Centre Universitaire de Mila

3 ème année licence LMD Informatique

# Module : Systèmes d'exploitation 2

Bessouf Hakim

# CHAPITRE 3:

# L’interblocage

- Modèles
- Prévention
- Évitement
- Détection/ Guérison
- Approche combinée

# Introduction

- Dans les système informatiques actuels différents processus s'exécute en parallèle et utilisent des ressources communes,. Dans le cas du dîner de philosophes les processus se retrouvent bloqué l'un attendant l’autre. Cette situation est appelé interblocage (deadlock). Un interblocage peut se provoquer entre deux ou plusieurs processus partageant deux ou plusieurs ressources.

# Les ressources

- Les ressources peuvent être matériels comme les imprimantes, scanners, graveurs DVD, mémoire … etc, ou logiciel comme les fichiers par exemple.
- Certaine ressource comme la mémoire peuvent être retirer aux processus sans provoquer d'erreur, ils sont dits préemptibles , d'autre sont non préemptibles comme c'est la cas des graveurs DVD.
- Chaque processus qui veut utiliser une ressource doit suivre les étapes suivantes :
- 1) Demander la ressource du système
- 2) Utiliser la ressource
- 3) Libérer la ressource

# Condition nécessaires a l'interblocage

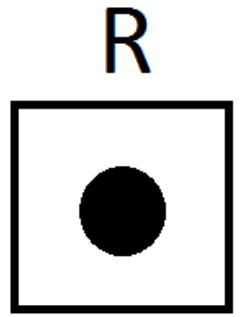
- Pour provoquer un interblocage les conditions suivantes doivent être remplies :

1. **Condition d'exclusion mutuel:** une ressource doit être attribué a un seul processus à la fois,
2. **Condition de détention et attente:** un processus détiens une ressource au moins et attend l’acquisition d'une autre ressource qui est utilisé par un autre processus, un processus qui a demandé des ressources peut donc demander des ressources supplémentaires,
3. **Condition de préemption (réquisition):** une ressource ne peut être retiré d'un processus jusqu’à ce qu'il la libère,
4. **Condition d'attente circulaire:** il doit y avoir un cycle entre les processus (deux processus au moins) dans lequel l’un attend l'autre.

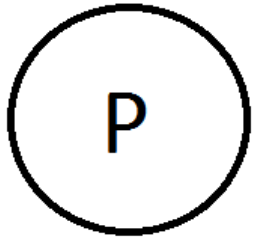
# Graphe d'allocation des ressources

- Pour modéliser les interblocage on utilise un graphe d'allocation des ressources:

- Les ressources sont représentés par des carrés,

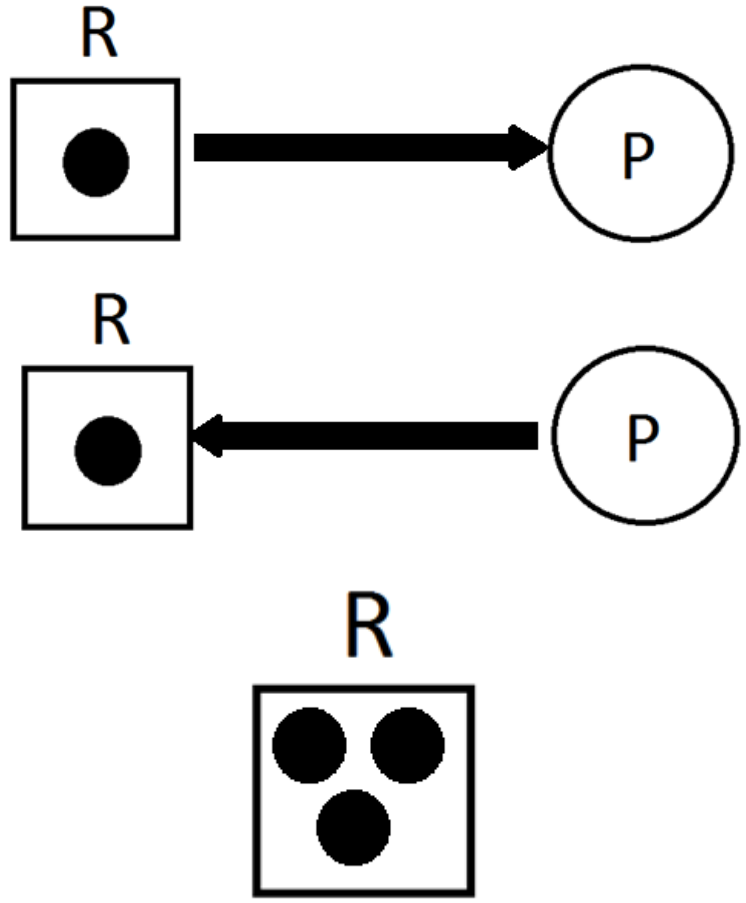



- Les processus sont représentés par des cercles,



# Graphe d'allocation des ressources

- Un arc allant d'une ressource à un processus signifie que cette ressources est attribuée à ce processus,
- Un arc allant d'un processus à une ressource signifie que le processus a demandé cette ressource et qu'il est bloqué en attendant que le système lui attribut la ressource,
- Une ressource peut avoir plusieurs instances,



# Graphe d'allocation des ressources

- **Exemple:**
- On considère un graphe d'allocation des ressources représenté par les ensembles P , R et E.
- P = {P1 , P2 , P3} //processus R = {R1, R2 , R3 , R4} //ressources
- E = { R1 → P1 , R1 → P2 , R2 → P2 , P2 → R4 ,
- R4 → P3} //demandes et attributions
- P1,P2, et P3 sont des processus, R1,R2 ,R3 et R4 sont des ressources.

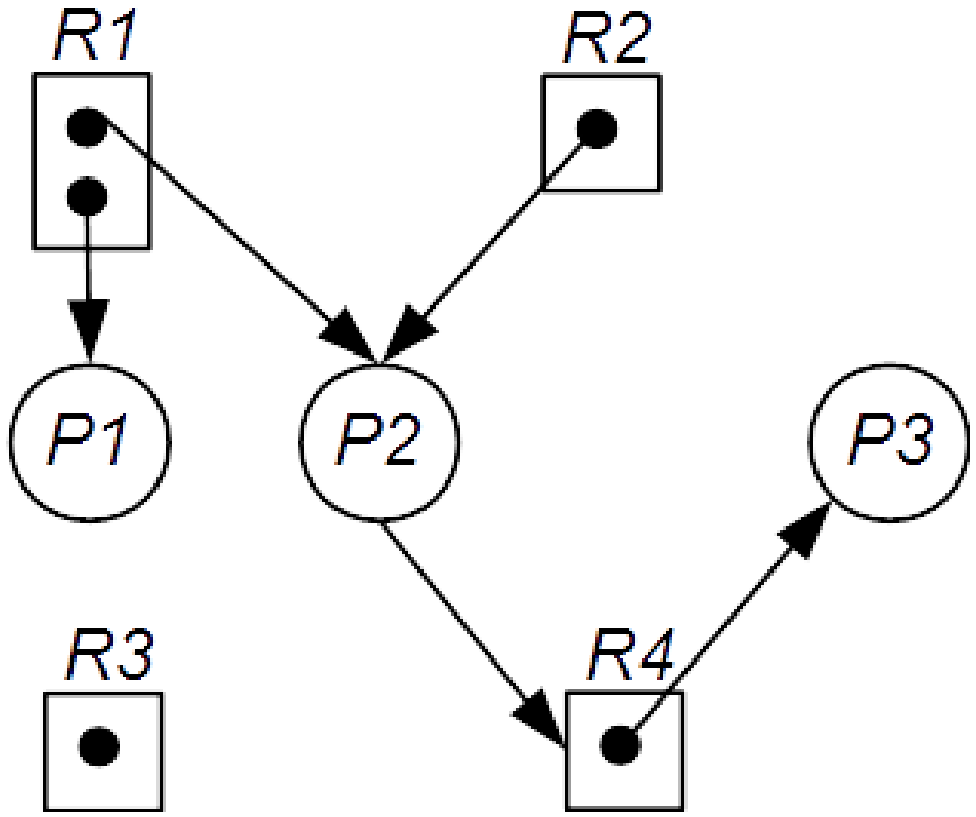
- R2 → P2 : veut dire que le P1 détient une instance de la ressource R2 ( une instance de R2 est alloué à P2),
- P2 → R4 : veut dire que P1 attend une instance de R4.

# Graphe d'allocation des ressources

P = {P1 , P2 , P3} //processus

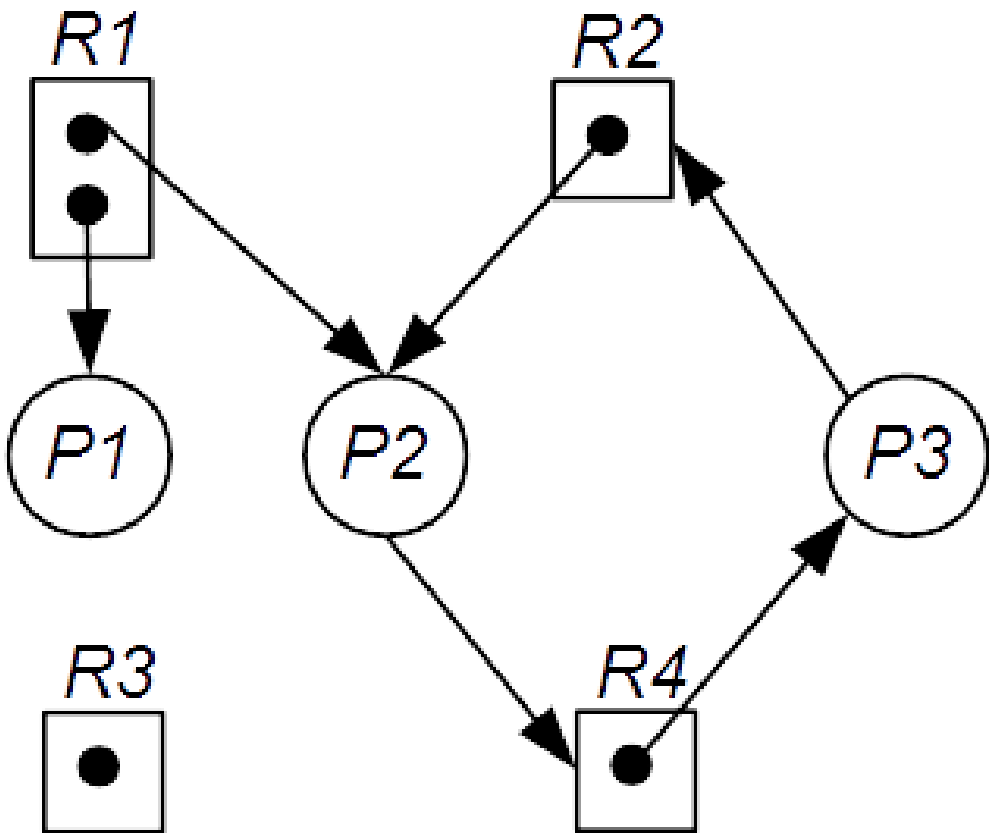
R = {R1, R2 , R3 , R4} //ressources

E = { R1 → P1 , R1 → P2 , R2 → P2 ,
P2 → R4 , R4 → P3}



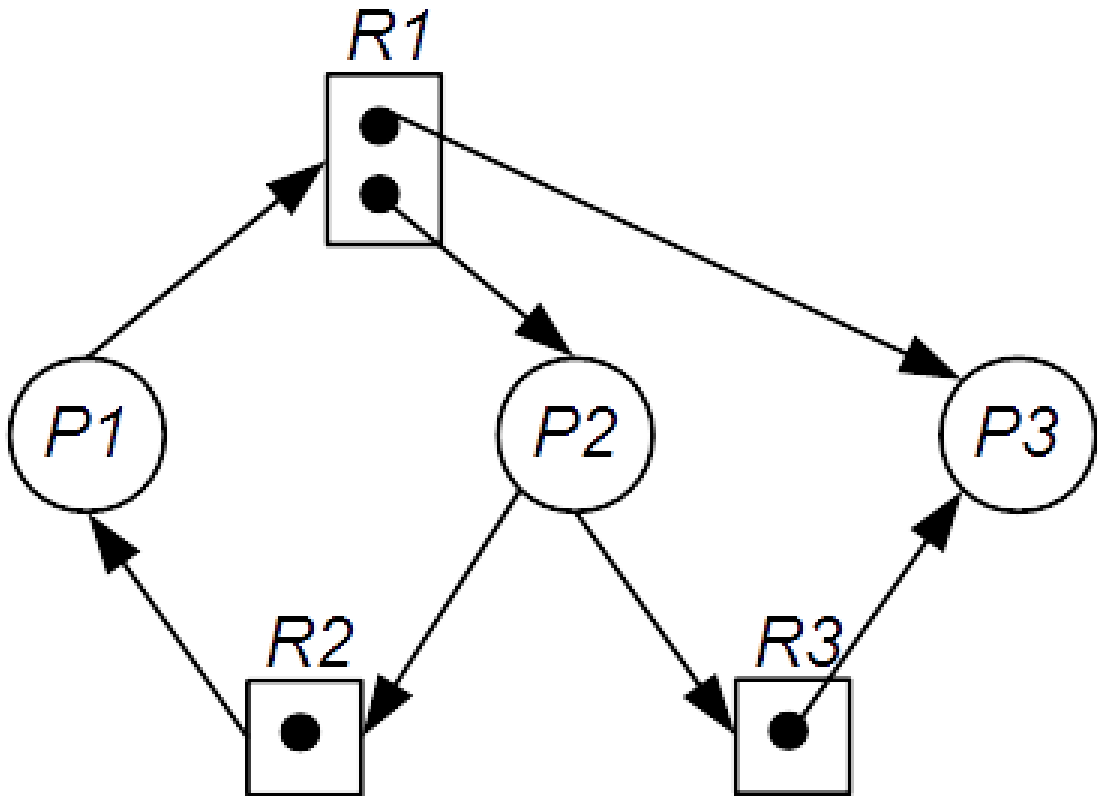
# Graphe d'allocation des ressources

- Le graphe précédent ne contiens pas de cycles il n'y a pas donc d'interblocage, mais si par exemple P3 demande une instance de R2 ( P3 → R2) on aura une situation d'interblocage :
- (R2 → P2 , P2 → R4 , R4 → P3 , P3 → R2)



# Graphe d'allocation des ressources

- Si les ressources ont plusieurs instance la présence de cycle ne veut pas forcément dire qu'il y a une situation d'interblocage comme c'est le cas du graphe suivant :



# Méthodes pour traiter l'interblocage

- Pour traiter les interblocages on peut suivre les méthodes suivantes :

1) Ignorer complètement le problème et supposer que les interblocages ne se produisent jamais,
2) Utiliser des protocoles pour prévenir l'interblocage en empêchant l'apparition d'au moins une condition d'interblocage,
3) Laisser le système tomber dans un état d'interblocage et le corriger ensuite.
4) Éviter l'interblocage d'une manière dynamique et cela en allouant les ressources avec précaution,

# Politique de l'autruche

- Cette méthode consiste simplement à ne rien faire et d'ignorer complètement le problème en supposant qu'il ne se produira jamais.
- Cette politique est utilisé par la plus part des systèmes d'exploitation. Si les interblocages ne se produisent que rarement cette méthode peut être très efficace puisque elle ne consommer aucune ressource supplémentaire.

