Chapitre 7

Interblocages

7.1 Les processus et les ressources

L'exécution d'un processus nécessite un ensemble de ressources (mémoire principale, disques, fichiers, périphériques, etc.) qui lui sont attribuées par le système d'exploitation. L'utilisation d'une ressource passe par les étapes suivantes :

- **Demande de la ressource :** Si l'on ne peut pas satisfaire la demande il faut attendre. La demande sera mise dans une table d'attente des ressources.
- **Utilisation de la ressource :** Le processus peut utiliser la ressource.
- **Libération de la ressource :** Le processus libère la ressource demandée et allouée.

Lorsqu'un processus demande un accès exclusif à une ressource déjà allouée à un autre processus, le système d'exploitation décide de le mettre en attente jusqu'à ce que la ressource demandée devienne disponible ou lui retourner un message indiquant que la ressource n'est pas disponible : réessayer plus tard.

Les ressources peuvent être de plusieurs types :

- **Reutilisables ou disponibles** Existent-elles après son utilisation ?
  - Reutilisables :
    - Toutes les ressources physiques.
    - Quelques unes logiques (fichiers, mutex, verrous, etc.).
  - Disponibles :
    - Un processus engendre une ressource et un autre l'utilise.
    - Ressources associées à la communication et à la synchronisation comme les messages, les signaux, les sémaphores, etc.

- **D'usage partagé :** Est-ce que la ressource peut être utilisée par plusieurs processus en même temps ? Les ressources partagées n'affectent pas les interblocages.
- **Avec un ou multiples exemplaires :** Existent-ils de multiples exemplaires d'une même ressource ?
- **Preémptible ou non preémptible :** Est-ce qu'on a le droit de retirer une ressource quand elle est utilisée par un processus ?

La différence principale entre ressources préemptibles et non préemptibles est que les premières peuvent être retirées sans risque au processus qui les détient, tandis que les deuxièmes ne peuvent être retirées sans provoquer des problèmes. Comme exemples de ressources preémptibles nous avons le processeur (où le changement de contexte correspond à l'expropriation et l'état du processeur est copié à la **Table de Contrôle de Processus** (**BCP**)) et la mémoire virtuelle (le remplacement est une expropriation et le contenu de la page doit être copié au *swap*). Les imprimantes et les scanners sont des exemples de ressources non preémptibles. Pour étudier le problème des interblocages, nous allons considérer uniquement les ressources non préemptibles.

7.2 Définition d'un interblocage

Des problèmes peuvent survenir, lorsque les processus obtiennent des accès exclusifs aux ressources. Par exemple, un processus P_1 détient une ressource R_1 et attend une autre ressource R_2 qui est utilisée par un autre processus P_2 ; le processus P_2 détient la ressource R_2 et attend la ressource R_1. On a une situation d'**interblocage** (*deadlock* en anglais) car P_1 attend P_2 et P_2 attend P_1. Les deux processus vont attendre indéfiniment comme montré sur la figure 7.1.

En général, un ensemble de processus est en interblocage si chaque processus attend la libération d'une ressource qui est allouée à un autre processus de l'ensemble. Comme tous les processus sont en attente, aucun ne pourra s'exécuter et donc libérer les ressources demandées par les autres. Ils attendront tous indéfiniment.

▶ **Exemple 1.** Interblocages.

Accès aux périphériques. Supposons que deux processus **A** et **B** veulent imprimer, en utilisant la même imprimante, un fichier stocké sur une bande magnétique. La taille de ce fichier est supérieure à la capacité du disque. Chaque processus a besoin d'un accès exclusif au dérou-

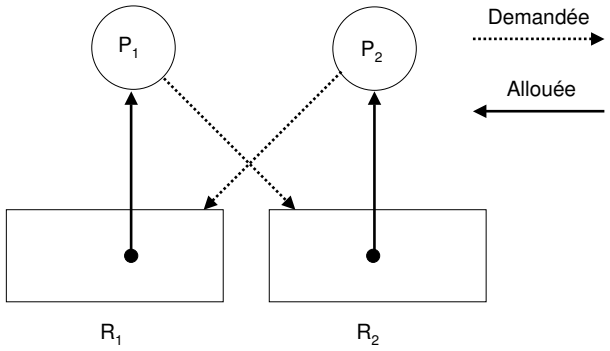

FIG. 7.1 – Situation d'interblocage de deux processus.

leur et à l'imprimante simultanément. On a une situation d'interblocage si :
- Le processus **A** utilise l'imprimante et demande l'accès au dérouleur.
- Le processus **B** détient le dérouleur de bande et demande l'imprimante.

