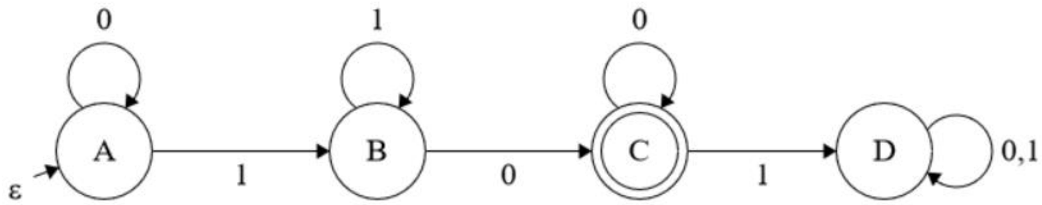
Έστω ότι η L_1 είναι κανονική. Έστω string $z = 0^n1^n$, με $n_0 = n_1 = n$, δηλαδή $n_0 \neq 2n_1$, άρα $z \in L_1$. Έστω $z = u\ v\ w$, με $u = 0^{n-1}$, $v = 0$, $w = 1^n$, $|uv| \leq n$, $|v| \geq 1$. Αφού η L_1 είναι κανονική, τότε για $i=n+1$, θα πρέπει $uv^iw \in L_1 \Rightarrow 0^{n-1}0^{n+1}1^n \in L_1 \Rightarrow n_0 = 2n$, $n_1 = n \Rightarrow n_0 = 2n_1$. Άρα, μέσω του λήμματος άντλησης, καταλήγουμε σε άτοπο και η L_1 δεν είναι κανονική.

β) $L_2 = \{0^n1^+0^m,\ n,m \geq 1,\ n \leq 2m\}$.

Έστω ότι η L_2 είναι κανονική. Έστω string $z = 0^n1^n0^n$, με $n = m = n$, δηλαδή $n \neq 2m = 2n$, άρα $z \in L_1$. Έστω $z = u\ v\ w$, με $u = 0^{n-1}$, $v = 0$, $w = 1^n0^n$, $|uv| \leq n$, $|v| \geq 1$. Αφού η L_2 είναι κανονική, τότε για $i=n+1$, θα πρέπει $uv^iw \in L_2 \Rightarrow 0^{n-1}0^{n+1}1^n0^n \in L_1 \Rightarrow n = 2n$, $m = n \Rightarrow n = 2m = 2n$. Άρα, μέσω του λήμματος άντλησης, καταλήγουμε σε άτοπο και η L_2 δεν είναι κανονική.

γ) $L_3 = \{0^n1^+0^m,\ n,m \in N\}$.

Μπορούμε να δείξουμε πως η γλώσσα είναι κανονική, εάν υπάρχει DFA τέτοιο ώστε να την υλοποιεί. Πράγματι, το εξής DFA την υλοποιεί και άρα η L_3 είναι κανονική:



Εικόνα 1.4γ: Το DFA υλοποιεί την L_3.

Η παράσταση είναι:

$A = \varepsilon + A0 = \varepsilon 0^* = 0^*$ (1)

$B = A1 + B1 = A11^*$ (2)

$C = B0 + C0 = B00^*$ (3)

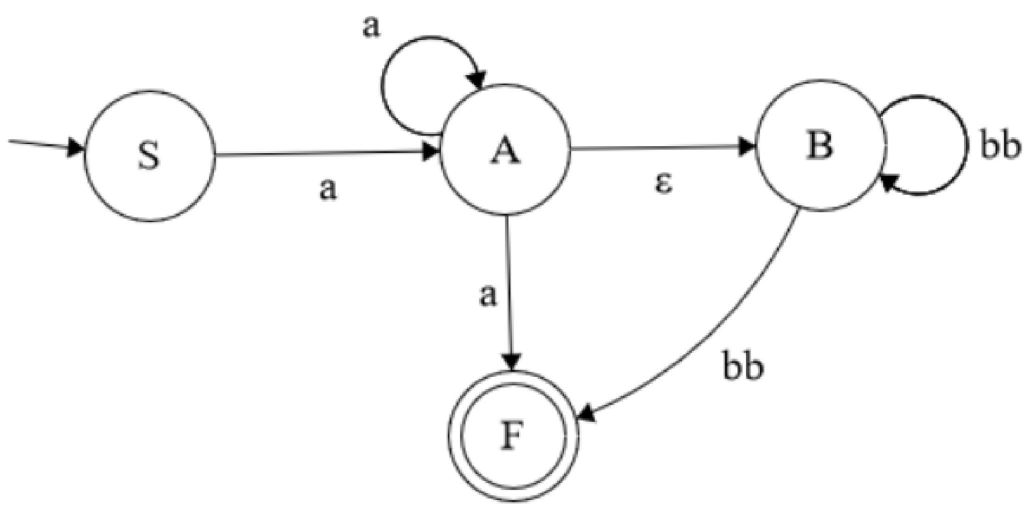
$D = D0 + D1$ (4)