# Prévention de l'interblocage

- Puisque les quatre conditions précédentes sont nécessaires pour que l'interblocage se produit, il suffit donc de garantir qu'au mois une de ces conditions n'est pas satisfaite pour garantir que l'interblocage ne se produise pas.
- **S'attaquer à la condition d'exclusion mutuelle:**

  Si les ressources sont **partageables** entre les processus il n y aura jamais d'interblocage. Mais dans la pratique on ne peut pas nier cette condition puisque certaine ressources sont impossible à partager (imprimantes par exemple).

# Prévention de l'interblocage

- **S'attaquer à la condition de détention et attente**

  Une première méthode consiste à obliger le processus à allouer toutes les ressources qu'il a besoin au moment de l’allocation, s’il y a des ressources manquantes aucune ressource ne sera alloué au processus ( **tous ou rien**).

  Une deuxième méthode consiste à interdire qu'un processus n'occupe plus de ressources, On ne vas pas allouer d'autres ressources à un processus tant qu'il n'a pas libéré toutes les ressources déjà occupés.

# Prévention de l'interblocage

Les deux méthode précédentes garantisse que l'interblocage ne se produise pas mais ont certain inconvénients:

L’utilisation des ressource est inefficaces, les ressource seront longtemps inutilisés, un processus peut attendre infiniment si au moins une des ressources dont il a besoin est toujours alloué (famine).

# Prévention de l'interblocage

**S'attaquer à la condition de non-préemption**

Dans ce cas si un processus est en attente d’une ressource supplémentaires les ressources qu'il détiens peuvent lui être retirés si elles sont demandés par d'autres processus.

Cette méthodes peut être utilisé seulement si l'état des ressources peut être sauvegardé et restauré et ne peut être appliqué à des ressources comme les imprimantes ou les graveurs DVD par exemple.

# Prévention de l'interblocage

**S'attaquer à la condition de l'attente circulaire**

- Une méthode consiste assigner a chaque ressource un numéro, et d'ordonner les ressources selon ce numéro. Par exemple :

1. Lecteur CD , 2. Scanner , 3. Imprimante , 4. ...

- Un processus ne peut pas demander une nouvelle ressource que si elle a un numéro plus grand que les ressource déjà obtenus.
- dans notre cas un processus ne peut pas demander le lecteur DVD s'il détiens l'imprimante.
- Cette méthode impose un ordre totale d'allocation des ressource qui évite la formation de cycle et par conséquent évite l'interblocage.
- Cette méthode peut ne pas être applicable si le nombre de ressources est grand puisque aucun ordonnancement ne peut fonctionner.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- Dans cette méthode le système laisse l'interblocage se produire puis il corrige la situation après, il ne cherche pas donc à empêcher l'interblocage.

**Cas des ressources avec une seule instance**

- Un cycle dans le graphe veut dire qu'un interblocage s'est produit, on peut donc utiliser un algorithme de détection des cycles d'un graphe pour détecter l'interblocage.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme de détection d'interblocage : *[Ref:Systèmes d'exploitation Andrew Tanenbaum p : 467]*

Soit **L** une liste de nœuds.

1) Pour chaque nœud N dans le graphe suivre les étapes suivantes, avec N comme point de départ,
2) Initialiser L à liste_vide et désigné tous les arcs comme non marqués,
3) Ajouter le nœud en cours à la fin de la liste L : Si le nœud apparaît deux fois à la liste L alors le graphe contiens un cycle (indiqué par L) terminer donc l'algorithme, Sinon continuer les étapes,
4) Si pour un nœud donné il y a un arc sortant non marqué alors aller à l'étape 5 Sinon aller à l'étape 6,
5) Choisir au hasard un arc sortant non marqué, suivre cette arc pour arriver à un nouveau nœud, maquer l'arc choisit, et aller a l'étape 3,
6) Si le nœud est le nœud initial alors le graphe ne contiens pas de cycle terminer donc l'algorithme, Sinon supprimer l'arc emprunté , revenir au nœud précédent et aller a l'étape 4.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

➢ Étape 2

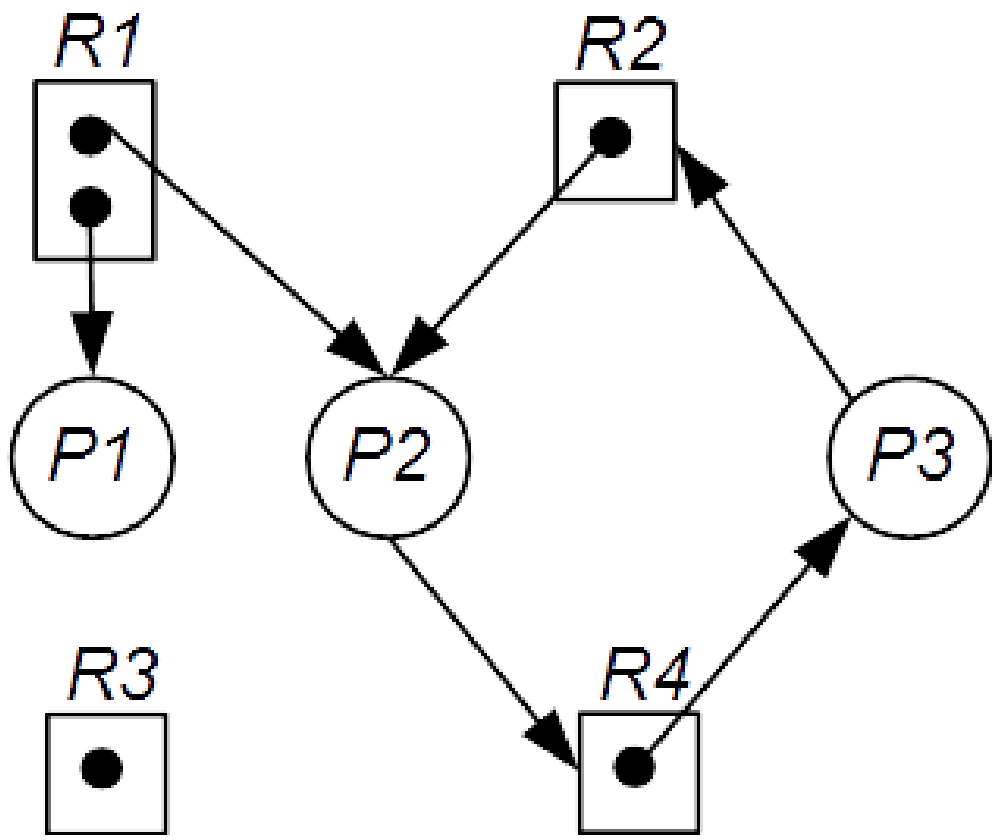
Étape 3

Étape 4

Étape 5

Étape 6

L = liste_vide



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

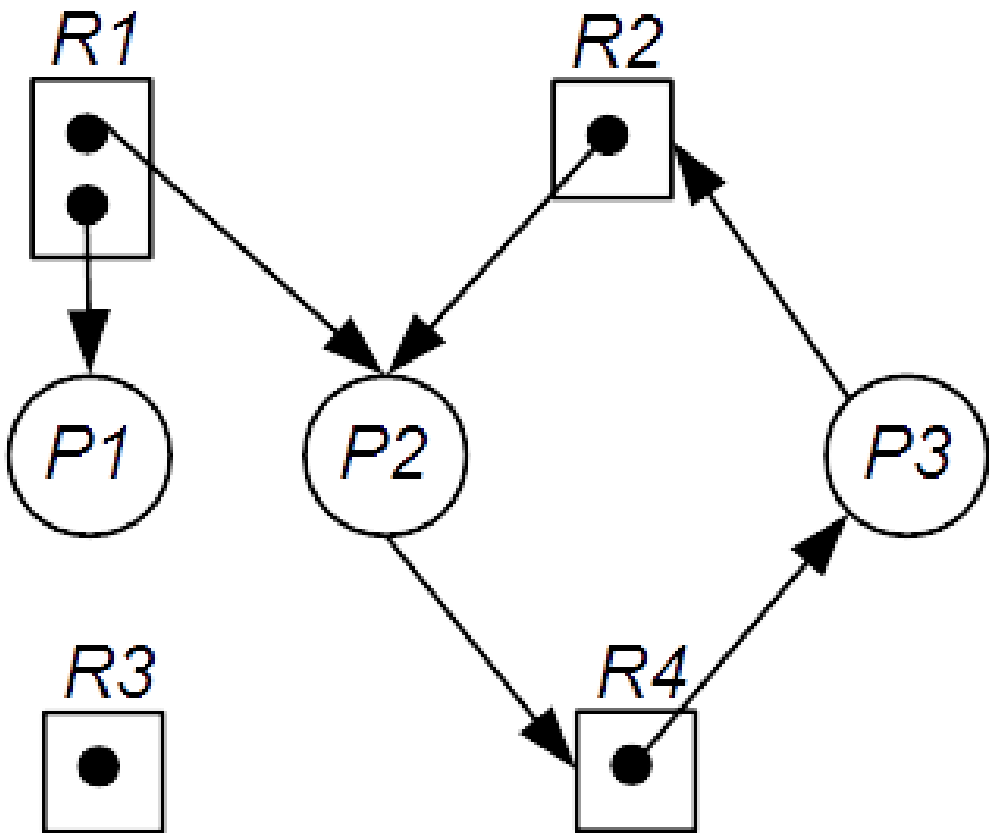
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

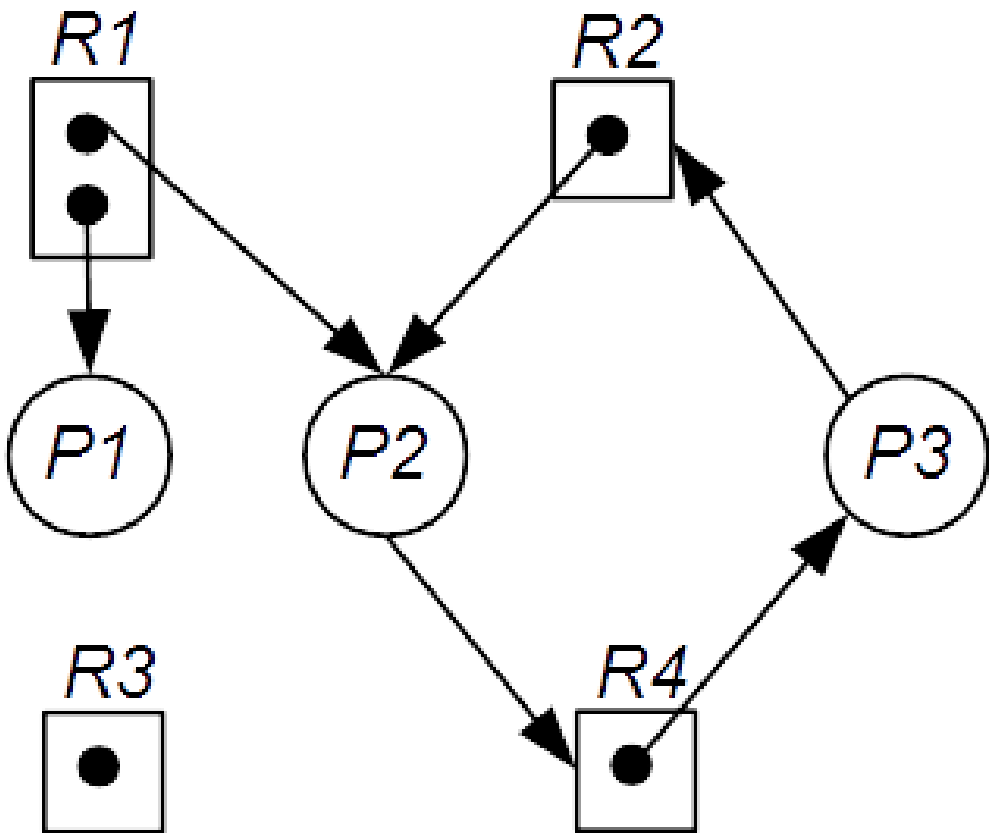
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

