Σειρά Προβλημάτων 1 – Λύσεις

Άσκηση 1

(α) Χρησιμοποιούμε τις επιπλέον μεταβλητές PC_0, PC_1, (program counters) οι οποίες παίρνουν ως τιμές ονόματα των γραμμών του κώδικα όπως φαίνεται πιο κάτω.

```
P[0]
      while true do{
      A1 atomic{ if (fork[i] = false
                  then fork[i] = true;
            else
                  goto line 2}
      A2 atomic{ if (fork[(i+1)mod n] = false
                  then fork[(i+1)mod n] = true;
            else
                  goto line 6}
      A3 //eat;
      A4 fork[i] = false;
      A5 fork[(i+1)mod n] = false;
        }
P[1]
      while true do{
      B1 atomic{ if (fork[i] = false
                  then fork[i] = true;
            else
                  goto line 2}
      B2 atomic{ if (fork[(i+1)mod n] = false
                  then fork[(i+1)mod n] = true;
            else
                  goto line 6}
      B3 //eat;
      B4 fork[i] = false;
      B5 fork[(i+1)mod n] = false;
        }
```

Ο γράφος μεταβάσεων δίνεται από την πλειάδα (V, S , T, p) όπου V = {PC_0, PC_1, fork[2]},

S = σύνολο όλων των δυνατών αναθέσεων τιμών στις πιο πάνω μεταβλητές

T = {

T_1 : $PC_0=A_1 \wedge fork[0] = false \rightarrow (PC_0, fork[0]) := (A_2, true)$,

T_2 : $PC_0=A_1 \wedge fork[0] = true \rightarrow PC_0 := A_1$,

T_3 : $PC_0=A_2 \wedge fork[1] = false \rightarrow (PC_0, fork[1]) := (A_3, true)$,

T_4 : $PC_0=A_2 \wedge fork[0] = true \rightarrow (PC_0) := A_2$,

T_5 : $PC_0=A_3 \rightarrow PC_0 := A_4$,

T_6 : $PC_0=A_4 \rightarrow (PC_0, fork[0]) := (A_5, false)$,

T_7 : $PC_1=A_5 \rightarrow (PC_0, fork[1]) := (A_1, false)$,

T_8 : $PC_1=B_1 \wedge fork[1] = false \rightarrow (PC_1, fork[1]) := (B_2, true)$,

T_9 : $PC_1=B_1 \wedge fork[1] = true \rightarrow PC_1 := B_1$,

T_{10} : $PC_1=B_2 \wedge fork[0] = false \rightarrow (PC_1, fork[0]) := (B_3, true)$,

T_{11} : $PC_1=B_2 \wedge fork[1] = true \rightarrow PC_1 := B_2$,

T_{12} : $PC_1=B_3 \rightarrow PC_1 := A_4$,

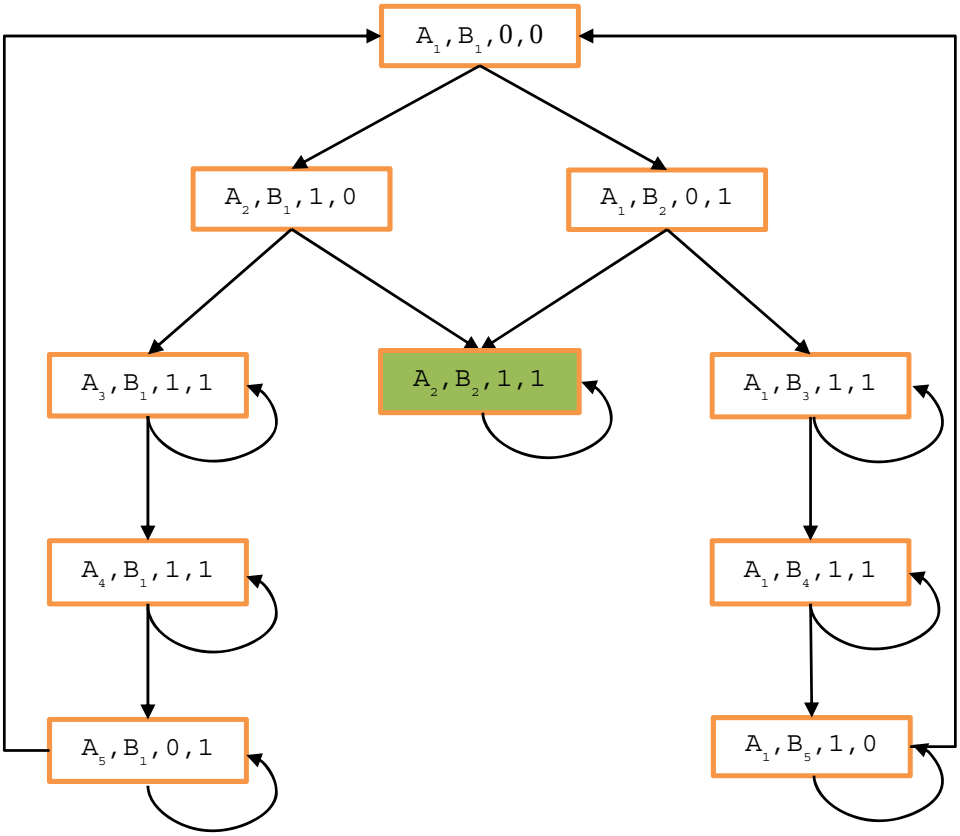
T_{13} : $PC_1=B_4 \rightarrow (PC_1, fork[1]) := (B_5, false)$,