Accès à une base de données. Supposons deux processus **A** et **B** qui demandent des accès exclusifs aux enregistrements d'une base de données. On arrive à une situation d'interblocage si :
- Le processus **A** a verrouillé l'enregistrement R_1 et demande l'accès à l'enregistrement R_2.
- Le processus **B** a verrouillé l'enregistrement R_2 et demande l'accès à l'enregistrement R_1.

Circulation routière. Considérons deux routes à double sens qui se croisent comme dans la figure 7.2, où la circulation est impossible. Un problème d'interblocage y est présent.

La résolution des interblocages constitue un point théorique très étudié. Mais, pour l'instant, il n'y a pas de solution complètement satisfaisante et la plupart des systèmes —notamment Unix— ne cherchent pas à traiter ce phénomène. Cependant, si on ne peut pas résoudre le problème dans sa généralité, on devra tenir compte de certaines techniques qui minimisent les risques. D'un autre côté, certaines tendances font de l'étude des interblocages un domaine très important : plus la technologie progresse, plus il y a de ressources, et un nombre grandissant de processus cohabitent dans un système.

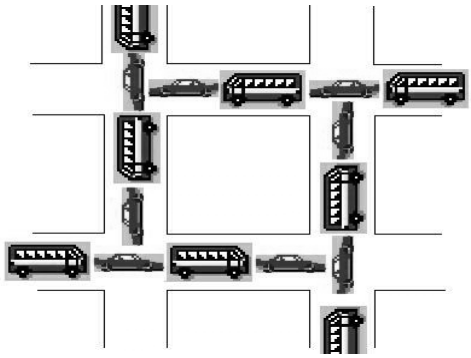

FIG. 7.2 – Problème de circulation routière.

7.2.1 Conditions nécessaires pour l'interblocage

Pour qu'une situation d'interblocage ait lieu, les quatre conditions suivantes doivent être remplies (Conditions de Coffman) :

- **L'exclusion mutuelle.** A un instant précis, une ressource est allouée à un seul processus.
- **La détention et l'attente.** Les processus qui détiennent des ressources peuvent en demander d'autres.
- **Pas de préemption.** Les ressources allouées à un processus sont libérées uniquement par le processus.
- **L'attente circulaire.** Il existe une chaîne de deux ou plus processus de telle maniére que chaque processus dans la chaîne requiert une ressource allouée au processus suivant dans la chaîne.

Par exemple, dans le problème de circulation de la figure 7.2 le trafic est impossible. On observe que les quatre conditions d'interblocage sont bien remplies :

- **Exclusion mutuelle** : Seulement une voiture occupe un endroit particulier de la route à un instant donné.
- **Détention et attente** : Aucun voiture ne peut faire marche arrière.
- **Pas de préemption** : On ne permet pas à une voiture de pousser une autre voiture en dehors de la route.
- **Attente circulaire** : Chaque coin de la rue contient des voitures dont le mouvement depend des voitures qui bloquent la prochaine intersection.

7.3 Graphe d'allocation des ressources

Le **graphe d'allocation des ressources** est un graphe biparti composé de deux types de nœuds et d'un ensemble d'arcs :

- Les **processus** qui sont représentés par des cercles.
- Les **ressources** qui sont représentées par des rectangles. Chaque rectangle contient autant de points qu'il y a d'exemplaires de la ressource représentée.
- Un arc orienté d'une ressource vers un processus signifie que la ressource est allouée au processus.
- Un arc orienté d'un processus vers une ressource signifie que le processus est bloqué en attente de la ressource.

Ce graphe indique pour chaque processus les ressources qu'il détient ainsi que celles qu'il demande.

▶ **Exemple 2.** Soient trois processus **A**, **B** et **C** qui utilisent trois ressources **R**, **S** et **T** comme illustré à la figure 7.3 :

A	B	C
Demande R	Demande S	Demande T
Demande S	Demande T	Demande R
Libère R	Libère S	Libère T
Libère S	Libère T	Libère R

FIG. 7.3 – Besoins de trois processus.