(1), (2) $\Rightarrow B = 0^*11^* \Rightarrow$ (3) $\Rightarrow C = 0^*11^*00^* \Rightarrow 0^n1^+0^m$

Άσκηση 5

α) Έστω G : S → aA,A → a | aA | B,B → bb | bbB. Περιγράψτε σε φυσική γλώσσα τη γλώσσα που παράγει η G.

Μέσω των κανόνων παραγωγής δημιουργούμε το αυτόματο:



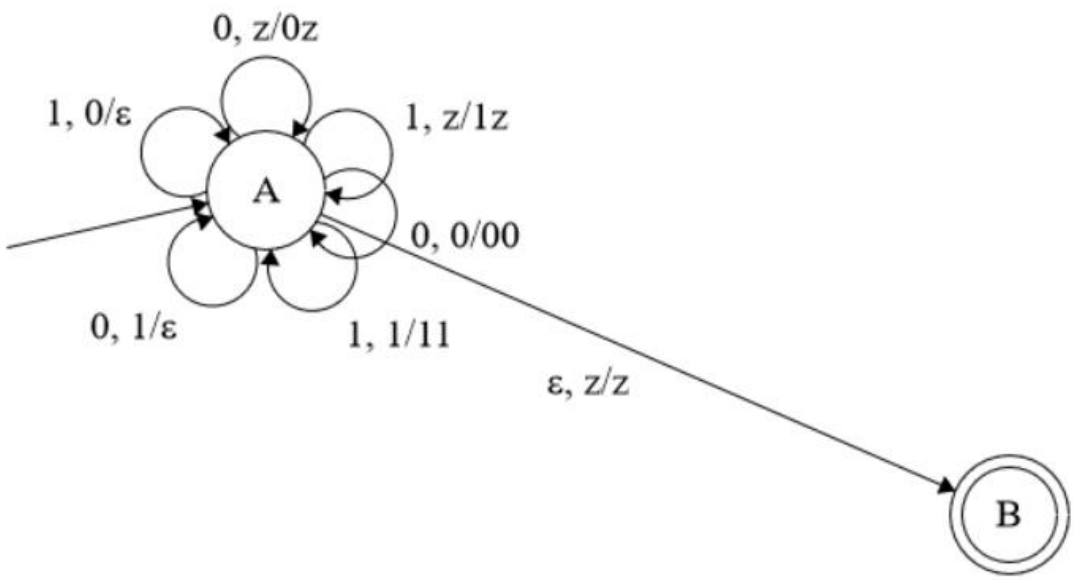
Εικόνα 1.5α: Αυτόματο Άσκησης 1.5α.

Ουσιαστικά, παράγονται συμβολοσειρές που έχουν αρχή ένα τουλάχιστον ‘α’ και είτε τελειώνουν σε ‘α’ είτε με ένα τουλάχιστον bb, δηλαδή είτε a…a είτε aa…abb…bb.

Άρα: { $a(a^+ + (bb)^+$ }

β) Περιγράψτε αυτόματο για τη γλώσσα {w ∈ {0, 1}∗ | το πλήθος των 1 στο w είναι ίσο με αυτό των 0}.

Αρχικά, θα παραθέσουμε το PDA και θα το εξηγήσουμε στη συνέχεια:



Εικόνα 1.5β: PDA Άσκησης 1.5β.