T_{14} : $PC_1=B_5 \rightarrow (PC_1, fork[0]) := (B_1, false)$,

}

και $p \equiv PC_0=A_1 \wedge PC_1=B_1 \wedge fork[0] = false \wedge fork[1] = false$

(β) Το σύστημα μεταβάσεων αποτελείται από τετράδες (A,B,t,x,) που αναφέρονται στις τιμές των μεταβλητών (PC_0, PC_1, fork[0] και fork[1]) που ισχύουν στην κάθε κατάσταση, βάζοντας 1 για true και 0 και για false. Ακολουθεί το σχετικό σχεδιάγραμμα.



(γ) Από τη μελέτη του γράφου μεταβάσεων στο μέρος (β) οδηγούμαστε στο συμπέρασμα ότι οι διεργασίες μπορούν να οδηγηθούν σε κατάσταση όπου καμιά δεν μπορεί να προχωρήσει, άρα κανένας φιλόσοφος δεν μπορεί να φάει. Αυτό μπορεί να συμβεί, για παράδειγμα, μετά από την ακολουθία μεταβάσεων: T_1, T_8 όπου οδηγούμαστε στην κατάσταση με το πράσινο χρώμα

Άσκηση 2

(α) Χρησιμοποιούμε τις επιπλέον μεταβλητές PC_1, PC_2, (program counters) οι οποίες παίρνουν ως τιμές ονόματα των γραμμών του κώδικα όπως φαίνεται πιο κάτω.

Θεωρήστε τις δύο πιο κάτω παράλληλες διεργασίες:

P_1
```
A0: if lock = 0 then lock:= 1;
A1: x := 1;
A2: if lock = 1 then lock:= 0;
A3:
```

P_2
```
B0: if lock = 0 then lock:= 1;
B1: x := 2;
B2: if lock = 1 then lock:= 0;
B3:
```

