

## Φροντιστήριο 1 – Σκελετοί Λύσεων

### Λύση Άσκησης 1

(α) Χρησιμοποιούμε τις επιπλέον μεταβλητές PC1, PC2, (program counters) οι οποίες παίρνουν ως τιμές ονόματα των γραμμών του κώδικα όπως φαίνεται πιο κάτω.

```
Bool T[2] = [0,0];

while True do {
A1   T[0] = 1;
A2   while T[1] = 1 do { /* nothing */; }
A3   CRITICAL;
A4   T[0] = 0;
}
||
while True do {
B1   T[1] = 1;
B2   while T[0] = 1 do { /* nothing */; }
B3   CRITICAL;
B4   T[1] = 0;
}
```

Ο γράφος προγράμματος δίνεται από την πλειάδα  $(V, S, T, \rho)$  όπου

$$V = \{PC1, PC2, T[2]\}$$

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

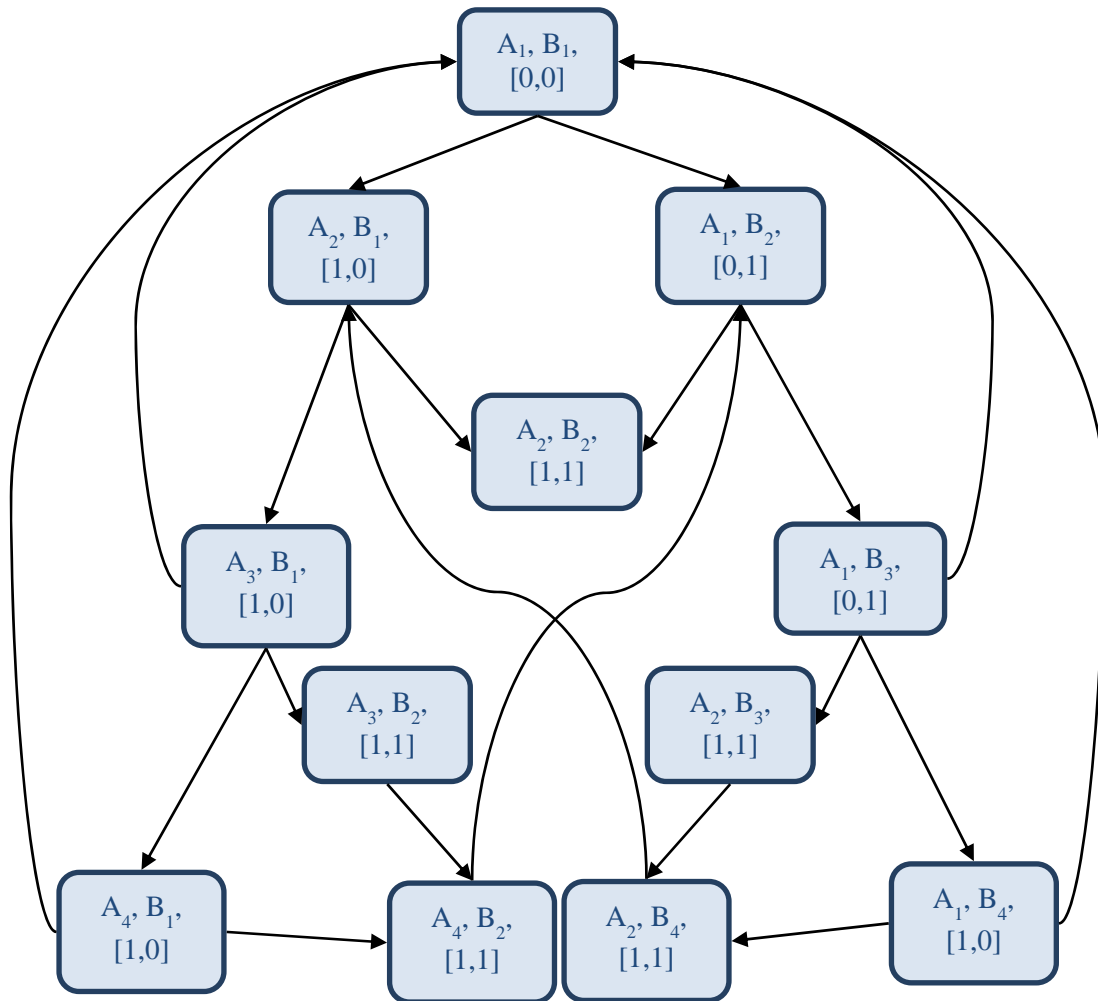
$T = \{$

T1:	PC1 = A1	→	(PC1, T[0]) := (A2, 1) ,
T2:	PC1 = A2 $\wedge$ (T[1] == 0)	→	PC1 := A3 ,
T3:	PC1 = A3	→	PC1 := A4 ,
T4:	PC1 = A4	→	(PC1, T[0]) := (A1, 0) ,
T5:	PC2 = B1	→	(PC2, T[1]) := (B2, 1) ,
T6:	PC2 = B2 $\wedge$ (T[0] == 0)	→	PC2 := B3 ,
T7:	PC2 = B3	→	PC2 := B4 ,
T8:	PC2 = B4	→	(PC2, T[1]) := (B1, 0) ,

$\}$

$$\rho \equiv (PC1 = A1) \wedge (PC2 = B1) \wedge (T = [0,0])$$

(β) Ακολουθεί το σύστημα μεταβάσεων όπου οι τριάδες  $(A, B, [x, y])$  αναφέρονται στις τιμές των μεταβλητών (PC1, PC2 και T) που ισχύουν στην κάθε κατάσταση.