Η εξήγηση είναι σχετικά απλή. Κάθε κίνηση χαρακτηρίζεται από μια τριάδα α, β/γδ. Το α αναπαριστά την είσοδο στο PDA, το β την πιο πρόσφατη εισαγωγή στη στοίβα (η κορυφή), γ είναι η πλέον καινούργια κορυφή και το δ το αμέσως προηγούμενο (το z συμβολίζει την άδεια στοίβα). Αν το δ δεν υφίσταται στην τριάδα, τότε κάνουμε pop την κορυφή της στοίβας. Πιο συγκεκριμένα:

- **0, z/0z:** Διαβάζουμε 0, η στοίβα είναι άδεια => τοποθετούμε το 0 στην στοίβα (μετά είναι το z).
- **1, z/0z:** Διαβάζουμε 1, η στοίβα είναι άδεια => τοποθετούμε το 1 στην στοίβα (μετά είναι το z).
- **0, 0/00:** Διαβάζουμε 0, στην κορυφή της στοίβας είναι 0 => τοποθετούμε το 0 στην στοίβα (μετά είναι το 0).
- **1, 1/11:** Διαβάζουμε 1, στην κορυφή της στοίβας είναι 1 => τοποθετούμε το 1 στην στοίβα (μετά είναι το 1).
- **0, 1/ε:** Διαβάζουμε 0, στην κορυφή της στοίβας είναι 1 => δεν τοποθετούμε τίποτα στην στοίβα και αφαιρούμε την κορυφή της (δηλαδή το 1).
- **1, 0/ε:** Διαβάζουμε 1, στην κορυφή της στοίβας είναι 0 => δεν τοποθετούμε τίποτα στην στοίβα και αφαιρούμε την κορυφή της (δηλαδή το 0).
- **ε, z/z:** Δεν χρειάζεται να διαβάσουμε κάτι (ε-κίνηση), στην κορυφή της στοίβας είναι το z (η στοίβα είναι άδεια, δηλαδή δεν υπάρχει επιπλέον 0 ή 1) => μετακινούμαστε στην τελική κατάσταση.

Άσκηση 6

α) Αποδείξτε ότι η κλάση των γλωσσών χωρίς συμφραζόμενα είναι κλειστή ως προς τις πράξεις ένωση, παράθεση και άστρο του Kleene.

Απόδειξη Ένωσης

Έστω ότι το $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_2)$ αναγνωρίζει τη γλώσσα A_1 και το $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ τη γλώσσα A_2. Κατασκευάζουμε το $N = (Q, \Sigma, \delta, q_0, F)$ έτσι ώστε να αναγνωρίζει τη γλώσσα $A_1 \cup A_2$.

- $Q = \{q_0\} \cup Q_1 \cup Q_2$.
  Οι καταστάσεις του N είναι όλες οι καταστάσεις των N_1 και N_2 συν μια νέα εναρκτήρια κατάσταση q_0 (από την οποία μπορούμε να βρεθούμε στις q_1 και q_2 με μία ε-κίνηση).
- Η εναρκτήρια κατάσταση του N είναι η q_0.
- $F = F_1 \cup F_2$.
  Οι καταστάσεις αποδοχής του N είναι όλες οι καταστάσεις αποδοχής των N_1 και N_2. Έτσι, το N αποδέχεται εάν αποδέχεται κάποιο από τα N_1 και N_2.
- Ορίζουμε τη δ έτσι ώστε για κάθε $q \in Q$ και κάθε $\alpha \in \Sigma$ το $\delta(q, \alpha)$ να είναι:
  - $\delta_1(q, \alpha)$ εάν $q \in Q_1$.
  - $\delta_2(q, \alpha)$ εάν $q \in Q_2$.
  - $\{q_1, q_2\}$ εάν $q = q_0$ και $\alpha = \varepsilon$.
  - $\varnothing$ εάν $q = q_0$ και $\alpha \neq \varepsilon$.

Απόδειξη Παράθεσης

Έστω ότι το $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_2)$ αναγνωρίζει τη γλώσσα A_1 και το $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ τη γλώσσα A_2. Κατασκευάζουμε το $N = (Q, \Sigma, \delta, q_0, F)$ έτσι ώστε να αναγνωρίζει τη γλώσσα A_1A_2.

- $Q = Q_1 \times Q_2$.
- Η εναρκτήρια κατάσταση του N είναι η q_1q_2..
- $F = F_1F_2$.
  Οι καταστάσεις αποδοχής του N είναι όλες οι καταστάσεις αποδοχής των N_1 και ταυτόχρονα N_2. Έτσι, το N αποδέχεται μόνο εάν αποδέχεται το N_1 και το N_2.
- Ορίζουμε τη δ έτσι ώστε για κάθε $q \in Q$ και κάθε $\alpha \in \Sigma$ το $\delta(q, \alpha)$ να είναι:
  - $\delta_1\delta_2(q, \alpha)$ εάν $q \in Q_1 \times Q_2$.
  -
  - $\{q_1q_2\}$ εάν $q = q_0$ και $\alpha = \varepsilon$.
  - $\varnothing$ εάν $q = q_0$ και $\alpha \neq \varepsilon$.