Ο γράφος μεταβάσεων δίνεται από την πλειάδα (V, S , T, p) όπου

$V = \{PC_1, PC_2, \texttt{lock}, x\}$

S = σύνολο όλων των δυνατών αναθέσεων τιμών στις πιο πάνω μεταβλητές

T = {

T_1 : $PC_1=A_0 \wedge lock = 0 \rightarrow (PC_1, lock) := (A_1, 1)$,

T_2 : $PC_1=A_0 \wedge lock \mathrel{!}= 0 \rightarrow PC_1 := A_1$,

T_3 : $PC_1=A_1 \rightarrow (PC_1, x) := (A_2, 1)$,

T_4 : $PC_1=A_2 \wedge lock = 1 \rightarrow (PC_1, lock) := (A_3, 0)$,

T_5 : $PC_1=A_2 \wedge lock \mathrel{!}= 1 \rightarrow PC_1 := A_3$,

T_6 : $PC_2=B_0 \wedge lock = 0 \rightarrow (PC_2, lock) := (B_1, 1)$,

T_7 : $PC_2=B_0 \wedge lock \mathrel{!}= 0 \rightarrow PC_2 := B_1$,

T_8 : $PC_2=B_1 \rightarrow (PC_2, x) := (B_2, 2)$,

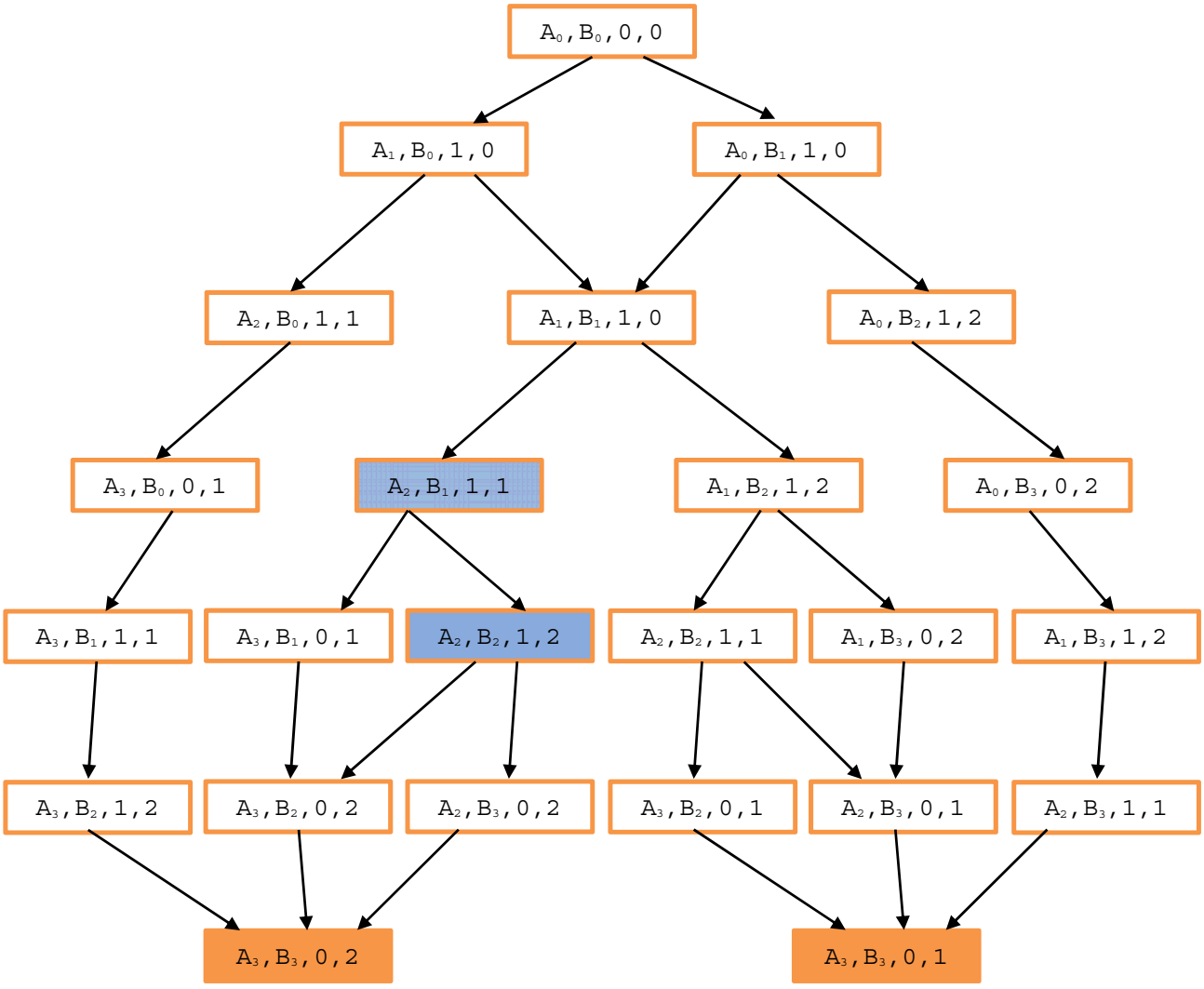
T_9 : $PC_2=B_2 \wedge lock = 1 \rightarrow (PC_2, lock) := (B_3, 0)$,

T_{10} : $PC_2=B_2 \wedge lock \mathrel{!}= 1 \rightarrow PC_2 := B_3$,

}

$p \equiv PC_1=A_0 \wedge PC_2=B_0 \wedge lock = 0 \wedge x = 0$

Το σύστημα μεταβάσεων αποτελείται από τετράδες (A,B,l,x,) που αναφέρονται στις τιμές των μεταβλητών (PC_1, PC_2, lock και x) που ισχύουν στην κάθε κατάσταση. Ακολουθεί το σχετικό σχεδιάγραμμα.



