Programación concurrente

Master de Computación

I Conceptos y recursos para la programación concurrente:

I.4 Patologías de las aplicaciones concurrentes.

J.M. Drake

M. Aldea

Patologías de los programas concurrentes

- Seguridad y vivacidad
- Interbloqueos

Patologías de los programas concurrentes

- Las patologías características de los programas concurrentes se pueden clasificar en uno de los dos tipos siguientes:
  - **Propiedades de seguridad**: No se debe ejecutar algo que conduzca a un error:
    - Entra en una sección crítica cuando está otro proceso.
    - Un proceso no respeta un punto de sincronismo.
    - Uno o varios procesos permanecen a la espera de un evento que nunca se producirá (interbloqueo).
  - **Propiedades de vivacidad**: Las sentencias que se ejecuten deben contribuir a un avance constructivo hacia el objetivo del programa:
    - Bloqueos activos: Dos procesos ejecutan sentencias que no hacen avanzar al programa.
    - Aplazamiento indefinido: Un programa se queda sin tiempo de procesador para avanzar.
    - Interbloqueo (también se considera un fallo de vivacidad)

Características de las patologías

- Un programa concurrente correcto no debe presentar ninguna patología.
- Las mejoras en seguridad y vivacidad suelen tener efectos contrapuestos:
  - La práctica de la ingeniería software pone énfasis en el diseño de la seguridad. Se asegura que el código no hace nada imprevisto o peligroso.
  - El mayor tiempo en el diseño de una aplicación concurrente se emplea en aspectos relacionados con la vivacidad. Evitar bloqueos, incrementar el rendimientos, etc.

Seguridad en sistemas concurrentes

- Las prácticas seguras de programación concurrente son generalizaciones de las reglas de programación secuencial segura.
- La seguridad en los programas concurrentes tiene un componente de carrera temporal específico: Hay que garantizar que sólo se acceda a los objetos cuando tienen un estado coherente (consistente):
  - Un objeto es coherente si satisface todos los invariantes entre su atributos que son inherentes a su naturaleza.
  - En los programas concurrentes se deben especificar todos los invariantes que caracterizan cada objeto. El diseño debe garantizar que se satisfacen cuando se accede a ellos.
- Fallos de seguridad son los conflictos de lectura/escritura a bajo nivel:
  - Conflicto de lectura/escritura: Un thread no puede escribir un nuevo valor en un atributo, mientras otro thread lo está leyendo.
  - Conflicto de escritura/escritura: Dos threads concurrentes no pueden escribir un mismo atributo.

Vivacidad y sistemas concurrentes

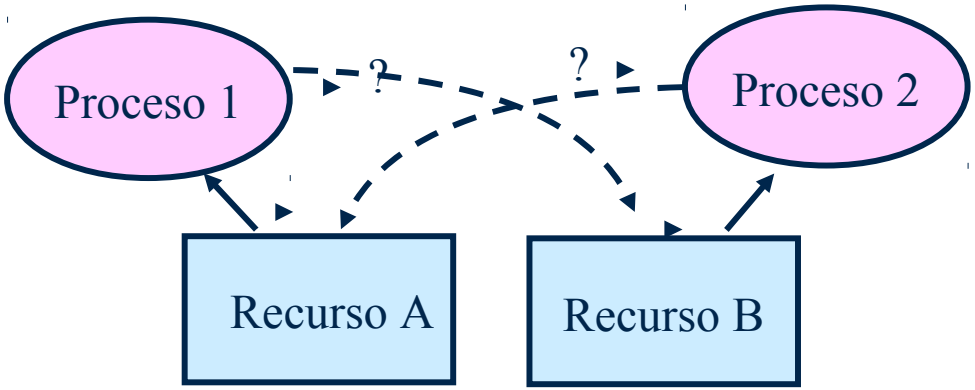
- En un programa concurrente hay muchas causas (aceptables) por las que un thread en estado activo, no se ejecuta:
  - **Bloqueo**: La ejecución se suspende porque se requiere un recurso que está siendo utilizado por otro thread.
  - **Espera**: La ejecución se suspende a la espera de un evento, bien de temporización o bien procedente de otro thread.
  - **Entrada**: La ejecución se suspende en espera de un evento de entrada o salida
  - **Expulsión**: Se interrumpe la ejecución porque otro thread de mayor prioridad pasa a ser ejecutando en el procesador.
  - **Fallo**: Se suspende porque se ha producido una excepción en el propio thread.