Απόδειξη Άστρο του Kleene

Έστω ότι το $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ αναγνωρίζει τη γλώσσα A_1. Κατασκευάζουμε το $N = (Q, \Sigma, \delta, q_0, F)$ έτσι ώστε να αναγνωρίζει την $A_1{}^*$.

- $Q = \{q_0\} \cup Q_1$.

Οι καταστάσεις του N είναι αυτές του N_1, συν μια νέα εναρκτήρια κατάσταση.

- Η εναρκτήρια κατάσταση του N είναι η νέα κατάσταση q_0.
- $F = \{q_0\} \cup F_1$.

  Οι καταστάσεις αποδοχής είναι οι παλιές, συν τη νέα εναρκτήρια κατάσταση.
- Ορίζουμε τη δ έτσι ώστε για κάθε $q \in Q$ και κάθε $\alpha \in \Sigma$ το $\delta(q, \alpha)$ να είναι:
  - $\delta_1(q, \alpha)$ εάν $q \in Q_1$ και $q \neq F_1$.
  - $\delta_1(q, \alpha)$ εάν $q \in F_1$ και $\alpha \neq \varepsilon$.
  - $\delta_1(q, \alpha) \cup \{q_1\}$ εάν $q \in F_1$ και $\alpha = \varepsilon$.
  - $\{q_1\}$ εάν $q = q_0$ και $\alpha = \varepsilon$.
  - $\varnothing$ εάν $q = q_0$ και $\alpha \neq \varepsilon$.

β) Τι ισχύει για τις πράξεις της αναστροφής και του συμπληρώματος;

Αναστροφή

Έστω γραμματική Γ και έστω Η η ανάστροφη της, έτσι ώστε αν $A \rightarrow \omega$ στην Γ, τότε $A \rightarrow \omega^R$ στην Η.

Επαγωγικά θα δείξουμε πως $A \Rightarrow^*_G \omega$ ανν $A \Rightarrow^*_H \omega^R$:

- Σε 0 βήματα $A \Rightarrow^0_G A$ ανν $A \Rightarrow^*_H A$.
- Έστω ότι ισχύει $A \Rightarrow^*_G \omega_1 B \omega_2$ ανν $A \Rightarrow^*_H \omega_2^R B \omega_1^R$ μπορούμε να εφαρμόσουμε οποιαδήποτε παραγωγή $B \rightarrow \chi$ στην Γ (και στην Η το ανάστροφο) και να λάβουμε $A \Rightarrow^*_G \omega_1 \chi \omega_2$ ανν $A \Rightarrow^*_H \omega_2^R \chi \omega_1^R$, αντίστοιχα όπου πράγματι το $\omega_2^R \chi \omega_1^R$ είναι το ανάστροφο του $\omega_1 \chi \omega_2$.

Συμπλήρωμα

- Έστω γλώσσες $L_1 = \{a^m b^m c^n : m, n \geq 0\}$, $L_2 = \{a^m b^n c^n : m, n \geq 0\}$.
- Η τομή τους είναι $L = \{a^n b^n c^n : m, n \geq 0\}$, η οποία δεν είναι γ.χ.σ.
- Αν το συμπλήρωμα τους ήταν γ.χ.σ., τότε θα ήταν κλειστές για την τομή, πράγμα που δεν συμβαίνει.
- Άρα δεν είναι κλειστές ως προς το συμπλήρωμα.

Άσκηση 7

Αποδείξτε ότι το πρόβλημα του ελέγχου αν ένα πρόγραμμα τερματίζει με είσοδο την κενή συμβολο-σειρά είναι μη επιλύσιμο.