(β)

i. Η διεργασία 1 τελικά θα φτάσει στη γραμμή 3: Παρατηρούμε ότι σε κάθε εκτέλεση του συστήματος, η ιδιότητα ικανοποιείται αφού είναι δυνατή η μετάβαση στις καταστάσεις με πορτοκαλί χρώμα.
ii. Εάν το lock είναι ίσο με 1 τότε κάποτε θα γίνει ίσο με 0: Η ιδιότητα ικανοποιείται αφού οι καταστάσεις που φαίνονται με πορτοκαλί χρώμα έχουν το lock ίσο με 0. Όλες οι καταστάσεις μετά από ένα αριθμό μεταβάσεων καταλήγουν σε μια από αυτές τις δύο καταστάσεις.
iii. Όταν η διεργασία 1 φτάσει στη γραμμή 2, στο επόμενο βήμα θα πάει στη γραμμή 3: Η ιδιότητα δεν ικανοποιείται, αφού υπάρχει μετάβαση (γαλάζιες καταστάσεις) όπου η διεργασία 1 είναι στη γραμμή 2 αλλά στο επόμενο βήμα παρεμβάλλεται η διεργασία 2.
iv. Σε κάθε εκτέλεση στο μέλλον και οι δύο διεργασίες βρίσκονται στη δική τους πρώτη γραμμή ταυτόχρονα: Η ιδιότητα δεν ικανοποιείται αφού υπάρχουν εκτελέσεις κατά τις οποίες οι δύο διεργασίες δεν θα βρεθούν ταυτόχρονα στην πρώτη τους γραμμή.
v. Εάν η διεργασία 1 φτάσει στην γραμμή 3 τότε θα μείνει για πάντα στην γραμμή 3: Η ιδιότητα ικανοποιείται αφού από τον πιο πάνω γράφο παρατηρούμε σε κάθε κατάσταση στην οποία $pc_1 = 3$, δεν έχει μετάβαση όπου οδηγεί σε κατάσταση όπου $pc_1 != 3$

Άσκηση 3

(α) Η ατομική πρόταση active ικανοποιείται απείρως συχνά.

G F active

Η πρόταση ικανοποιείται από την αρχική κατάσταση της δομής αφού κάθε μονοπάτι θα πρέπει να περάσει απείρως συχνά από τουλάχιστον την κατάσταση 1.

(β) Η ατομική πρόταση entry ικανοποιείται απείρως συχνά.

G F entry

Η πρόταση δεν ικανοποιείται από την αρχική κατάσταση της δομής. Για παράδειγμα, το μονοπάτι 0123123123123… είναι τέτοιο ώστε η πρόταση entry να ικανοποιείται μόνο από την αρχική κατάσταση.

(γ) Είναι πάντα δυνατή η ικανοποίηση της ατομικής πρότασης entry.

Η πρόταση δεν μπορεί να διατυπωθεί στην LTL

(δ) Κάθε αίτημα ακολουθείται από μια ανταπόκριση.

G (request → **F** response)

Η πρόταση ικανοποιείται από την αρχική κατάσταση της δομής αφού σε κάθε μονοπάτι η κατάσταση 2 (ύπαρξη αιτήματος) ακολουθείται από την κατάσταση 3 (ύπαρξη ανταπόκρισης).

(ε) Αν από κάποια στιγμή και μετά δεν υπάρξει καινούργιο request, τότε θα ικανοποιηθεί απείρως συχνά η ατομική πρόταση entry.

FG ¬request → **GF** entry

Η πρόταση ικανοποιείται από την αρχική κατάσταση της δομής αφού στα μονοπάτια στα οποία ικανοποιείται η πρόταση **FG** ¬request (δηλαδή τα μονοπάτια που οδηγούνται στον κύκλο 010101...) ικαοποιείται απείρως συχνά η ατομική πρόταση entry.

(ζ) Η ατομική πρόταση response ικανοποιείται μόνο εφόσον έχει προηγηθεί ικανοποίηση της ατομικής πρότασης request.