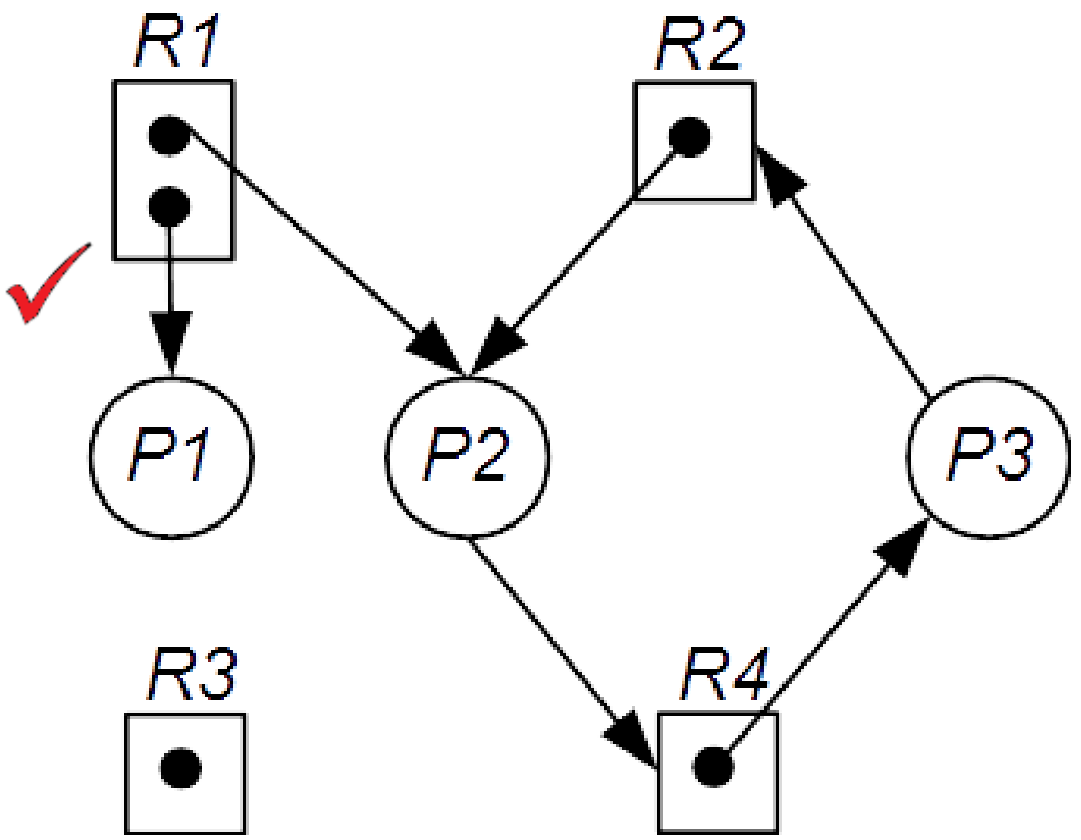
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

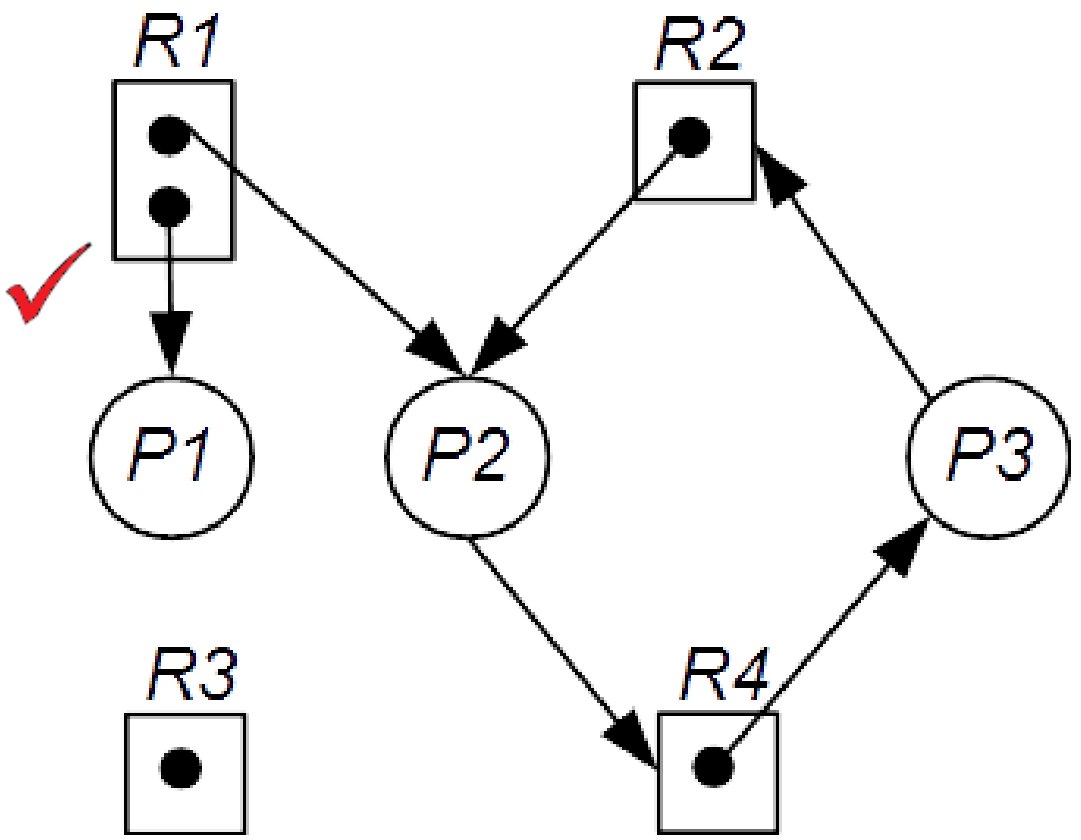
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

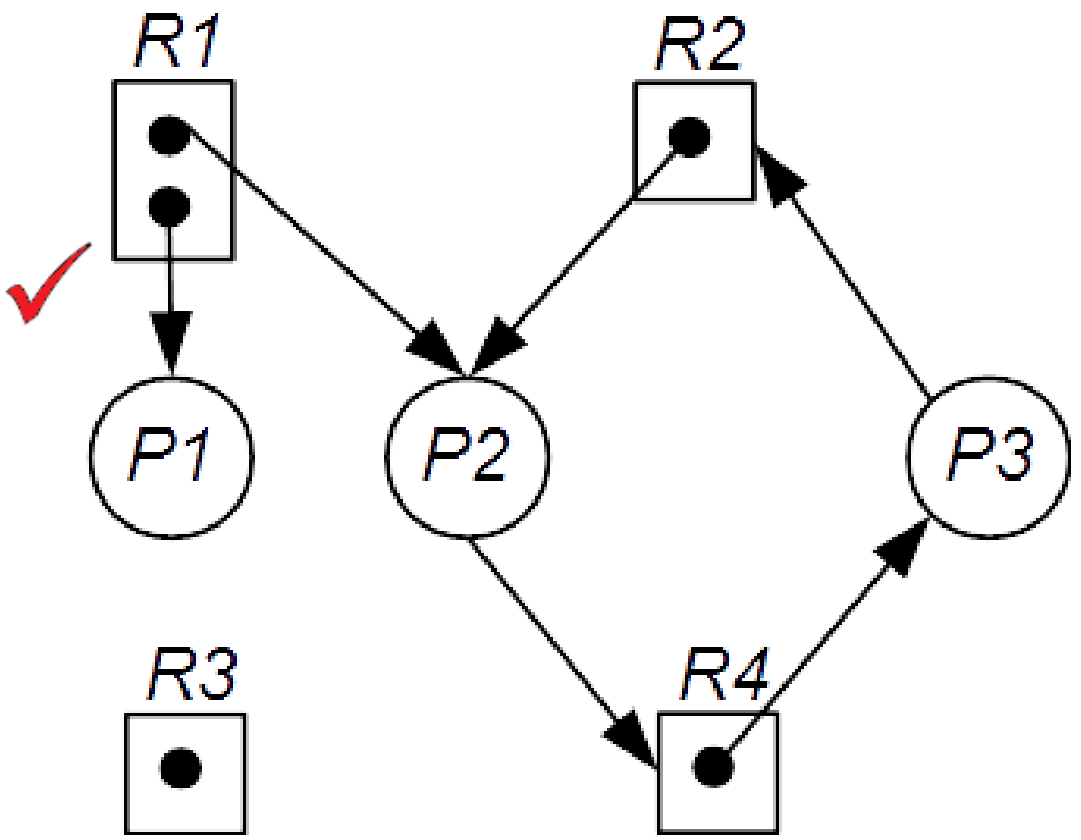
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1,P1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

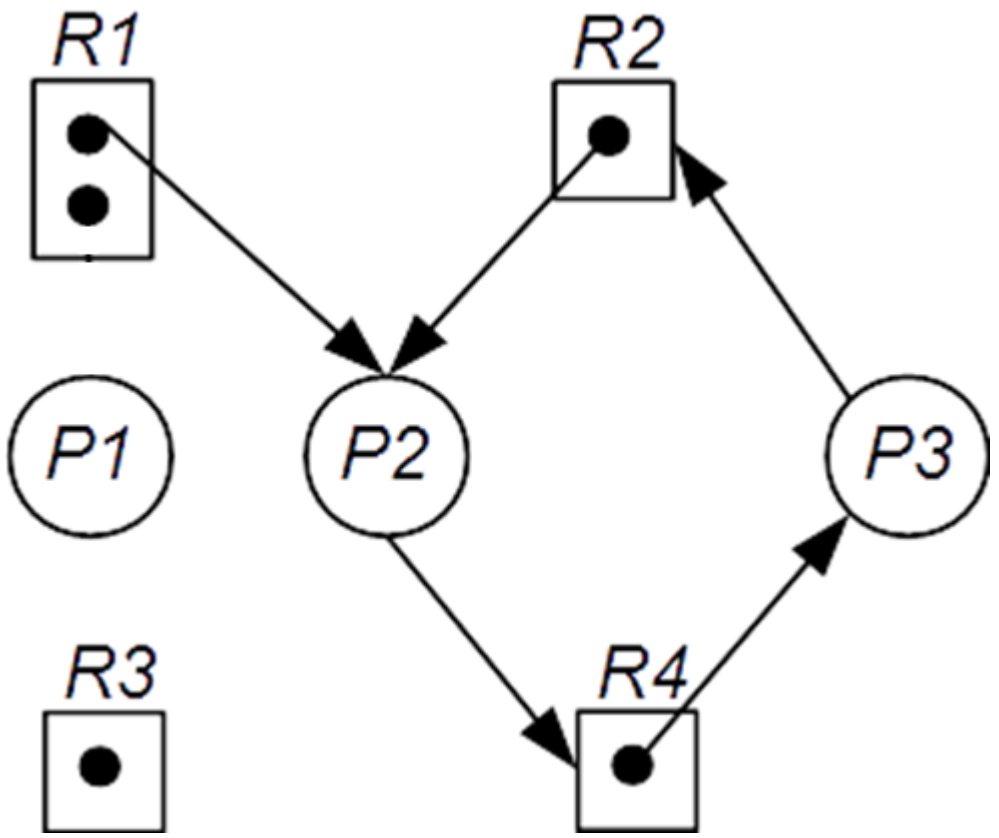
Étape 3

Étape 4

Étape 5

➢ Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

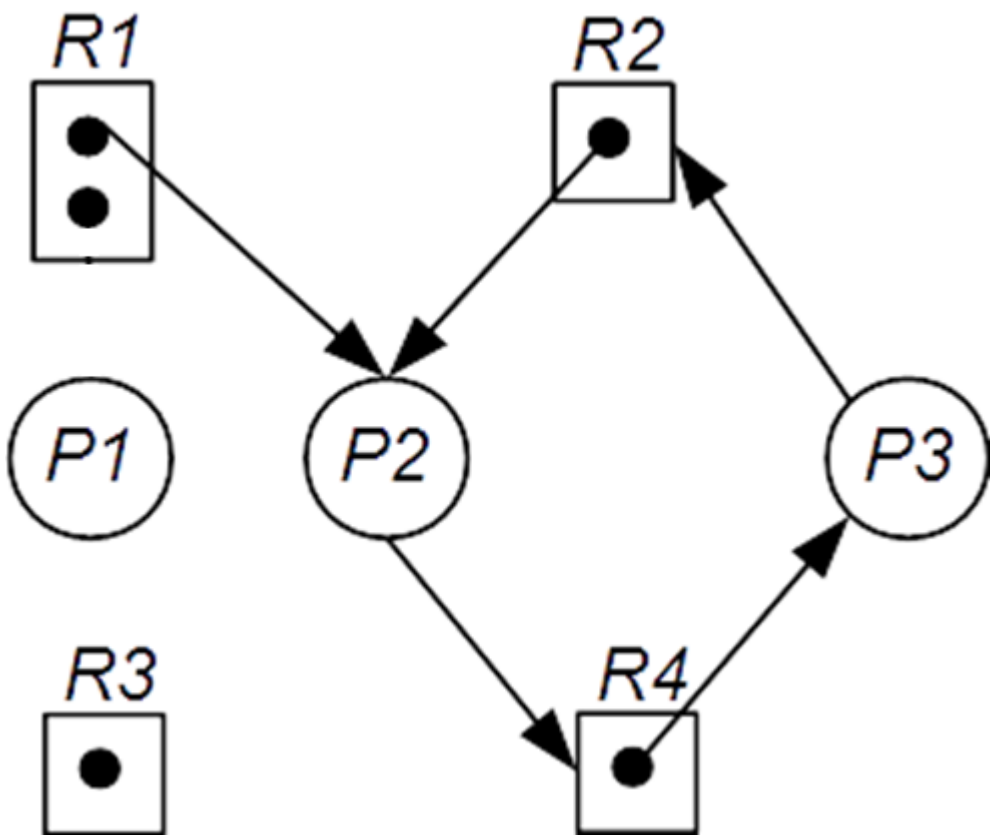
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

