

Question 1 On suppose qu’on a un répondeur qui peut enregistrer un seul appel, et qu’un appel enregistré peut être écouté une seule fois. Le répondeur peut être dans un des modes *ready* (S), *recording* (R), *playing* (P) ou *message available* (M). Le répondeur a une lampe qui s’allume après l’enregistrement d’un message, et reste allumée jusqu’il soit écouté.

Modélisez le répondeur par un système de transitions.

Question 2 Soit P le programme suivant :

```

for k = 1 to 10 do {
  read(x);
  x := x+1;
  write(x);
}

```

Si la valeur initiale de x est 0, est-ce que le programme $P \parallel P$ a une exécution maximale à la fin de laquelle la valeur de x est 2 ?

Question 3 Le programme suivant est un protocole d’exclusion mutuelle pour 2 processus. Le protocole utilise les variables partagées booléennes b_1, b_2 , et la variable partagée x , dont les valeurs possibles sont 1, 2. Le programme P_i , $i = 1, 2$, est

```

loop forever do {
1:  noncrit-section;
2:   $b_i := \text{true}$ ;
3:   $x := 3 - i$ ;
4:  wait until  $((x = i) \parallel \neg b_{3-i})$ ;
5:  crit-section;
6:   $b_i := \text{false}$ ;
}

```

1. Dessinez le ST de $P_1 \parallel P_2$.
2. Vérifiez si le protocole satisfait l’exclusion mutuelle.
3. Vérifiez si le protocole est équitable (aucun processus ne peut rester pour toujours dans la boucle “wait”).

Question 4 Le programme suivant est un protocole d’exclusion mutuelle pour les processus P_1, P_2 , avec variables partagées entières y_1, y_2 (initialement 0).

```

Pi
loop forever do {
1:   noncrit-section;
2:   yi := y3-i + 1;
3:   wait until ((y3-i = 0) || yi < y3-i);
4:   crit-section;
5:   yi := 0;
}

```

1. Construisez la partie accessible du ST de $P_1 \parallel P_2$ pour $y_1 \leq 2$ et $y_2 \leq 2$.
2. Est-ce que le ST est infini ?
3. Vérifiez que le protocole satisfait l'exclusion mutuelle.
4. Vérifiez si le protocole peut atteindre une configuration où les 2 processus s'attendent mutuellement.
5. Vérifiez si le protocole est équitable.