Fallos de vivacidad

- Una falta temporal de ejecución es normal y aceptable. Sin embargo una falta permanente o ilimitada de ejecución es un fallo de vivacidad:
  - **Interbloqueo**: Dependencia circular entre threads, en el que cada uno requiere para ejecutarse un recurso que posee el otro.
  - **Perdida de señales**: Un thread permanece inactivo porque comenzó a esperar a un evento después de que el evento se haya producido.
  - **Fallo continuado**: Una actividad falla repetidamente.
  - **Inanición**: La CPU que tiene que ejecutar el thread está permanente ocupada por la ejecución de otros threads con mayor prioridad de ejecución.
  - **Falta de recursos**: El sistema no dispone de los recursos que necesitan los threads para su ejecución.
  - **Fallo distribuido**: El thread requiere para ejecutarse el acceso a una maquina remota que no está accesible.
  - **Inversión de prioridad no acotada**: Un thread de alta prioridad a la espera de un recurso debe esperar por threads de prioridad menor que no utilizan el recurso

Interbloqueos

- Los interbloqueos están relacionados con la reserva no ordenada de recursos a los que se accede en exclusión mutua.
- Ejemplo:
  - Los procesos P_1 y P_2 utilizan los recursos R_A y R_B.
  - El proceso P_1 toma R_A y es expulsado de la CPU por P_2.
  - El proceso P_2 toma R_B y queda a la espera de R_A.
  - El proceso P_1 trata de tomar R_A queda a la espera.
  - Los procesos P_1 y P_2 permanecen indefinidamente bloqueados.



Condiciones bajo las que hay bloqueos.

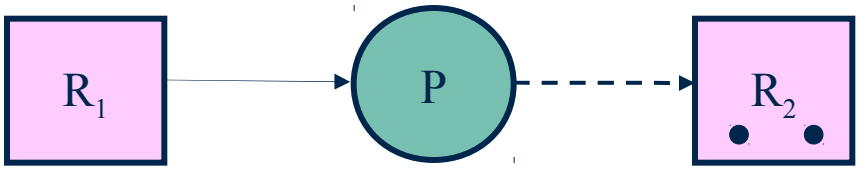
- Para que un interbloqueo pueda ocurrir, se requiere que se produzcan simultáneamente estas condiciones
  - Los procesos utilizan recursos bajo régimen de exclusión mutua.
  - Los procesos mantienen reservados los recursos tomados mientras esperan los otros recursos que necesitan.
  - No existe capacidad de despojar a los procesos de los recursos ya tomados.
  - Existe una cadena circular de requerimientos y reservas que conducen al interbloqueo.

Condiciones con las que eludir los interbloqueos.

- Cuando las condiciones de interbloqueos se pueden presentar los bloqueos se pueden eludir con la siguientes estrategias:
  - Puede establecerse que si un thread se bloquea por falta de un recurso, libere todos los que tenía tomados en espera del que falta.
  - Puede establecerse que los threads tomen en bloque todos los recursos que necesitan.
  - Se puede evitar que ejecute un thread que pudiera tomar un recurso que pudiera conducir a un interbloqueo
  - El controlador puede tener capacidad de retirar la cesión de los recursos asignados a los procesos bloqueados.
  - Puede establecerse un orden de reserva de recursos que haga imposible las dependencias circulares.