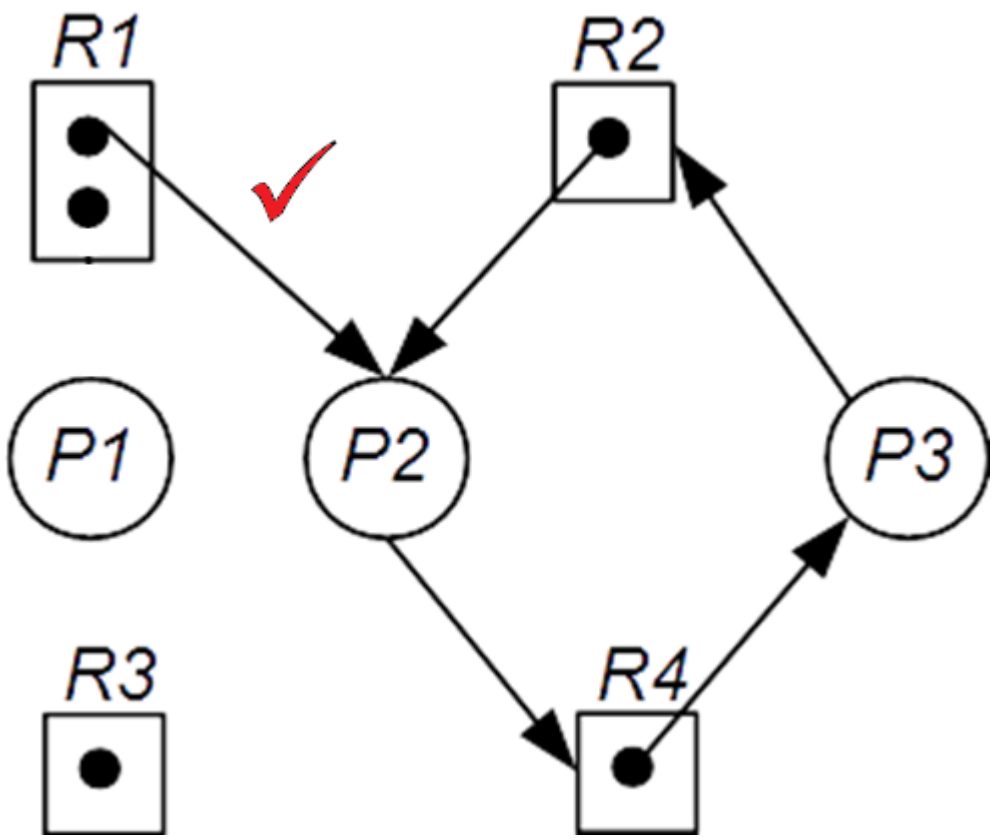
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

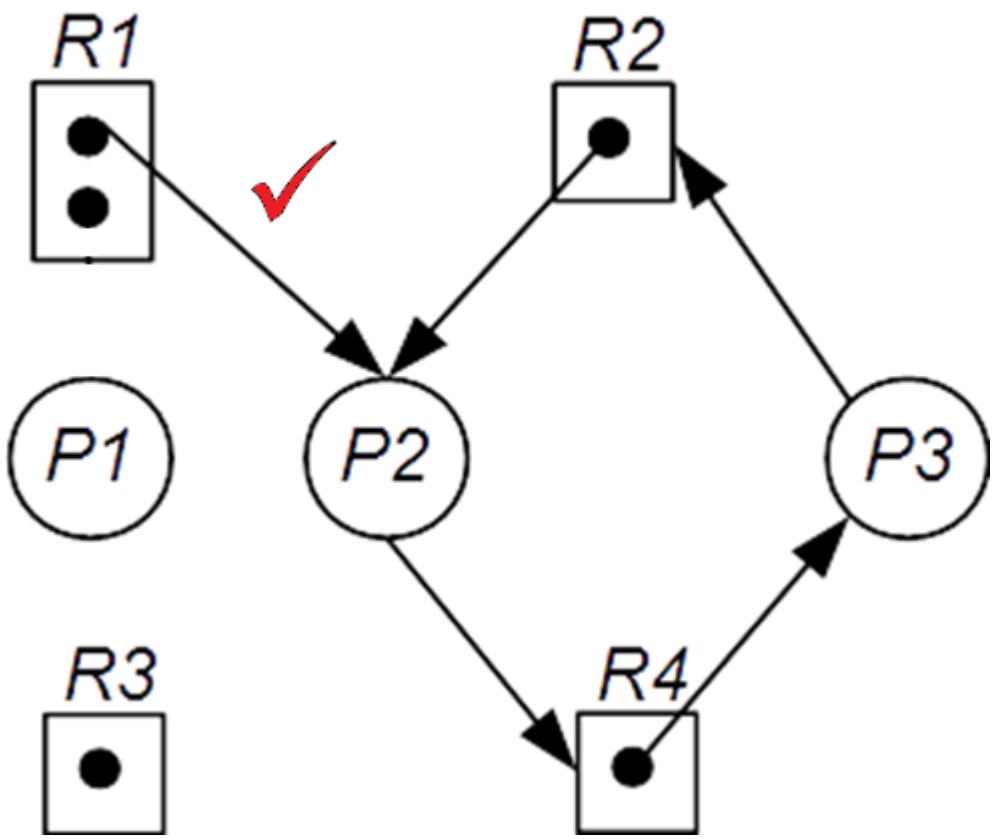
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

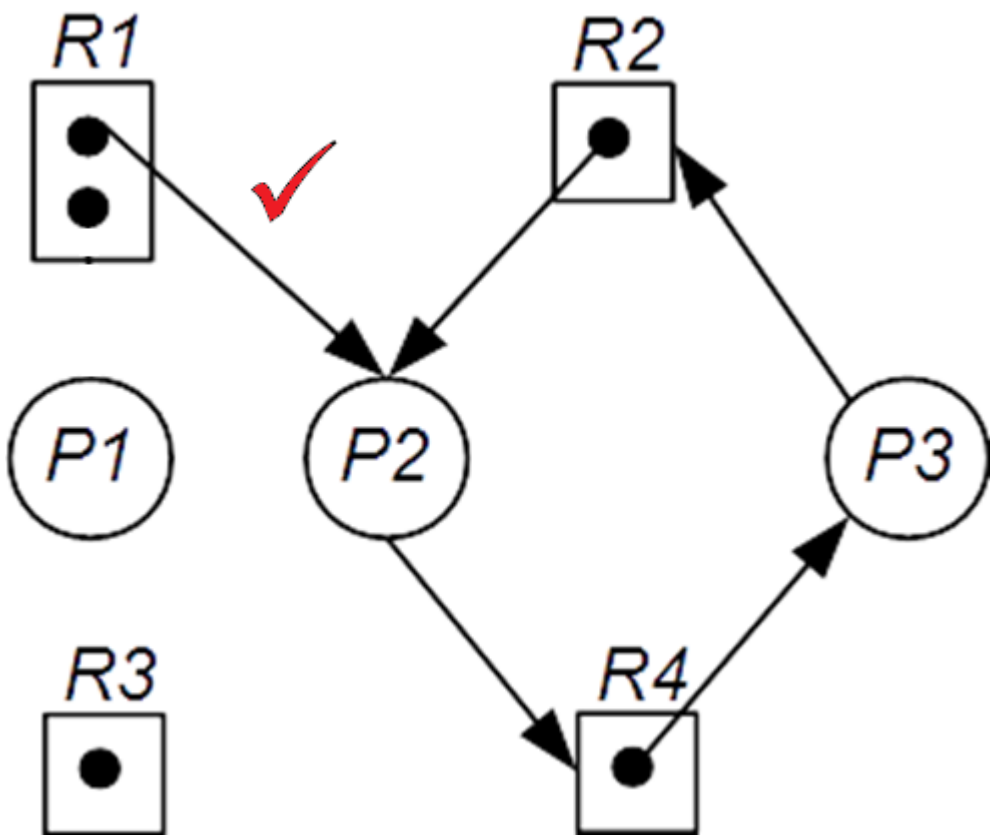
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1,P2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

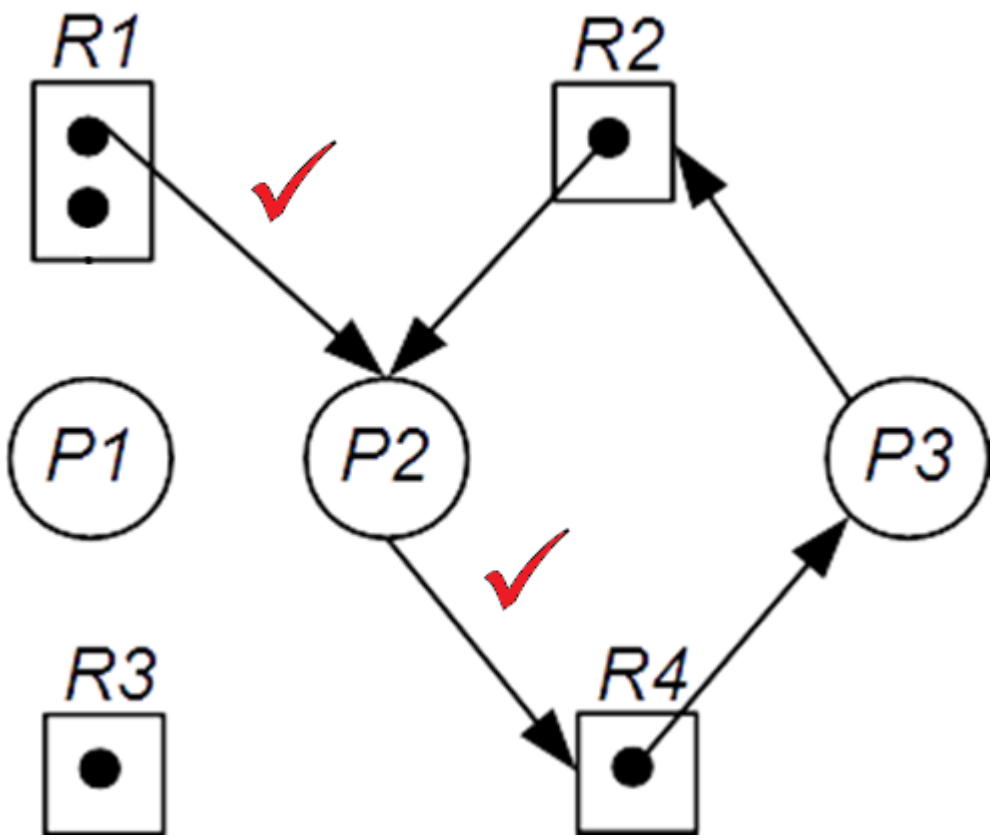
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1,P2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

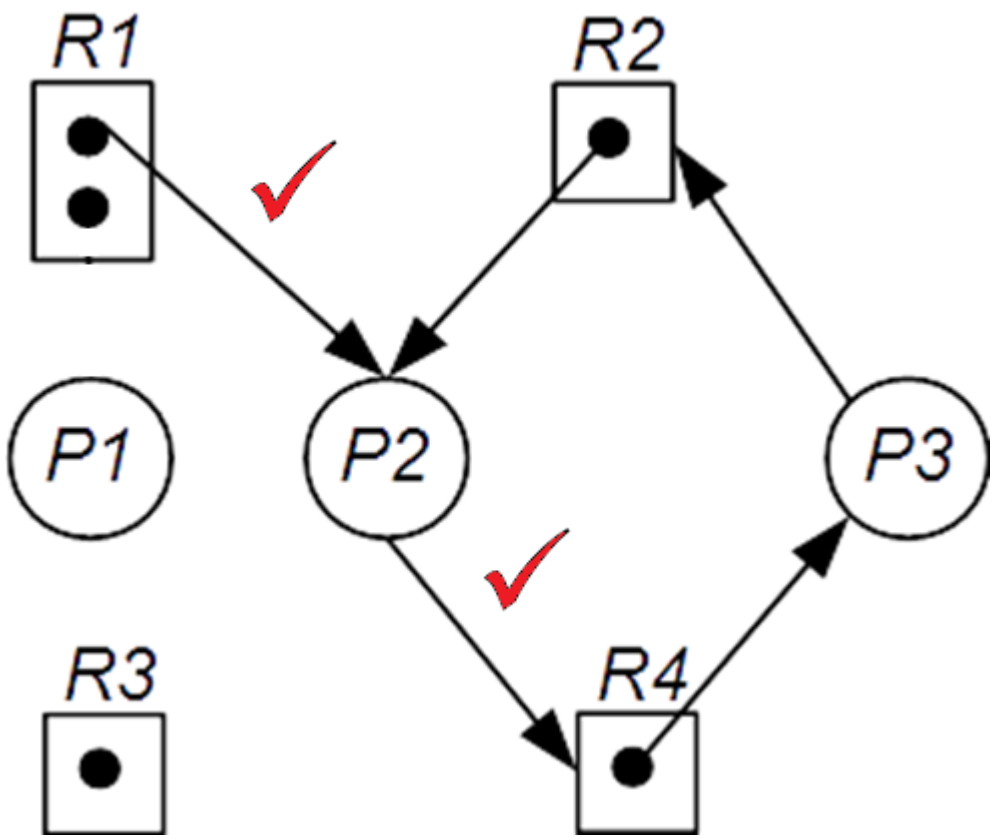
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P2,R4