(γ) Για να είναι κατάλληλος ο αλγόριθμος για διασφάλιση αμοιβαίου αποκλεισμού, θα πρέπει να είναι αδύνατο και οι δύο διεργασίες να βρίσκονται ταυτόχρονα στα σημεία του κώδικα  $A_3$  και  $B_3$ . Δηλαδή δεν θα πρέπει να υπάρχει κατάσταση στο σύστημα μεταβάσεων, όπου η μεταβλητή  $PC1 = A_3$  και η  $PC2 = B_3$ . Από τη μελέτη του συστήματος μεταβάσεων στο μέρος (β) οδηγούμαστε στο συμπέρασμα ότι είναι αυτό δεν είναι δυνατό. Επομένως, ο αλγόριθμος διασφαλίζει αμοιβαίο αποκλεισμό. Εντούτοις, ο αλγόριθμος είναι δυνατό να εμφανίσει αδιέξοδα: επιτρέπει στις διεργασίες να εισέλθουν σε μια κατάσταση όπου  $T=[1,1]$  από όπου καμιά διεργασία δεν είναι σε θέση να εισέλθει στο κρίσιμο της τμήμα αφού οι συνθήκες των μεταβάσεων  $T_2$  και  $T_6$  δεν ικανοποιούνται.

## Λύση Άσκησης 2

(α) Χρησιμοποιούμε τις επιπλέον μεταβλητές PC1, PC2, (program counters) οι οποίες παίρνουν ως τιμές ονόματα των γραμμών του κώδικα όπως φαίνεται πιο κάτω.

```
X := 0; P := 0; in:=0; out := 0;
```

```
producer
```

```
  P1 while true do
  P2   produce?in
  P3   X := X + 1;
  P4   if X > 1
  P5     consume!in
  endwhile
```

```
||
```

```
consumer
```

```
  C1 while true do
  C2   consume?out
  C3   if (X > 1)
  C4     P := P + 1;
  C5     X := X - 1;
  endwhile
```