- Έστω πρόγραμμα Π, που δέχεται για είσοδο την κενή συμβολοσειρά και έστω ότι υπάρχει αλγόριθμος Α που αναγκάζει το Π να τερματίσει με "ναι".
- Θεωρούμε πως υπάρχει αλγόριθμος Β που αποφασίζει το Halting Problem. Έστω το ζευγάρι προγράμματος Π και η είσοδος Ε.
- Κατασκευάζουμε το πρόγραμμα Τ που έχει:
    - Είσοδο την κενή συμβολοσειρά
    - Τρέχει το Π με Ε. Αν τερματίσει, επιστρέφει την έξοδο.
- Ο αλγόριθμος Β ορίζεται ως:
    - Ο αλγόριθμος Α τρέχει στο Τ
    - Επιστρέφει το αποτέλεσμα της εξόδου.
- Εξαιτίας της ίδιας του της κατασκευής, ο Β τερματίζει και επιστρέφει "ναι" μόνο αν το Τ τερματίζει με Ι. Έτσι, μόνο αν ο Α υπάρχει, τότε το Halting Problem είναι decidable.
- Άτοπο.
- Άρα ένα πρόγραμμα χωρίς είσοδο είναι undecidable.

Άσκηση 8

Διατυπώστε αλγόριθμο που να δέχεται σαν είσοδο οποιονδήποτε τύπο σε μορφή Horn και να τυπώνει αν είναι ικανοποιήσιμος, μαζί με κάποιο συνδυασμό των στοιχείων του που τον επαληθεύει.

- Διαβάζουμε τον τύπο. Μπορούμε να διαβάσουμε είτε στοιχείο Χ, είτε ένωση, είτε τομή, είτε άρνηση.
- Έχουμε τον πίνακα Π, όπου για στοιχείο Χ το βάζει στην αμέσως επόμενη κενή θέση, για ένωση βάζει +, για τομή βάζει *, για άρνηση βάζει -.
- Έχουμε ένα πίνακα τιμών ΠΤ όπου κάθε θέση του έχει 0 αρχικά. Η $1^{η}$ θέση αντιστοιχεί στο Χ1 στοιχείο, η $2^{η}$ στο Χ2 κ.ο.κ. Οι τιμές μπορεί να είναι 0 (false) ή 1 (true).
- Ο τύπος που θα μας δίνετε ουσιαστικά μετατρέπεται με αυτή τη διαδικασία σε παράσταση μέσω του Π. Παράδειγμα:
    - Έστω ο τύπος (χ1 U -χ3) Ո χ2 = (χ1 + -χ3) * χ2 που θέλουμε να δούμε αν είναι ικανοποιήσιμος.
    - Ο πίνακας Π θα είναι: Π = (χ1, +, -, χ3, *, χ2, EOF)
    - Ο πίνακας τιμών των μεταβλητών χ1, χ2, χ3 είναι ΠΤ = (0, 0 ,0)
    - Ο τύπος μετατράπηκε σε παράσταση (χ1 + -χ3) * χ2.
    - Για να είναι ο τύπος ικανοποιήσιμος, θα πρέπει (για κάποιες τιμές των χ) η παράσταση να είναι διάφορη του μηδενός (δηλαδή true). Δηλαδή, κάθε παράγοντας του γινομένου να είναι διάφορος του 0 (δηλαδή true).
    - Ο αλγόριθμος μας θα εξετάζει κάθε επιμέρους όρο του γινομένου ώστε να εξακριβώσει εάν (με τις τιμές του ΠΤ) είναι όλοι διάφοροι του 0.
- Έστω while loop:
    1. Έστω ακέραιος Α=0 (άθροισμα).
    2. Ο αλγόριθμος διαβάζει τα στοιχεία του Π.
        A. Εάν διαβάσει στοιχείο $χ_i$, προσθέτει την τιμή του (από τον ΠΤ) στο Α και $Μ=χ_i$.
        B. Εάν διαβάσει -, διαβάζει και την επόμενη θέση (η οποία είναι σίγουρα ένα στοιχείο Χ) και προσθέτει στο Α το συμπλήρωμα του (δηλαδή, ανιχνεύει την τιμή του στον ΠΤ και αν είναι 0 προσθέτει 1, ενώ αν είναι 1, προσθέτει 0).
        C. Εάν διαβάσει +, προχωρά στην επόμενη θέση και επιστρέφει στο βήμα Α.
        D. Εάν διαβάσει *:
            - Αν Α = 0, τότε βρίσκει το αμέσως προηγούμενο * και διαβάζει το πρώτο στοιχείο Χ. Αν ΠΤ[Χ] = 0, την αλλάζει σε 1 και επιστρέφει στην αρχή του Π. Αν ΠΤ[Χ] = 1, τότε προχωρά στο επόμενο, έως ότου βρει κάποιο στοιχείο Χ με τιμή ΠΤ[Χ] = 0. Εάν φτάσει σε * ή EOF (δηλαδή όλοι οι όροι είναι 1, **τυπώνει ΟΧΙ και τερματίζει.**

- Αν Α > 0, προχωρά στην επόμενη θέση (η οποία είναι σίγουρα κάποιο στοιχείο Χ ή -) και ξεκινά από το βήμα 1 πάλι.