Si les processus sont exécutés de façon séquentielle : **A** suivi de **B** suivi **C**, il n'y pas d'interblocage. Supposons maintenant que l'exécution des processus est gérée par un ordonnanceur du type circulaire. Si les instructions sont exécutées dans l'ordre :

1. **A** demande **R**
2. **B** demande **S**
3. **C** demande **T**
4. **A** demande **S**
5. **B** demande **T**
6. **C** demande **R**

on atteint une situation d'interblocage, fait qui est montré sur la figure 7.4.

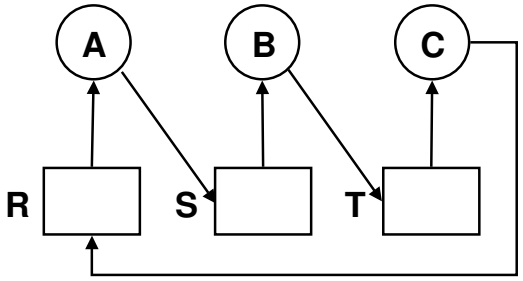

FIG. 7.4 – Situation d'interblocage de trois processus.

7.3.1 Réduction du graphe d'allocation des ressources

Un graphe réduit peut-être utilisé pour déterminer s'il existe ou non un interblocage. Pour la **réduction d'un graphe d'allocation des ressources**, les flèches associées à chaque processus et à chaque ressource doivent être vérifiées.

- Si une ressource possède seulement des flèches qui sortent (il n'y a pas des requêtes), on les efface.
- Si un processus possède seulement des flèches qui pointent vers lui, on les efface.
- Si une ressource a des flèches qui sortent, mais pour chaque flèche de requête il y a une ressource disponible dans le bloc de ressources où la flèche pointe, il faut les effacer.

▶ **Exemple 3.** (Pris de [**?**]) Considérez quatre processus $P_1, \cdots, P_4$ qui utilisent des ressources du type R_1, R_2 et R_3. Le tableau 7.1 montre l'allocation courante et le nombre maximaux d'unités de ressources nécessaires pour l'exécution des processus. Le nombre de ressources disponibles est $A = [0, 0, 0]$. Le graphe d'allocation des ressources pour l'état courant est montré sur la figure 7.5.

Processus	R_1	R_2	R_3	R_1	R_2	R_3
P_1	3	0	0	0	0	0
P_2	1	1	0	1	0	0
P_3	0	2	0	1	0	1
P_4	1	0	1	0	2	0

TAB. 7.1 – Besoins de quatre processus.

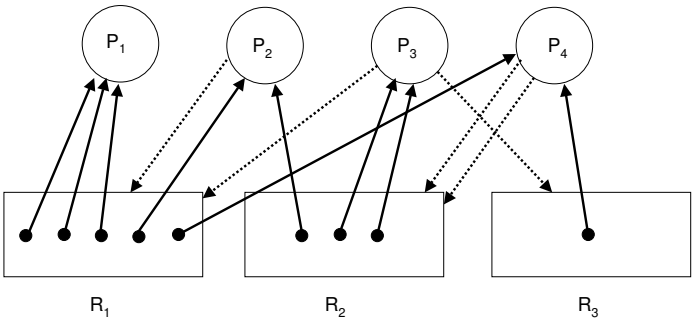

FIG. 7.5 – Graphe d'allocation des ressources.

P_1 possède seulement des flèches qui pointent vers lui, alors on les efface. Les requêtes de R_1 effectuées par P_2 peuvent donc être allouées, et P_2 aura seulement des flèches qui pointent vers lui. On les efface aussi. On efface la flèche de R_1 vers P_3 (qui a changé de direction) car R_1 n'a que des flèches sortant. Le graphe réduit final est montré sur la figure 7.6.