G ¬ response ∨ G(¬ response **U** request)

Η πρόταση ικανοποιείται από την αρχική κατάσταση της δομής αφού σε κάθε μονοπάτι η κατάσταση 3 (ύπαρξη ανταπόκρισης) ακολουθεί την κατάσταση 2 (ύπαρξη αιτήματος).

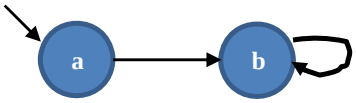
Άσκηση 4

i. **X** (a ∨ **F** a) → **F** a

Έστω δομή M με αρχική κατάσταση s και μονοπάτι w που ξεκινά από την s. Έχουμε ότι:

ω ⊨ **X** (a ∨ **F** a)	αν και μόνο αν	$\omega^1 \models a \vee \mathbf{F}\, a$
	αν και μόνο αν	$\omega^1 \models a$ ή $\omega^1 \models \mathbf{F}\, a$
	αν και μόνο αν	$\omega^1 \models a$ ή $\exists j \geq 0$, τ.ω. $(\omega^1)^j \models a$
	αν και μόνο αν	$\omega^1 \models a$ ή $\exists j \geq 1$, τ.ω. $\omega^j \models a$
	αν και μόνο αν	$\exists j \geq 1$, τ.ω. $\omega^j \models a$
	τότε	M, s ⊨ **F** a

ii. Από την πιο πάνω απόδειξη παρατηρούμε ότι το αντίστροφο δεν θα ισχύει εάν το a αληθεύει στην αρχική κατάσταση και δεν το βρίσκουμε στη συνέχεια. Αντιπαράδειγμα:



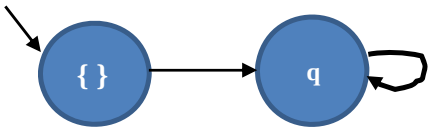
iii. **GF** p ∧ **FG** q → **FG** (**F** p ∨ q)

iv. Έστω δομή M με αρχική κατάσταση s και μονοπάτι w που ξεκινά από την s. Έχουμε ότι:

Αν w ⊨ **GF** p ∧ **FG** q	αν και μόνο αν	ω ⊨ **GF** p και ω ⊨ **FG** q
	αν και μόνο αν	(1) $\forall i \geq 0, \exists j \geq 0$, τέτοιο ώστε $(\omega^i)^j \models p$ και
		(2) $\exists\, k \geq 0$, τέτοιο ώστε, $\forall m \geq 0\ (\omega^k)^m \models q$
	αν και μόνο αν	(1) $\forall i \geq 0, \exists j \geq i$, τέτοιο ώστε $\omega^j \models p$ και
		(2) $\exists\, k \geq 0$, τέτοιο ώστε, $\forall m \geq k\ \omega^m \models q$
	τότε	$\exists\, k \geq 0$, τέτοιο ώστε, $\forall m \geq k$, $\omega^m \models q$
	τότε	$\exists k \geq 0$, τ.ω. $\forall m \geq k$, $\omega^m \models \mathbf{F}\, p \vee q$
	αν και μόνο αν	$\exists k \geq 0$, τ.ω. $\forall m \geq 0$, $(\omega^k)^m \models \mathbf{F}\, p \vee q$
	αν και μόνο αν	w ⊨ **FG** (**F** p ∨ q)

v. **GF** p ∨ **FG** q → **FG** (**F** p ∧ q)

Αντιπαράδειγμα όπου ισχύει το αριστερό μέλος (**GF** p ∨ **FG** q) αλλά όχι το δεξί **FG** (**F** p ∧ q).



Άσκηση 5

(i) Γράφουμε ϕ_α για την πρόταση που εκφράζει ασθενή δικαιοσύνη σε σχέση με την ενέργεια a:

$$\phi_a = \text{FG enabled\_a} \rightarrow \text{GF executed\_a}$$

Τότε η ζητούμενη ιδιότητα είναι η

$$\phi_a \rightarrow \phi$$

(ii) Γράφουμε ψ_α για την πρόταση που εκφράζει ισχυρή δικαιοσύνη σε σχέση με την ενέργεια a:

$$\psi_a = \text{GF enabled\_a} \rightarrow \text{GF executed\_a}$$

Τότε η ζητούμενη ιδιότητα είναι η

$$\psi_a \rightarrow \phi$$