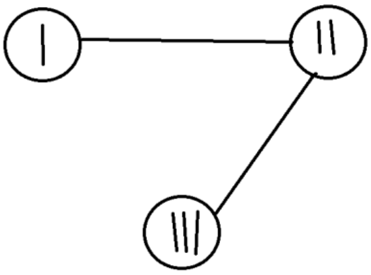
Ε. Εάν διαβάσει EOF:

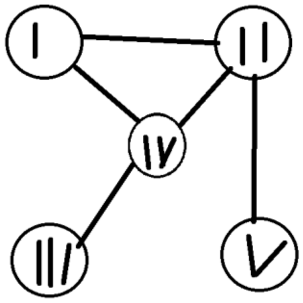
- Αν Α = 0, τότε βρίσκει το αμέσως προηγούμενο * και διαβάζει το πρώτο στοιχείο Χ. Αν ΠΤ[Χ] = 0, την αλλάζει σε 1 και επιστρέφει στην αρχή του Π. Αν ΠΤ[Χ] = 1, τότε προχωρά στο επόμενο, έως ότου βρει κάποιο στοιχείο Χ με τιμή ΠΤ[Χ] = 0. Εάν φτάσει σε * ή EOF (δηλαδή όλοι οι όροι είναι 1, **τυπώνει ΟΧΙ και τερματίζει.**
- Αν Α > 0, **τυπώνει ΝΑΙ και τον ΠΤ (0 false, 1 true).**
  Στις υποπεριπτώσεις των βημάτων

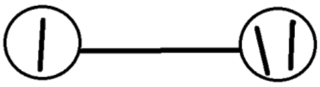
Άσκηση 9

a) Δώστε 3 παραδείγματα εισόδων για το κάθε πρόβλημα και την έξοδο.

Έστω k=2, r=5. Έστω οι γράφοι:





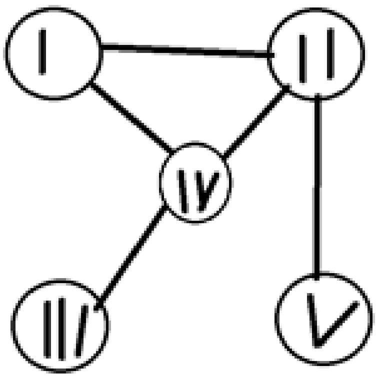


Independent Set: εκτυπώνει (αντίστοιχα): Ναι, Ναι, Όχι.

Vertex Cover: εκτυπώνει (αντίστοιχα): Όχι, Ναι, Όχι.

b) Δώστε 4 ζεύγη εισόδων της μορφής (G,k), (G,|V|−k), όπου η 1η είσοδος κάθε ζεύγους θα αφορά στο πρόβλημα IS και η δεύτερη στο πρόβλημα VC. Τι παρατηρείτε;

Έστω ο 2ος γράφος, δηλαδή ο:



Το μέγιστο ανεξάρτητο σύνολο έχει μέγεθος 3 και το ελάχιστο κάλυμμα κορυφών έχει μέγεθος 2. Ισχύει πως Output_IS = Output_VC. Πράγματι:

1. **k=2 => r=3.** IS: Ναι, VC: Ναι.
2. **k=3 => r=2.** IS: Ναι, VC: Ναι.
3. **k=4 => r=1.** IS: Όχι, VC: Όχι.
4. **k=5 => r=0.** IS: Όχι, VC: Όχι.

c) Δείξτε ότι αν είναι το VC είναι NP-πλήρες, τότε και το IS είναι NP-πλήρες και το αντίστροφο.

Από το ερώτημα (β), και από την σχέση Output_IS = Output_VC, μπορούμε να καταλάβουμε πως τα δύο προγράμματα αλληλεξαρτώνται, με αποτέλεσμα εάν ένα είναι NP-πλήρες τότε και το άλλο να είναι.