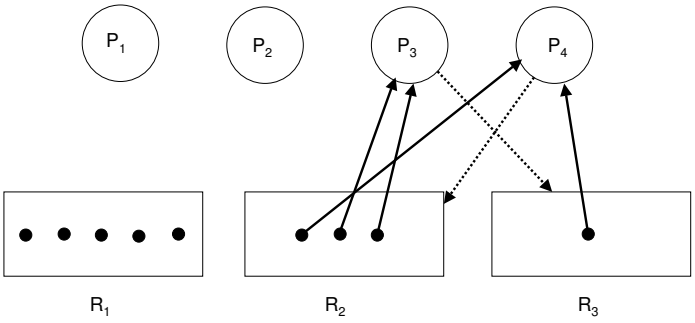

FIG. 7.6 – Graphe réduit d'allocation.

7.4 Traitement des interblocages

Comme nous l'avons mentionné, les situations d'interblocage peuvent se produire dans un système. La question qui se pose est donc : doit-il prendre en compte ce problème ou l'ignorer ?

Ignorer complètement les problèmes. On peut faire l'autruche et ignorer les possibilités d'interblocages. Cette « stratégie » est celle de la plu-

part des systèmes d'exploitation courants car le prix à payer pour les éviter est élevé.

Les détecter et y remédier. On tente de traiter les interblocages, en détectant les processus interbloqués et en les éliminant.

Les éviter. En allouant dynamiquement les ressources avec précaution. Le système d'exploitation peut suspendre le processus qui demande une allocation de ressource s'il constate que cette allocation peut conduire à un interblocage. Il lui attribuera la ressource lorsqu'il n'y aura plus de risque.

Les prévenir. En empêchant l'apparition de l'une des quatre conditions de leur existence.

7.5 La détection et la reprise

Dans ce cas, le système ne cherche pas à empêcher les interblocages. Il tente de les détecter et d'y remédier. Pour détecter les interblocages, il construit dynamiquement le graphe d'allocation des ressources du système qui indique les attributions et les demandes de ressources. Dans le cas des ressources à exemplaire unique, il existe un interblocage si le graphe contient au moins un cycle. Dans le cas des ressources à exemplaires multiples, il existe un interblocage si le graphe contient au moins un cycle terminal (aucun arc ne permet de le quitter).

Le système vérifie s'il y a des interblocages :

- A chaque modification du graphe suite à une demande d'une ressource (coûteuse en termes de temps processeur).
- Périodiquement ou lorsque l'utilisation du processeur est inférieure à un certain seuil (la détection peut être tardive).

7.5.1 Algorithme de détection des interblocages

Soient n le nombre de processus $P[1], P[2], \ldots, P[n]$ et m le nombre de types de ressources $E[1], E[2], \ldots, E[m]$ qui existent dans un système. L'**algorithme de détection des interblocages** utilise les matrices et vecteurs suivants :

- Matrice C des allocations courantes d'ordre (n x m). L'élément $C[i, j]$ désigne le nombre de ressources de type $E[j]$ détenues par le processus $P[i]$.

- Matrice R des demandes de ressources d'ordre (n x m). L'élément $R[i, j]$ est le nombre de ressources de type $E[j]$ qui manquent au processus $P[i]$ pour pouvoir poursuivre son exécution.
- Vecteur A d'ordre m. L'élément $A[j]$ est le nombre de ressources de type $E[j]$ disponibles (non attribuées).
- Vecteur E d'ordre m. L'élément $E[j]$ est le nombre total de ressources de type j existantes dans le système.

1. Rechercher un processus $P[i]$ non marqué dont la rangée $R[i]$ est inférieure à A
2. Si ce processus existe, ajouter la rangée $C[i]$ à A, marquer le processus et revenir à l'étape 1
3. Si ce processus n'existe pas, les processus non marqués sont en interblocage. L'algorithme se termine.

On peut démontrer mathématiquement que s'il existe au moins une séquence sûre, alors il existe un nombre infini de séquences sûres. Dans les états sûrs, le système d'exploitation possède le contrôle sur les processus. Dans un état non sûr le contrôle dépends du comportement des processus.

▶ **Exemple 4.** (Pris de [**?**]). Considérons un système où nous avons trois processus en cours et qui dispose de 4 types de ressources : 4 dérouleurs de bandes, 2 scanners, 3 imprimantes et un lecteur de CD ROM. Les ressources détenues et demandées par les processus sont indiquées par les matrices C et R.