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

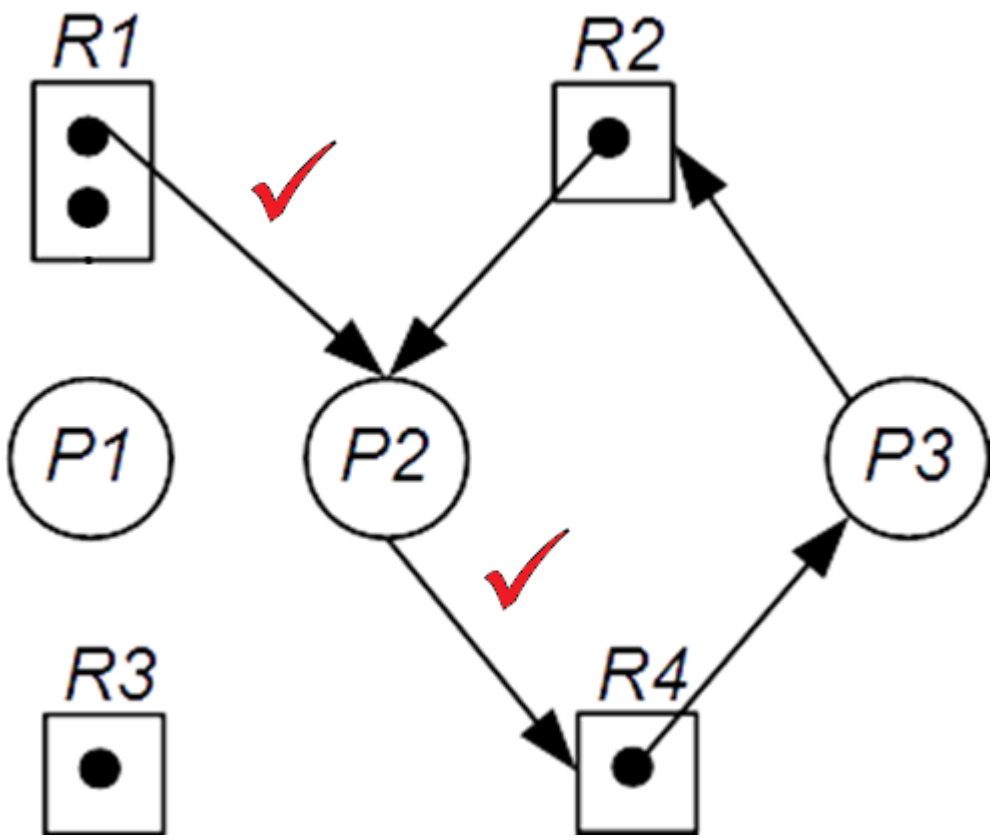
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1,P2,R4



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

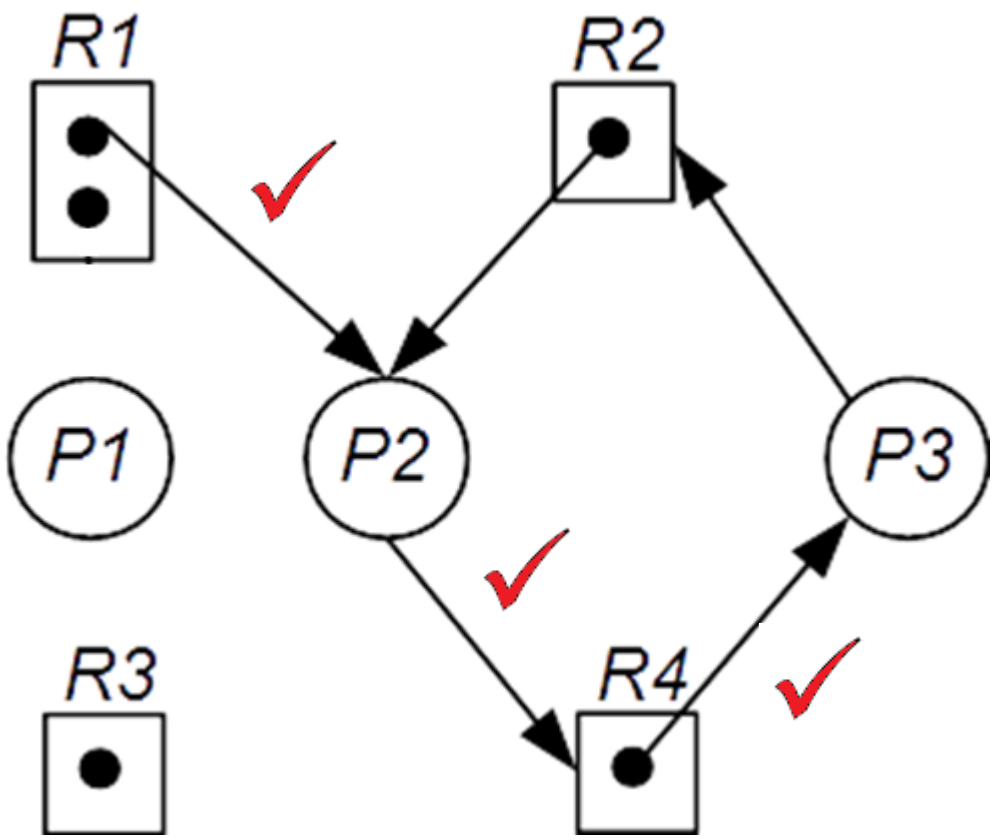
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1,P2,R4



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

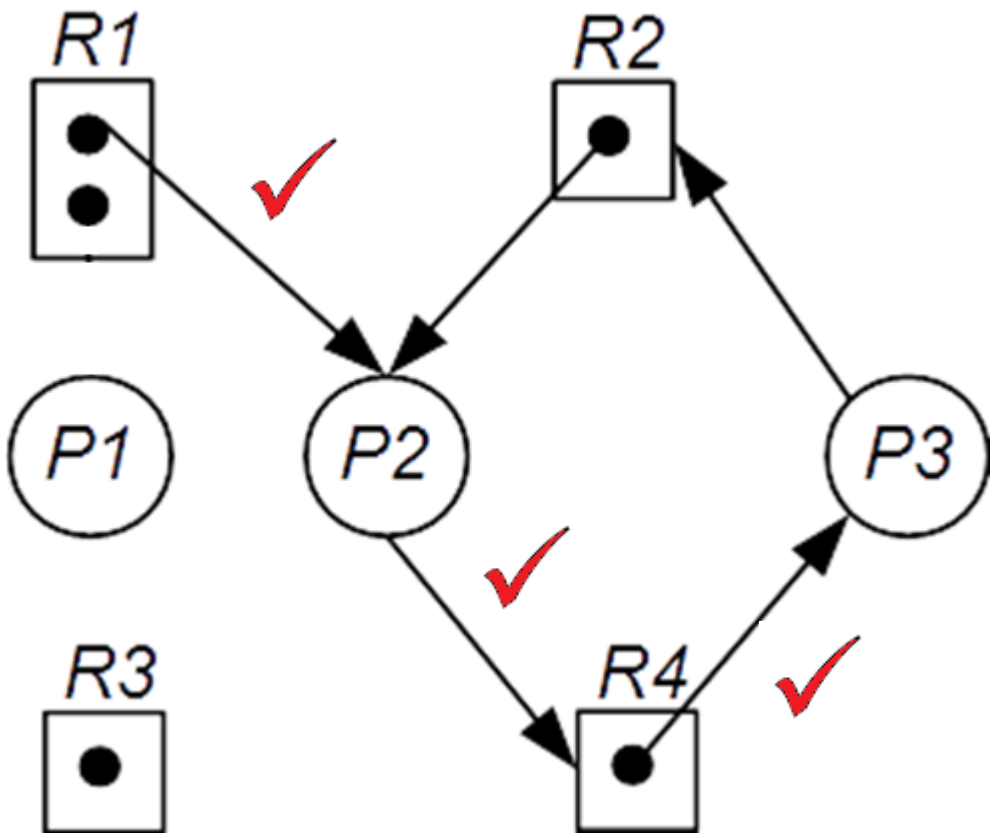
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P2,R4,P3



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

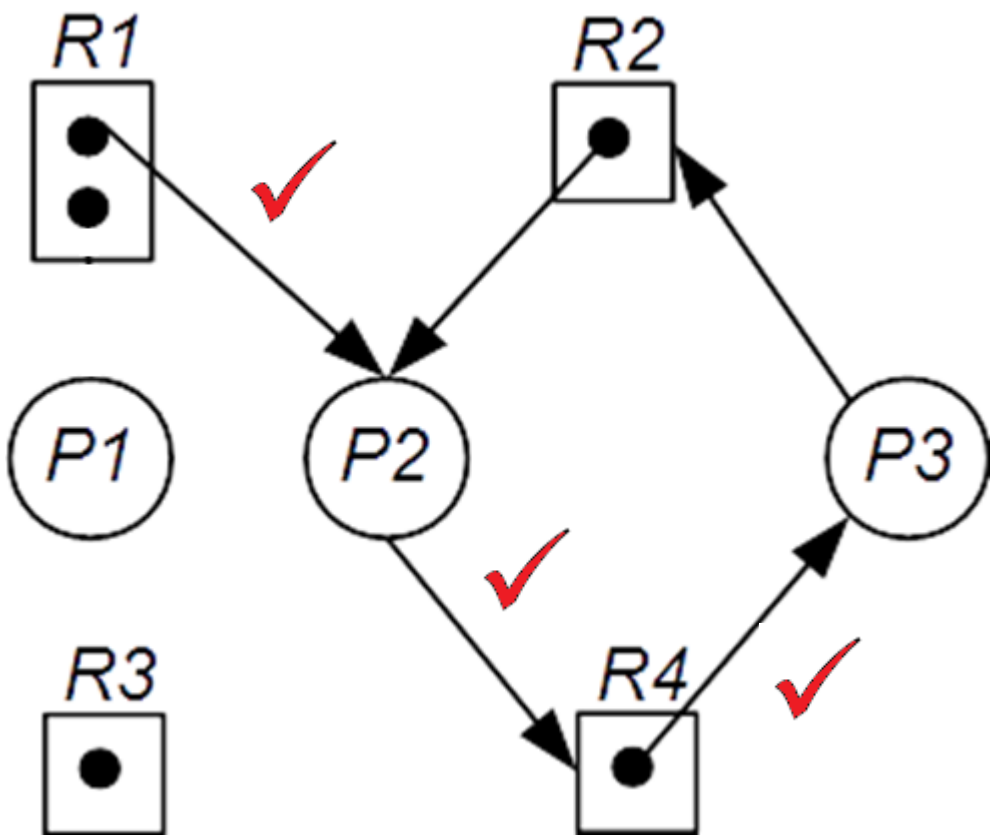
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1,P2,R4,P3



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

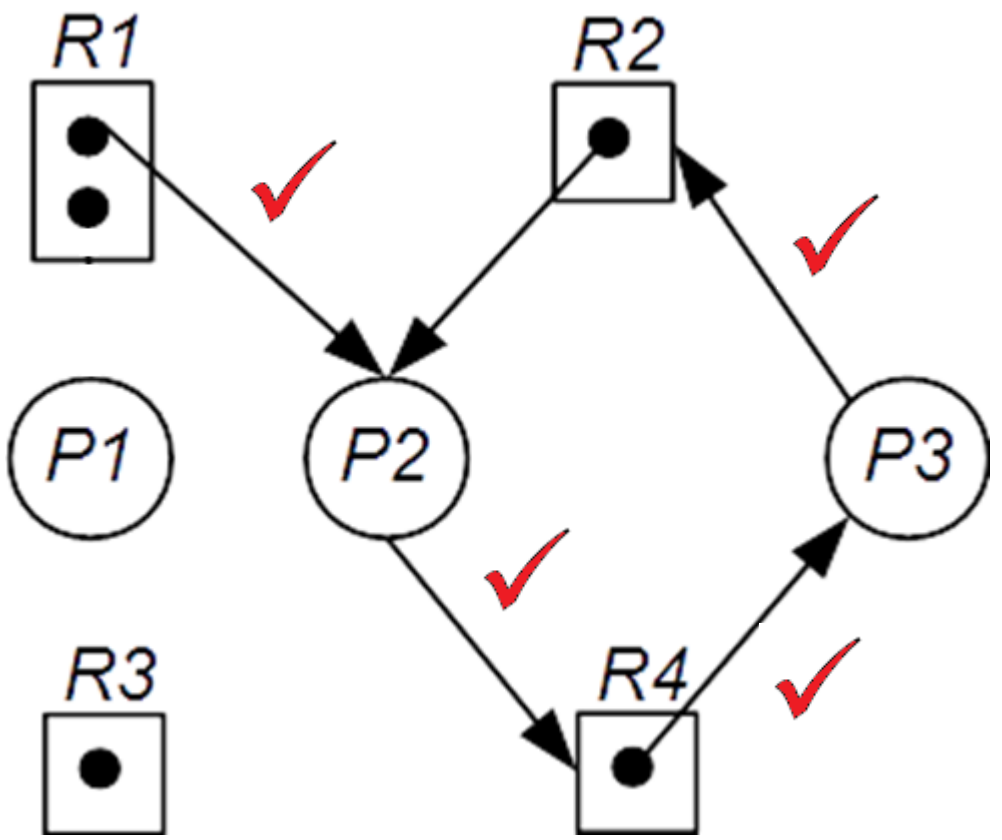
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1,P2,R4,P3