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

$$V = \{PC1, PC2, X, P, in, out\}$$

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

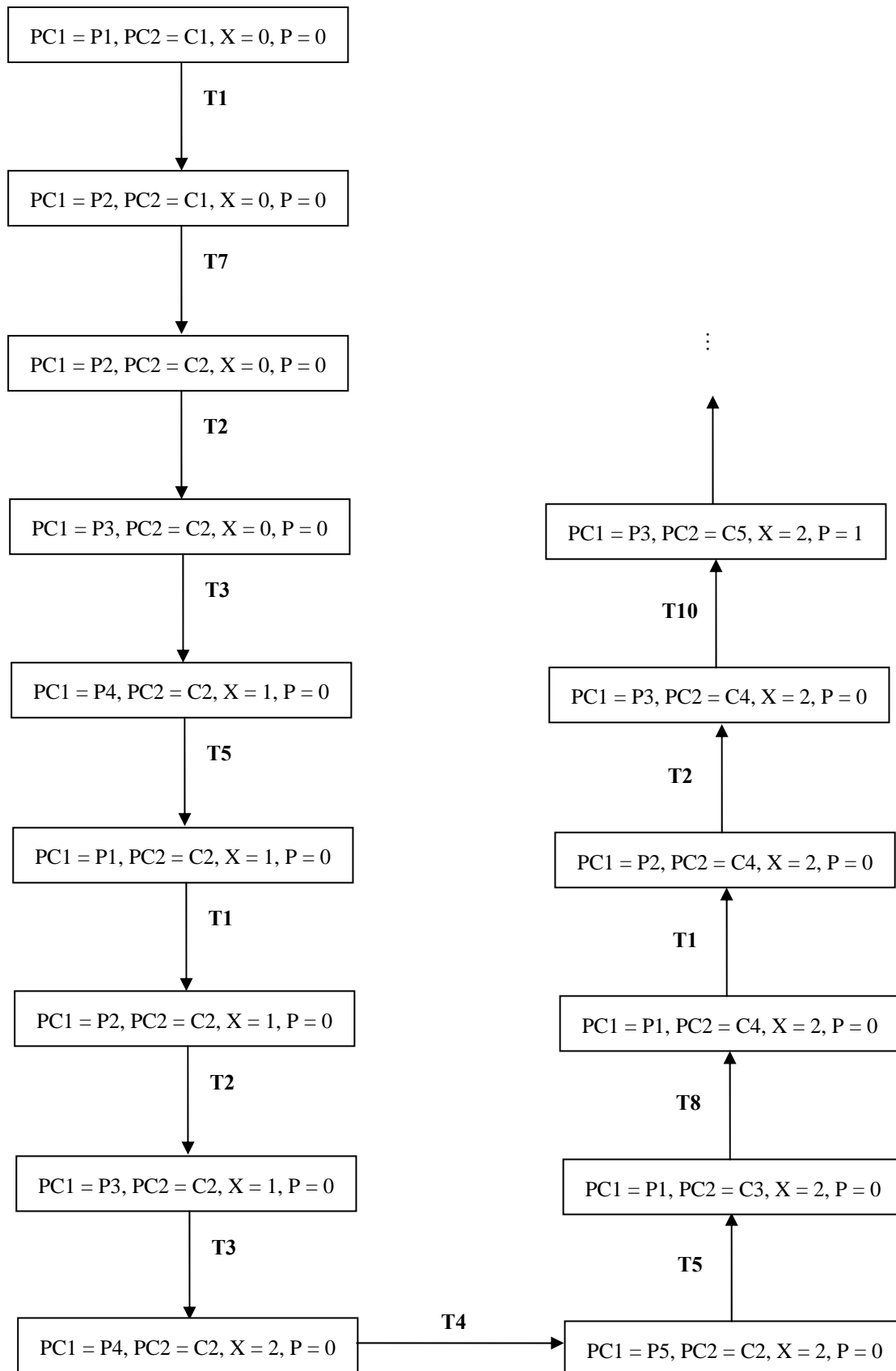
T = {

T1:	PC1= P1	→	PC1 := P2	,
T2:	PC1= P2, produce?in	→	PC1 := P3	,
T3:	PC1= P3	→	(PC1,X) := (P4, X+1)	,
T4:	PC1= P4, X>1	→	PC1 := P5	,
T5:	PC1= P4, X<=1	→	PC1 := P1	,