$E = [4, 2, 3, 1]$; $A = [2, 1, 0, 0]$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}; R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Seul le processus P3 peut obtenir les ressources dont il a besoin. Il s'exécute jusqu'à la fin puis libère ses ressources, ce qui donne : $A = [2, 2, 2, 0]$. Ensuite P2 peut obtenir les ressources dont il a besoin. Il s'exécute jusqu'à la fin et libère ses ressources, ce qui donne : $A = [4, 2, 2, 1]$. Enfin, P1 peut obtenir les ressources dont il a besoin. Il n'y a pas d'interblocage.

▶ **Exemple 5.** Considérons un système ayant sept processus, A à G, et six ressources R à W. L'attribution des ressources est la suivante :

- A détient R et demande S ;
- B demande T ;

– C demande S ;
– D détient U et demande S et T ;
– E détient T et demande V ;
– F détient W et demande S ;
– G détient V et demande U.

Construire le graphe d'allocation des ressources ? Y a-t-il un interblocage ? Si oui, quels sont les processus concernés ?

▶ **Exemple 6.**

Considérons un système ayant quatre processus, A, B. C et D, et trois types ressources R, S et T, à plusieurs exemplaires, avec 3R, 2S et 2T. L'attribution des ressources est la suivante :

– A détient une ressource de type R et demande une ressource de type S
– B détient 2 ressources de type S et demande une ressource de type R et une ressource de type T
– C détient 1 ressource de type R et demande une ressource de type S
– D détient 2 ressources de type T et demande une ressource de type R

Construire le graphe d'allocation des ressources. Y a-t-il un interblocage ? Si oui, quels sont les processus concernés ?

7.5.2 La reprise des interblocages

Lorsque le système détecte un interblocage, il doit le supprimer, ce qui se traduit généralement par la réalisation de l'une des opérations suivantes :

– Retirer temporairement une ressource à un processus pour l'attribuer à un autre.
– Restaurer un état antérieur (retour en arrière) et éviter de retomber dans la même situation.
– Supprimer un ou plusieurs processus.

7.6 L'évitement des interblocages

Dans ce cas, lorsqu'un processus demande une ressource, le système doit déterminer si l'attribution de la ressource est sûre. Si c'est le cas, il lui attribue la ressource. Sinon, la ressource n'est pas accordée. Un état est sûr si tous les processus peuvent terminer leur exécution (il existe une séquence d'allocations de ressources qui permet à tous les processus de se terminer).

Un état est non sûr si on ne peut garantir que les processus pourraient terminer leurs exécutions.

Mais comment déterminer si un état est sûr ou non sûr ? Djikstra a proposé en 1965 un algorithme d'ordonnancement, appelé l'**Algorithme du banquier** qui permet de répondre à cette question. Cet algorithme, utilise les mêmes informations que celles de l'algorithme de détection précédent : matrices A,E,C et R. Un état est caractérisé par ces quatre tableaux.

7.6.1 Algorithme du banquier

Cet algorithme (Dijistra, 1965[1], Habermann 1969 [2]) consiste à examiner chaque nouvelle requête pour voir si elle conduit à un état sûr. Si c'est le cas, la ressource est allouée, sinon la requête est mise en attente. L'algorithme détermine donc si un état est ou non sûr :

1. Trouver un processus P_i non marqué dont la rangée i de R est inférieure à A.
2. Si un tel processus n'existe pas alors l'état est non sûr (il y a interblocage). L'algorithme se termine.
3. Sinon, ajouter la rangée i de C à A, et marquer le processus.
4. Si tous les processus sont marqués alors l'état est sûr et l'algorithme se termine, sinon aller à l'étape 1.

L'algorithme du banquier permet bien d'éviter les interblocages mais il est peu utilisé en pratique car on ne connaît pas toujours à l'avance les besoins en ressources des processus.

▶ **Exemple 7.** On considère quatre processus $P_1, \cdots, P_4$ qui utilisent des ressources du type R_1, R_2 et R_3. Le tableau 7.2 montre l'allocation courante et le nombre maximaux d'unités de ressources nécessaires pour l'exécution des processus. Le nombre de ressources disponibles est $A = [0, 0, 0]$

1. a) Construire le graphe d'allocation des ressources pour l'état courant.
2. b) L'état courant est-il sûr ?
3. c) Correspond-il à un interblocage ?