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

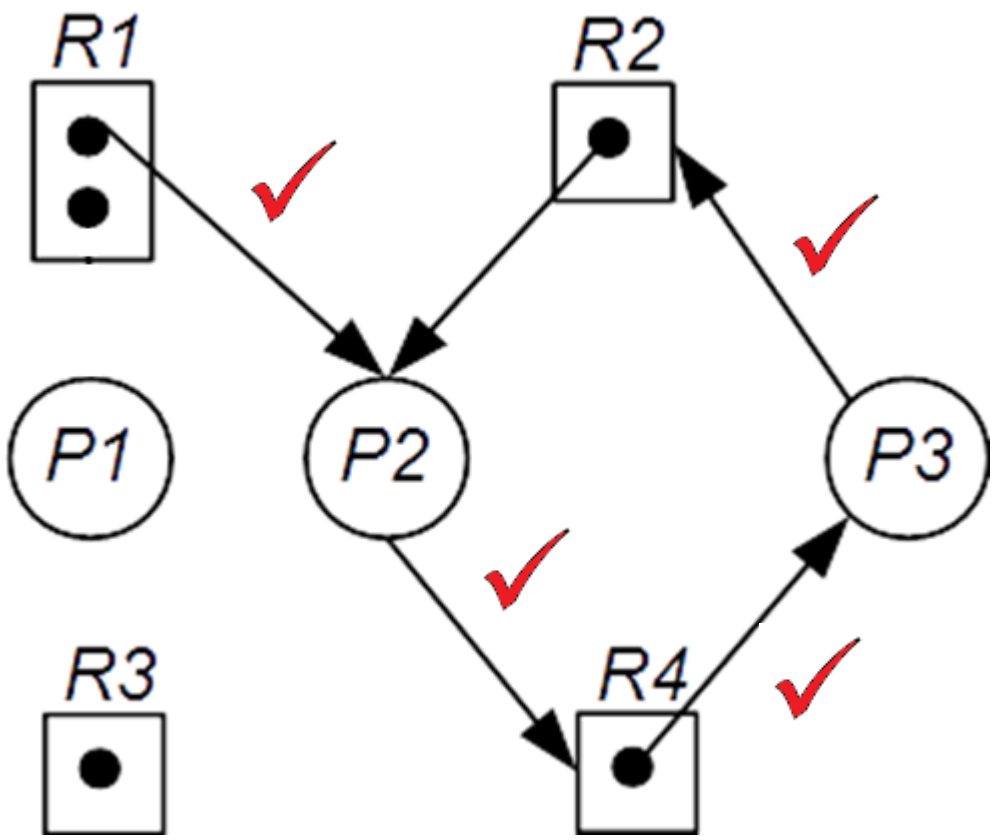
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P2,R4,P3,R2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

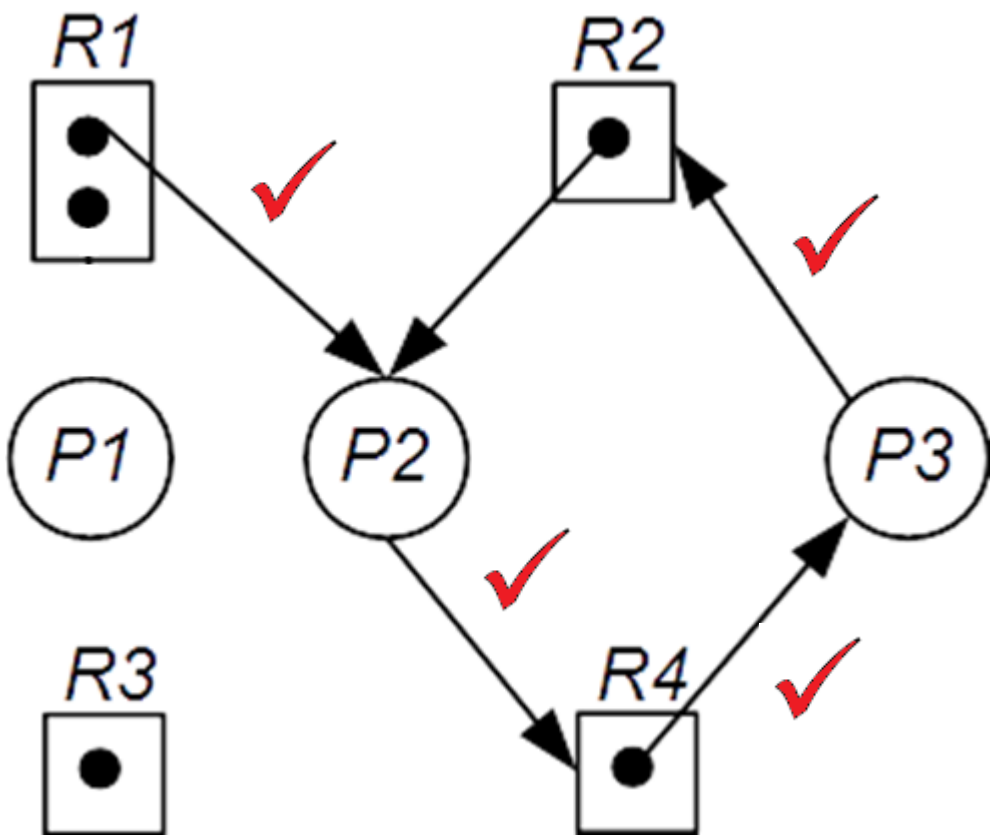
Étape 3

➢ Étape 4

Étape 5

Étape 6

L = R1,P2,R4,P3,R2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

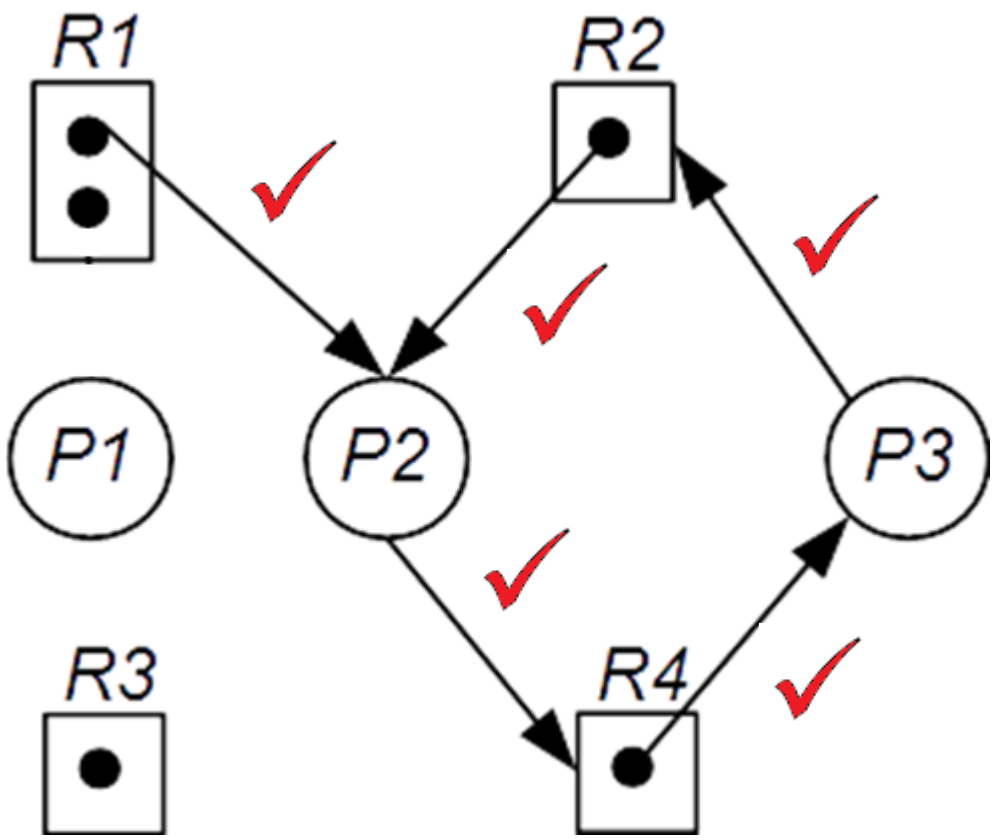
Étape 3

Étape 4

➢ Étape 5

Étape 6

L = R1,P2,R4,P3,R2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

Étape 2

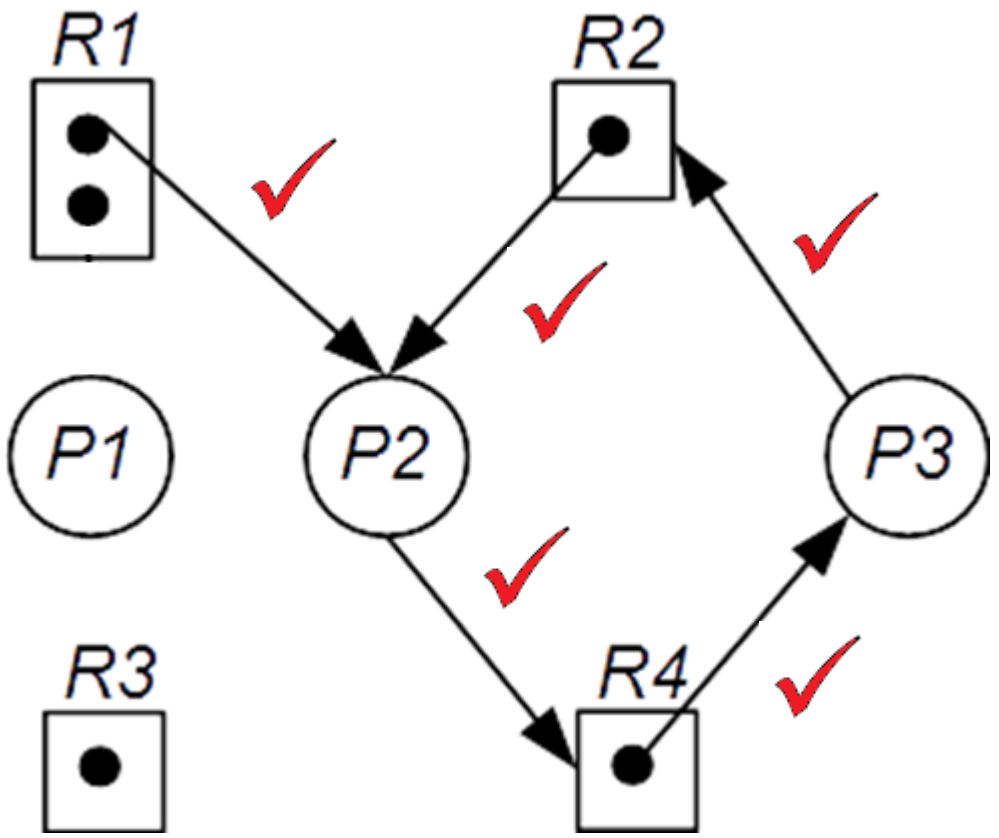
➢ Étape 3

Étape 4

Étape 5

Étape 6

L = R1,P2,R4,P3,R2,P2



# La détection et la reprise de l'interblocage (Détection/ Guérison)

Algorithme

Étape 1

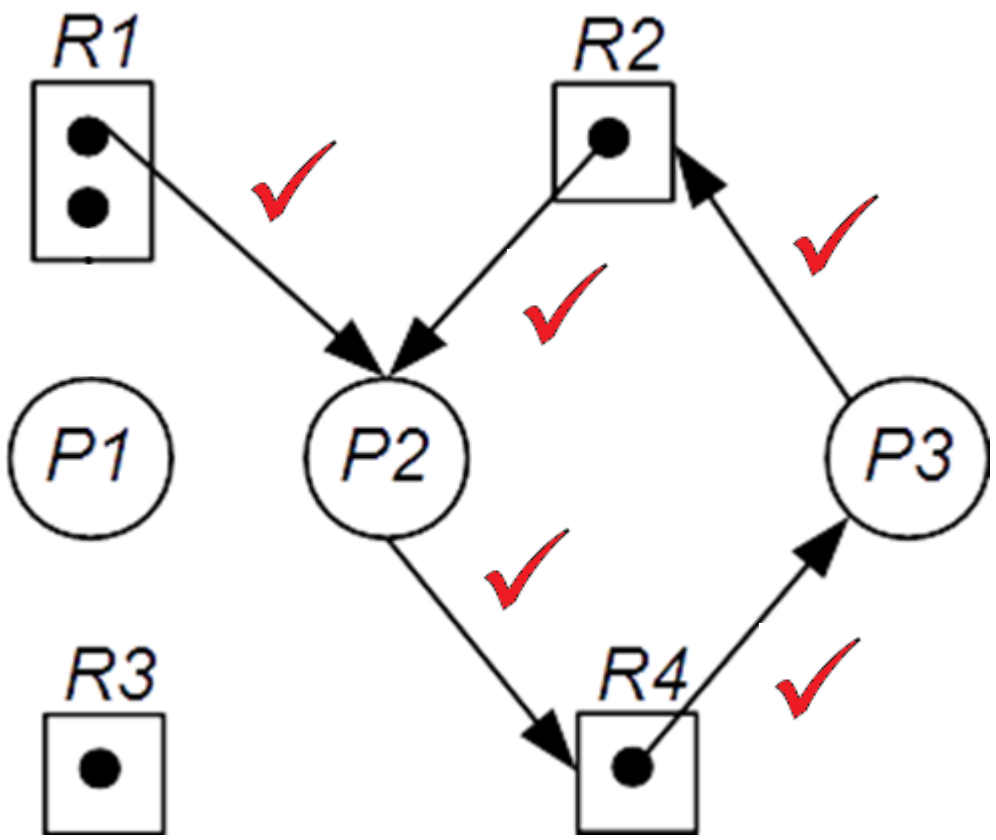
Étape 2

➢ Étape 3

Étape 4

Étape 5

Étape 6




L = R1,P2,R4,P3,R2,P2

le graphe contiens un cycle

# La détection et la reprise de l'interblocage (Détection/ Guérison)

**Cas des ressources avec plusieurs instances**

- Si on a plusieurs instances de certaines ressources un cycle dans le graphe d'allocation des ressources ne veut pas forcement dire qu'il y a un interblocages.
- Pour détecter l'interblocage dans une telle situation on procède comme suit :

# La détection et la reprise de l'interblocage (Détection/ Guérison)

Soit les données suivantes :

- **Un vecteur des ressources existantes** « V_existantes » de longueur m qui indique le nombre d'instances existantes pour chaque type de ressources,
- **Un vecteur des ressources encore disponibles** « V_disponibles » de longueur m qui indique le nombre d'instances encore disponibles pour chaque type de ressources,
- **Une matrice d'allocation des ressources** « M_alloués » de taille n x m qui indique le nombres de ressources allouées à chaque processus,
- **Une matrice de demande des ressources** « M_demandes » de taille n x m qui indique le nombres de ressources demandés par chaque processus,