T6:	PC1= P5, PC2 = C2	→	(PC1, PC2) := (P1, C3)	,
T7:	PC2= C1	→	PC2 := C2	,
T8:	PC2= C3, X > 1	→	PC2 := C4	,
T9:	PC2= C3, X <= 1	→	PC2 := C1	,
T10:	PC2= C4	→	(PC2, P) := (C5, P + 1)	,
T11:	PC2= C5	→	(PC2, X) := (C1, X - 1)	

}

$$p = X = 0 \wedge P = 0 \wedge PC1 = P1 \wedge PC2 = C1$$

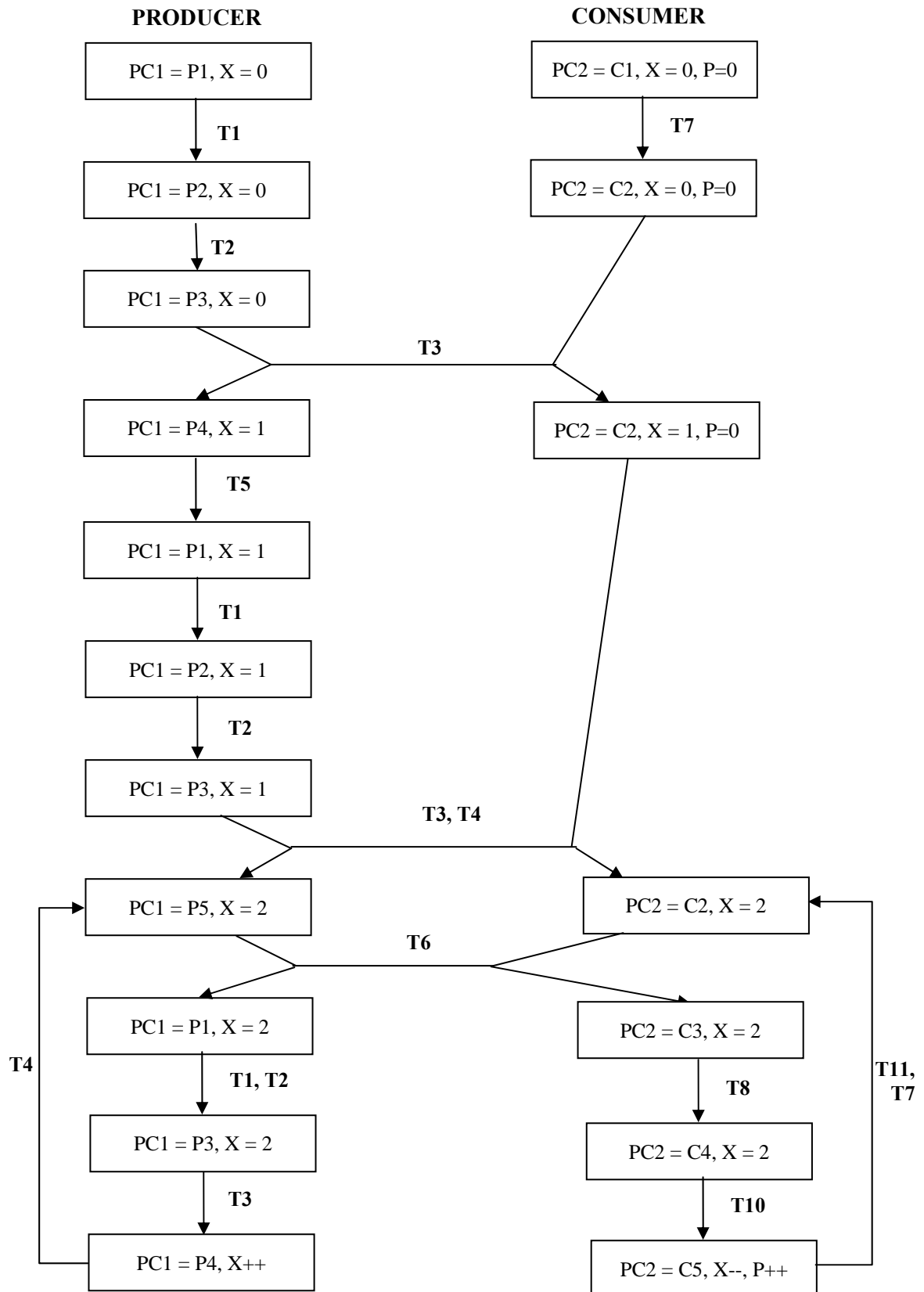
Πιο κάτω φαίνεται μια δυνατή εκτέλεση του συστήματος μεταβάσεων:



(β) Στο μοντέλο του συστήματος με σημασιολογία μερικής διάταξης θα πρέπει να ορίσουμε τη σχέση ανάμεσα στα διάφορα γεγονότα που μπορούν να λάβουν χώρα στο σύστημα. Συγκεκριμένα, γράφουμε  $T_j^i$  για την εκτέλεση της μετάβασης  $T_j$  κατά την  $i$ -οστή εκτέλεση του βρόχου, και έχουμε ότι για κάθε  $i$

$$\begin{aligned} T_1^i &< T_2^i < T_3^i < T_4^i < T_6^i < T_1^{i+1} \\ T_3^i &< T_5^i < T_1^{i+1} \\ &\text{και} \\ T_7^i &< T_6^i < T_8^i < T_{10}^i < T_{11}^i < T_7^{i+1} \\ T_6^i &< T_9^i < T_7^{i+1} \end{aligned}$$

Γραφικά, το μοντέλο του συστήματος φαίνεται στο πιο κάτω σχήμα.



### Λύση Άσκησης 3

Χρησιμοποιούμε τις επιπλέον μεταβλητές PC1, PC2, (program counters) οι οποίες παίρνουν ως τιμές ονόματα των γραμμών του κώδικα όπως φαίνεται πιο κάτω.

```
left
  l1: if (y1=n-k) then stop;
  l2: y3 := y3*y1;
  l3: y1--;
  l4: goto l1
```

||

```
right
  r1: if (y2=k) then stop;
  r2: y2++;
  r3: await y2 <= n-y1
  r4: y3 := y3/y2;
  r5: goto r1;
```

Ο γράφος προγράμματος δίνεται από την πλειάδα (V, S, T, ρ) όπου

$$V = \{PC1, PC2, y1, y2, y3, n, k\}$$

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

T = {

T1:	PC1= l1, y1 = n-k	→	stop ,
T2:	PC1= l1, y1 != n-k	→	PC1 := l2 ,
T3:	PC1= l2	→	(PC1, y3) := (l3 , y3*y1) ,
T4:	PC1= l3	→	(PC1, y1 := (l4 , y1-1) ,
T5:	PC1= l4	→	PC1 := l1 ,
T6:	PC2= r1, y2 = k	→	stop ,
T7:	PC2= r1, y2 != k	→	PC2 := r2 ,
T8:	PC2= r2	→	(PC2, y2) := (r3, y2+1) ,
T9:	PC2= r3, y2 <= n-y1	→	PC2 := r4 ,
T10:	PC2= r4	→	(PC2, y3) := (r5, y3/y2) ,
T11:	PC2= r5	→	PC2 := r1

}

$$\rho = y1 = n \wedge y2 = 0 \wedge y3 = 1 \wedge PC1 = l1 \wedge PC2 = r1, \quad n > 0, k > 0$$