---

[1] Dijkstra, E.W., 1965. Solution of a Problem in Concurrent Programming Control. Communications of the ACM, 8(9) :569, 1965.

[2] Habermann, A.N. Prevention of System Deadlocks. CACM, Vol. 12, No. 7, July 1969, pp. 373-385.

Processus	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	0	0	1	1	0
P_2	3	1	0	0	0	0
P_3	1	3	0	0	0	1
P_4	0	1	1	0	1	0

TAB. 7.2 – Besoins de quatre processus.

7.7 La prévention des interblocages

Pour prévenir les interblocages, on doit éliminer une des quatre conditions nécessaires à leur apparition.

L'exclusion mutuelle. Pour éviter l'exclusion mutuelle, il est parfois possible de sérialiser les requêtes portant sur une ressource. Par exemple, pour les imprimantes, les processus « spoolent » leurs travaux dans un répertoire spécialisé et un démon d'impression les traitera, en série, l'un après l'autre.

La Détention et l'attente. Pour ce qui concerne la deuxième condition, elle pourrait être évitée si les processus demandaient leurs ressources à l'avance. Ceci est en fait très difficile à réaliser dans la pratique car l'allocation est, en général, dynamique. Empêcher cette condition serait donc particulièrement coûteux.

Pas de préemption. La troisième condition n'est pas raisonnablement traitable pour la plupart des ressources sans dégrader profondément le fonctionnement du système. On peut cependant l'envisager pour certaines ressources dont le contexte peut être sauvegardé et restauré.

L'attente circulaire. Enfin, on peut résoudre le problème de l'attente circulaire en numérotant les ressources et en n'autorisant leur demande, par un processus, que lorsqu'elles correspondent à des numéros croissants ou en accordant aux processus une seule ressource à la fois (s'il a besoin d'une autre ressource, il doit libérer la première). Par exemple :

- *F(CD-ROM)=1*
- *F(imprimante)=2*
- *F(plotter)=3*
- *F(rubban)=4*

Ainsi on garantit qu'il n'aura pas de cycles dans le graphe des ressources. On peut exiger seulement que aucun processus ne demande une ressource dont le numéro est inférieur aux ressources déjà allouées. Mais ceci n'est pas non plus la panacée, car, en général, le

nombre potentiel de ressources est si important qu'il est très difficile de trouver la bonne fonction F pour les numéroter.

7.8 Exercices

1. Citez deux exemples de ressources réutilisables et non-réutilisables. Par quelles entités du système sont gérées les ressources réutilisables et non-réutilisables ?
2. Donnez un exemple de programmation qui pourrait mener à l'interblocage entre un groupe de processus.
3. Expliquez brièvement les quatre approches que le SE pourrait utiliser pour manipuler des situations d'interblocage.
4. Étant donné le graphe d'allocation des ressources sur la figure 7.7, complétez le tableau d'allocation des ressources 7.3 et déterminez le vecteur $A = (\bullet, \bullet, \bullet)$.

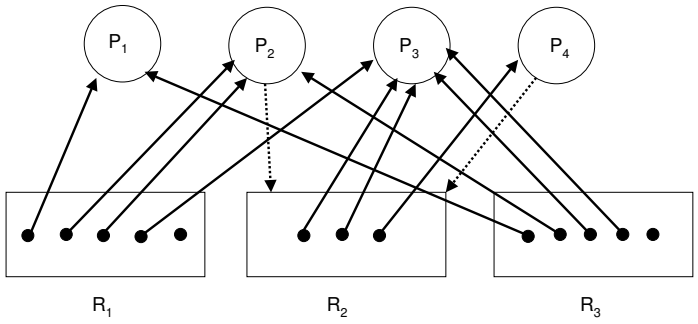

FIG. 7.7 – Graphe d'allocation des ressources.

Processus	R_1	R_2	R_3	R_1	R_2	R_3
P_1						
P_2						
P_3						
P_4						

TAB. 7.3 – Table de processus.

5. Considérez les tables 7.4, 7.5, et 7.6 qui indiquent les ressources disponibles d'un système, les réclamations maximales faites par des processus, et l'état d'allocation courant de ressources aux processus. Appliquez l'algorithme du banquier pour déterminer si l'état courant d'allocation est sûr.