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Exemple:**

- V_existantes = (2,1,1,3)

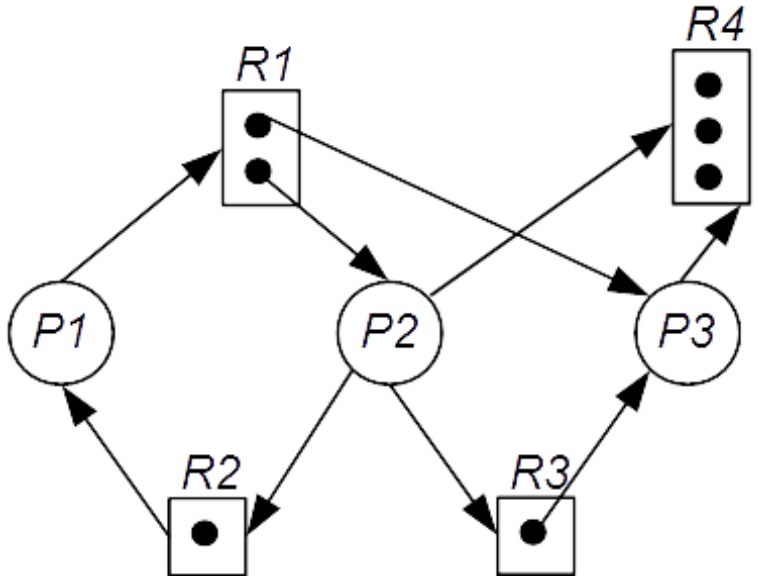
- V_disponibles = (0,0,0,3)

- M_alloués =

| | |
|---|---|
| 0 1 0 0 | P1 |
| 1 0 0 0 | P2 |
| 1 0 1 0 | P3 |

- M_demandes =

| | |
|---|---|
| 1 0 0 0 | P1 |
| 0 1 1 1 | P2 |
| 0 0 0 1 | P3 |



# La détection et la reprise de l'interblocage (Détection/ Guérison)

- Soit l'opération de comparaison « ≤ » , soit deux vecteurs A et B, A ≤ B veut dire que Ai ≤ Bi pour 1 ≤ i ≤ m
- On suppose qu'un processus conserve toutes les ressource allouées jusqu'à ce qu'il se termine.
- Un algorithme de détection de l'interblocage est comme suit *[Ref : Systèmes d'exploitation Andrew Tanenbaum p : 469]*

    1) Chercher un processus Pi non marqué pour lequel la rangée numéro i de **M_demandes** est inférieur ou égale ( **≤** ) à **V_disponibles**,
    2) Si on trouve le processus Pi alors ajouter la rangée numéro i de **M_alloués** à **V_disponibles**, marquer le processus **Pi** ; aller a l'étape 1; Sinon terminer l'algorithme.

- Lorsque l'algorithme se termine tous les processus non marqués sont en situation d'interblocage.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Exemple:**

- V_existantes = (2,1,1,3)

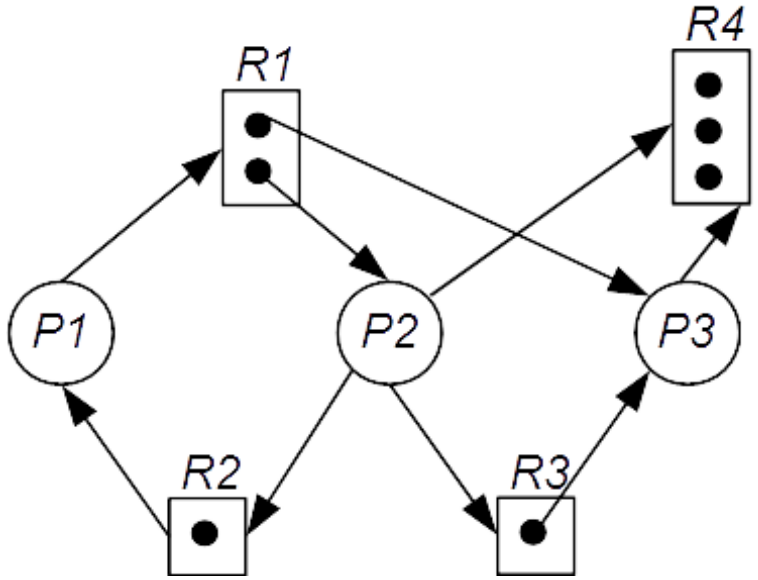
- V_disponibles = (0,0,0,3)

- M_alloués =

| | |
|---|---|
| 0 1 0 0 | P1 |
| 1 0 0 0 | P2 |
| 1 0 1 0 | P3 |

- M_demandes =

| | |
|---|---|
| 1 0 0 0 | P1 |
| 0 1 1 1 | P2 |
| 0 0 0 1 | P3 |



# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Exemple:**

- V_existantes = (2,1,1,3)

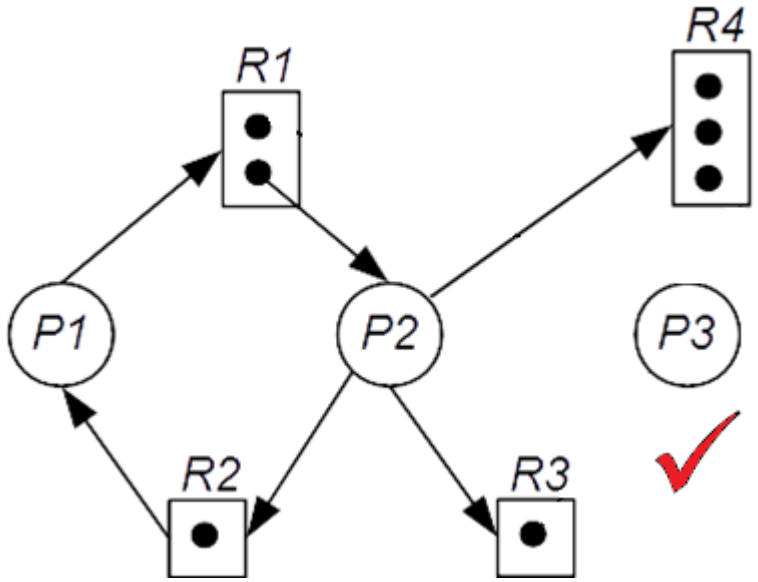
- V_disponibles = (1,0,1,3)

- M_alloués =

| | |
|---|---|
| 0 1 0 0 | P1 |
| 1 0 0 0 | P2 |
| 0 0 0 0 | P3 |

- M_demandes =

| | |
|---|---|
| 1 0 0 0 | P1 |
| 0 1 1 1 | P2 |
| 0 0 0 0 | P3 marqué |



# La détection et la reprise de l'interblocage (Détection/ Guérison)

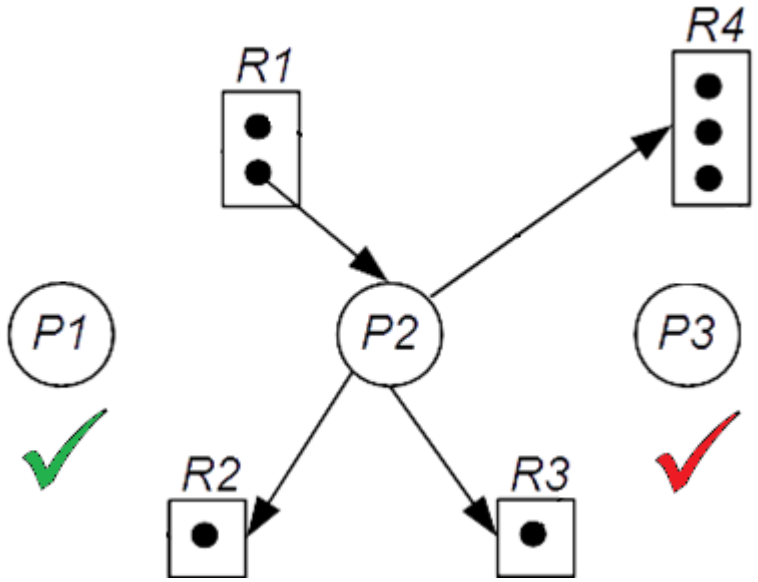
- **Exemple:**

- V_existantes = (2,1,1,3)

- V_disponibles = (1,1,1,3)

- M_alloués = 0 0 0 0 P1
  1 0 0 0 P2
  0 0 0 0 P3
  0 0 0 0 P1 marqué

- M_demandes = 0 1 1 1 P2
  0 0 0 0 P3 marqué



# La détection et la reprise de l'interblocage (Détection/ Guérison)

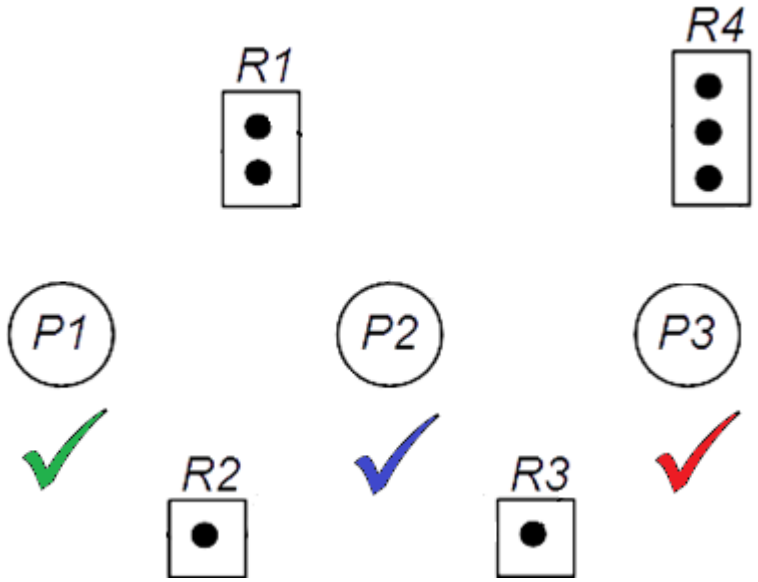
- **Exemple:**

- V_existantes = (2,1,1,3)

- V_disponibles = (2,1,1,3)

- M_alloués = 0 0 0 0 P1
  0 0 0 0 P2
  0 0 0 0 P3

- M_demandes = 0 0 0 0 P1 marqué
  0 0 0 0 P2 marqué
  0 0 0 0 P3 marqué




Tous les processus sont marqués il n' y a donc pas d'interblocage.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Exemple 2:**

- V_existantes = (1,1,1,3)

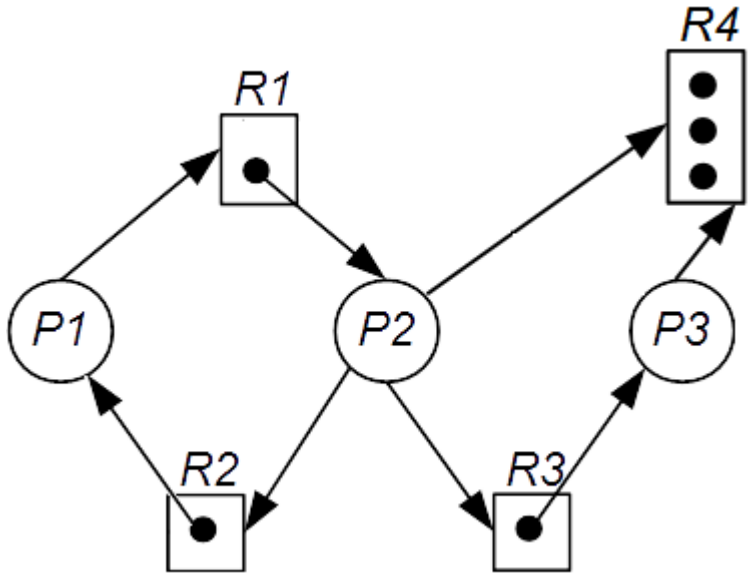
- V_disponibles = (0,0,0,3)

- M_alloués =

| | |
|---|---|
| 0 1 0 0 | P1 |
| 1 0 0 0 | P2 |
| 0 0 1 0 | P3 |

- M_demandes =

| | |
|---|---|
| 1 0 0 0 | P1 |
| 0 1 1 1 | P2 |
| 0 0 0 1 | P3 |



# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Exemple 2:**

- V_existantes = (1,1,1,3)

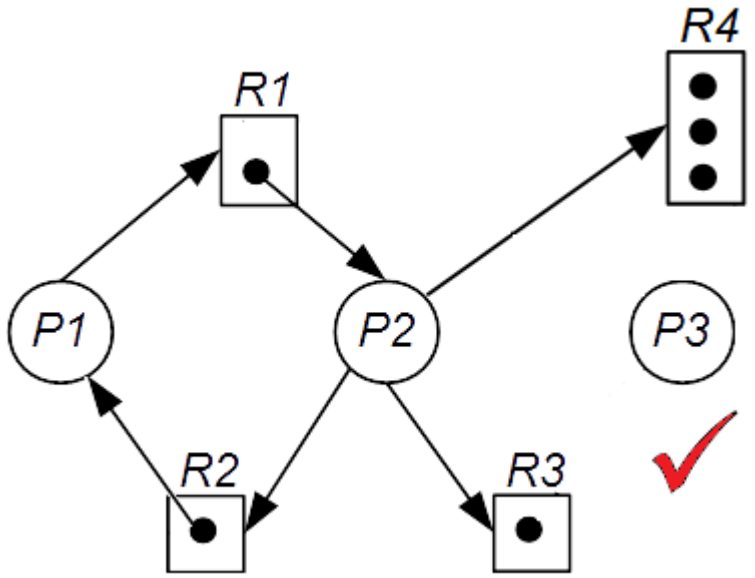
- V_disponibles = (0,0,1,3)