Type de ressources (R_j)	Unités disponibles (C_j)
R_0	5
R_1	6
R_2	9
R_3	4

TAB. 7.4 – Ressources disponibles.

Processus (P_i)	R_0	R_1	R_2	R_3
P_0	3	1	3	2
P_1	0	4	5	1
P_2	2	0	2	3
P_3	2	3	4	1
P_4	1	2	3	0

TAB. 7.5 – Réclamations maximales.

6. Considérez un système composé de 4 processus (i.e. P_0, P_1, P_2, P_3), et de 3 types de ressources sériellement réutilisables (i.e. R_0, R_1, R_2). Le vecteur de ressources disponibles est $A = [3, 2, 2]$. Les conditions d'utilisation de ressources sont les suivantes :
   – Processus P_0 détient 1 unité de R_0, et requiert 1 unité de R_1.
   – Processus P_1 détient 2 unités de R_1, et requiert 1 unité de R_0 et R_2.
   – Processus P_2 détient 1 unité de R_0, et requiert 1 unité de R_1.
   – Processus P_3 détient 2 unité de R_2, et requiert 1 unité de R_0.
   Montrez le graphe d'allocation des ressources pour le système décrit ci-haut. Réduisez le graphe et indiquez s'il y a des processus en état d'interblocage.
7. Considérez un système composé de 4 processus (i.e. P_0, P_1, P_2, P_3), et de 3 types de ressources non-réutilisables (i.e. R_0, R_1, R_2). 1 unité chacune de R_0 et de R_2 est disponible. Les conditions d'utilisation de ressources sont les suivantes :
   – Processus P_0 requiert 1 unité de R_0 et 1 unité de R_2.
   – Processus P_1 produit R_0 et R_2, et requiert 1 unité de R_1.
   – Processus P_2 requiert 1 unité chacune de R_0 et de R_2.
   – Processus P_3 produit R_1, et requiert 1 unité de R_2.
   Montrez le graphe d'allocation des ressources pour le système décrit ci-haut. Réduisez le graphe et indiquez s'il y a des processus en état d'interblocage.
8. Étant donnée les tableaux 7.7, 7.8 et le vecteur $A = (2, 1)$, le système

Processus (P_i)	R_0	R_1	R_2	R_3
P_0	2	0	1	1
P_1	0	2	3	0
P_2	1	0	2	1
P_3	0	2	1	1
P_4	1	1	2	0
$\sum$	4	5	9	3

TAB. 7.6 – État d'allocation courant.

est dans un état sûr ou non sûr ?

Processus	R_1	R_2
P_0	7	2
P_1	1	3
P_2	1	1
P_3	3	0

TAB. 7.7 – État courant d'allocation.

Processus	R_1	R_2
P_0	9	5
P_1	2	6
P_2	2	2
P_3	5	0

TAB. 7.8 – État des demandes d'allocation.

9. Étant donné le graphe d'allocation des ressources de la figure 7.8.
   Il y a une réponse correcte parmi les 5 suivantes. Choisissez et justifiez :
   (a) Le graphe a un cycle et donc on peut assurer qu'il n'y a pas d'interblocage.
   (b) Le graphe a un un cycle et donc on peut assurer qu'il y a un interblocage.
   (c) Il y a une séquence de terminaison des processus qui ne produit pas d'interblocage.
   (d) Il y a un interblocage entre les processus P_1, P_2, P_5 P_6.
   (e) Aucune des réponses antérieures n'est correcte.

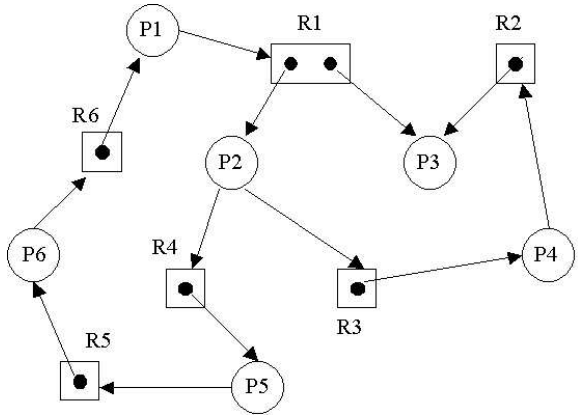

FIG. 7.8 – Graphe d'allocation des ressources.