- M_alloués =

| | |
|---|---|
| 0 1 0 0 | P1 |
| 1 0 0 0 | P2 |
| 0 0 0 0 | P3 |

- M_demandes =

| | |
|---|---|
| 1 0 0 0 | P1 |
| 0 1 1 1 | P2 |
| 0 0 0 0 | P3 marqué |



# La détection et la reprise de l'interblocage (Détection/ Guérison)

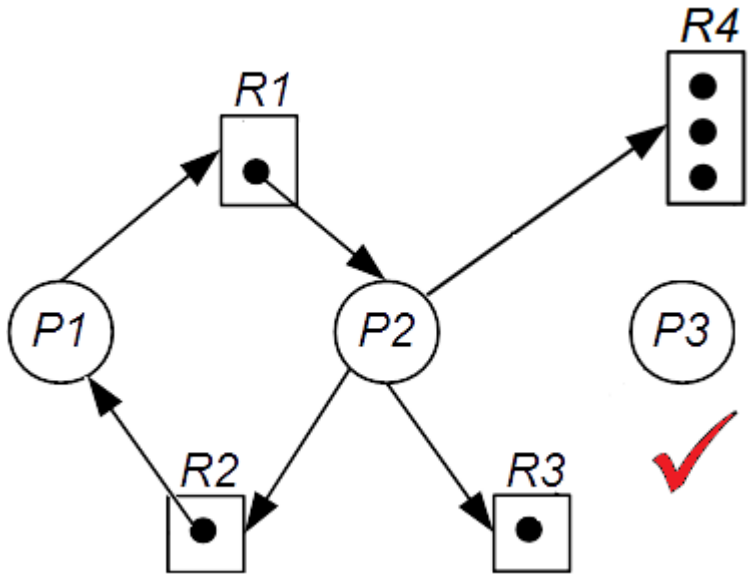
- **Exemple 2:**

- V_existantes = (1,1,1,3)

- V_disponibles = (0,0,1,3)

- M_alloués =
  0 1 0 0 P1
  1 0 0 0 P2
  0 0 0 0 P3

- M_demandes =
  1 0 0 0 P1
  0 1 1 1 P2
  0 0 0 0 P3 marqué




Les processus P1 et P2 sont dans une situation d'interblocage.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Quand exécuter l'algorithme ?**

On peut exécuter l'algorithme de détection d'interblocage à chaque fois que les ressources sont alloués, mais cette méthode n'est pas efficaces.

d'autres alternatives consistent à exécuter l'algorithme chaque T minutes , ou lorsque le taux d'utilisation du processeur est faible puisque en cas d'interblocage les processus bloqués ne s'exécute plus.

# La détection et la reprise de l'interblocage (Détection/ Guérison)

- **Comment corriger l'interblocage (Guérison) ?**

Pour éliminer un interblocage on peut terminer l'exécution de tous les processus bloqués pour libérer des ressources, ou bien les terminer un par un et vérifier à chaque fois s'il y a toujours un interblocage.

Les processus terminés devrons malheureusement reprendre leurs exécutions du début et les résultats calculés seront perdus.

# L'évitement de l'interblocage

- Dans ce cas on alloue les ressources au processus avec précaution en vérifiant à chaque fois l'état du système.
- Si un processus demande une ressource le système analyse la possibilité d’avoir un interblocage, si oui la demande du processus sera refusé sinon elle sera satisfaite.
- Chaque processus dans ce cas doit fournir des informations supplémentaires au systèmes sur les utilisations possibles des ressources. Il indique pour chaque ressource le nombre d'instances maximale qu'il peut demander.

# L'évitement de l'interblocage

**1) L'état sûr et non sûr (sain / non sain)**

- Un état sûr est un état dans lequel le système peut satisfaire toutes les demandes possibles de tous les processus en évitant l'interblocage.

➢Un état non sûr peut conduire à un interblocages et le système ne peut pas l'éviter,

➢Un état non sûr ne veut pas dire qu'il y a un interblocages,

➢Si l'état du système est sûr alors le système peut éviter l'interblocage.

# L'évitement de l'interblocage

**1) L'état sûr et non sûr (sain / non sain)**

- Exemple:

Soit les trois processus P1, P2 P3 et une ressource R. Le nombre d'instances disponibles de R est 10.Les demandes maximales des processus sont successivement 8, 7, et 5. Soit les deux états du système suivants :

L'état **(1)** est sûr puisque le système peut trouver une séquence sûre pour éviter l'interblocage.

Par contre l'état **(2)** n'est pas sûr, il peut se trouver dans un état d'interblocage et on ne peut pas l'éviter.

| | *alloués* | *Max* |
|---|---|---|
| *P1* | *4* | *8* |
| *P2* | *1* | *7* |
| *P3* | *2* | *5* |

**1) *V_Disponibles*** = 3

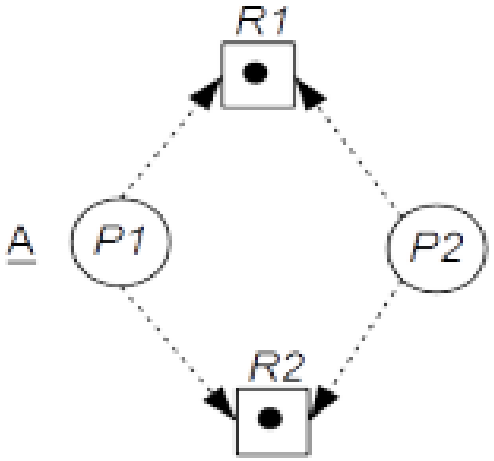
| | *alloués* | *Max* |
|---|---|---|
| *P1* | *4* | *8* |
| *P2* | *3* | *7* |
| *P3* | *1* | *5* |

**2) *V_Disponibles*** = 2

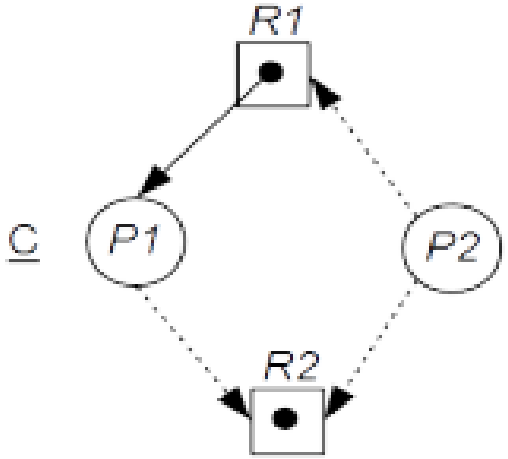
# L'évitement de l'interblocage

2) **L'algorithme du graphe d'allocation des ressources (**ressource à une instance**)**

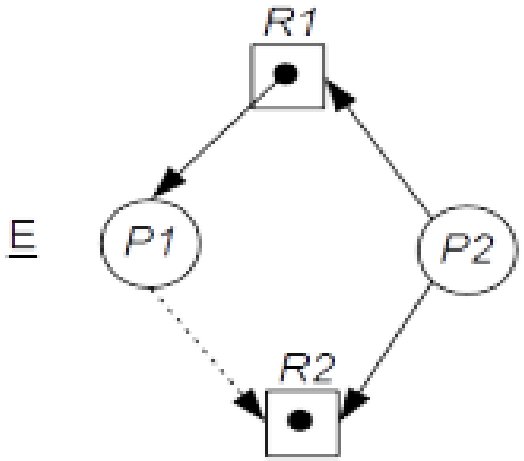
- Il utilise un graphe d'allocation des ressources et ajoute un autre type d'arc appelé **arc de réclamation** de ressource qui veut dire qu'un processus peut dans le future demander une ressource.
- Au début, pour chaque ressource **R** qui peut être demandé par un processus **P** on ajoute un arc de réclamation entre **P** et **R**.
- Si un Processus demande une ressource **R** elle ne peut lui être attribué que si elle a était déjà réclamé et qu'elle ne conduit pas à la formation d'un cycle dans le graphe, dans ce cas l'**arc de réclamation** est transformé en **arc d'allocation**. Sinon la demande sera **refusé**.
- Si un processus libère une ressource l'arc d'allocation entre le processus et la ressource est transformé en arc de réclamation.



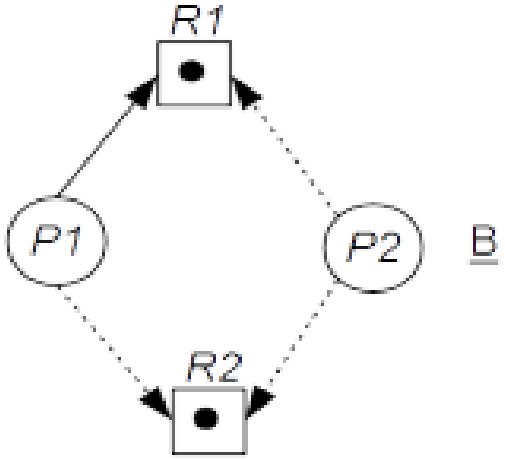

P1 et P2 réclament les ressources R1 et R2



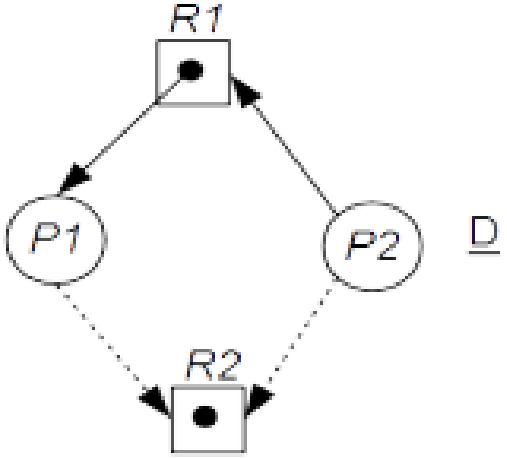

R1 est alloué à P1



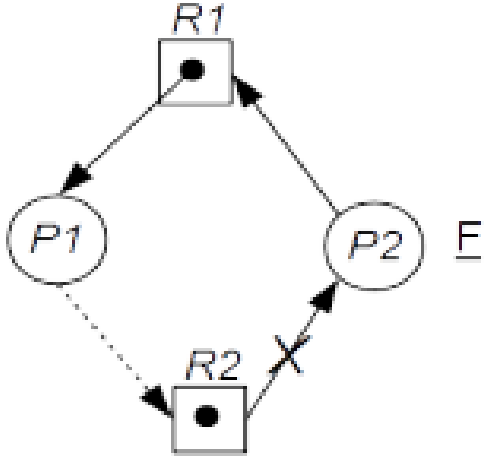

P2 demande R2




P1 demande R1




P2 demande R1 mais elles n'est pas disponible




R2 n'est pas alloué à P2 malgré qu'elle est disponible puisque sa vas créer un cycle dans le graphe

**L'algorithme du graphe d'allocation des ressources (**ressource à une instance**)**

# L'évitement de l'interblocage

3) **L'algorithme du banquier:** Peut être utilisé si on a plusieurs instances des ressources.

- Chaque processus indique le nombre d'instances maximal qu'il peut demander,

- Une ressource ne peut être alloués que si l'état du système reste sûr,

- Si une allocation de ressource met le système dans un état non sûr le processus qui a demandé la ressource est mis en attente.

L'algorithme du banquier permet de garantir que l'interblocage ne se produise pas, pour cela il alloue de nouvelle ressource à un processus seulement si l'état du système reste sûr. Pour vérifier si l'état du système est sûr ou pas on utilise la même méthode de détection de l'interblocage (cas de plusieurs ressource) vu précédemment et cela en supposant que chaque processus a besoin du maximum de ressources pour s'exécuter

# L'évitement de l'interblocage

**3) L'algorithme du banquier:**

**Exemple:**

Soit les trois processus P1, P2 P3 et P4 et quatre ressources R1,R2 et R3. Le nombre d'instances existantes sont successivement 6,6,4,4. Les demandes maximales des processus sont décrit par la matrice ***M_maximum*** comme suit:

L'état d'allocation des ressources est décrit par la matrice ***M_alloués*** comme suit :

***V_disponibles*** = ( 2, 1 , 2 , 3 )
L'état actuelle du système est sûr puisque **on peut éviter** la situation d'interblocage même si les processus demandent leurs ressources maximales.

Si le processus P3 demande 2 ressource supplémentaire de R4 elles lui seront accordées puisque l'état reste toujours sûr, par contre si le processus P2 demande 2 ressources de R3 le système vas le mettre en attente parce que il vas générer un état non sûr.

***M_maximum***

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 4 | 5 | 2 | 2 |
| P2 | 3 | 3 | 4 | 1 |
| P3 | 3 | 2 | 3 | 2 |

***M_alloués***

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| P1 | 2 | 2 | 0 | 0 |
| P2 | 1 | 2 | 1 | 1 |
| P3 | 1 | 1 | 1 | 0 |

# L'évitement de l'interblocage

**3) L'algorithme du banquier:**

- L'algorithme du banquier bien qu'il semble bien fait, il a certain inconvénients :
  - ***Coûteux*** : Il est très coûteux puisque il doit vérifier à chaque fois si l'état du système reste sain ou pas ,
  - ***Théorique*** : En pratique on ne peut pas obliger les processus à déclarer à l'avance les ressources maximales qu'ils peuvent utiliser,
  - ***Pessimiste*** : L'algorithme peut retarder une demande de ressources dès qu'il y a risque d'interblocage, mais l'interblocage peut éventuellement ne pas se produire.

# Chapitre 3 : L'interblocage

3.1 Introduction
3.2 Les ressources
3.4 Graphe d'allocation des ressources
3.5 Méthodes pour traiter l'interblocage
3.5.1 Politique de l'autruche
3.5.2 Prévention de l'interblocage
3.5.3 La détection et la reprise de l'interblocage (Détection/ Guérison)
3.5.4 L'évitement de l'interblocage
3.5.4.1 L'état sûr et non sûr (sain / non sain)
3.5.4.2 L'algorithme du graphe d'allocation des ressources
3.5.4.3 L'algorithme du banquier