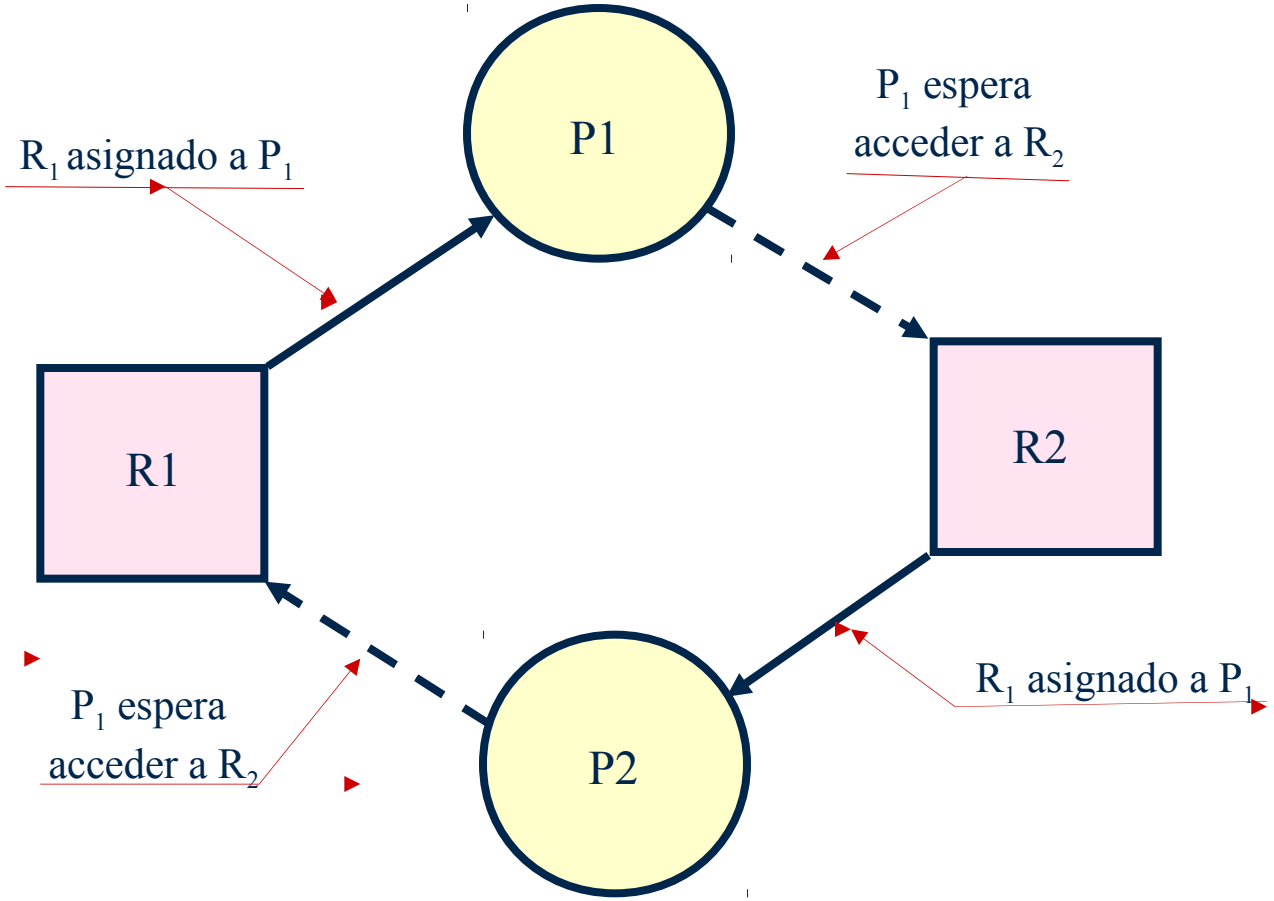
Grafo de reserva de recursos

- Es un grafo orientado que representa el estado de asignación y requerimiento de recursos por parte de los procesos de una aplicación:
  - Nudos: Existe un nudo en el grafo por cada proceso y por cada recurso de la aplicación.
  - Arcos:
    - Cuando un recurso está asignado a un proceso se establece un arco (continuo) del nudo que representa el recurso al nudo que representa al proceso.
    - Cuando un proceso tiene requerido un recurso se establece un arco (discontinuo) del nudo que representa el proceso al nudo que representa el recurso.
  - Cuando un recurso tiene varias replicas, se introducen tantos puntos como replicas tenga. Cada una de ellas puede ser asignada independientemente.

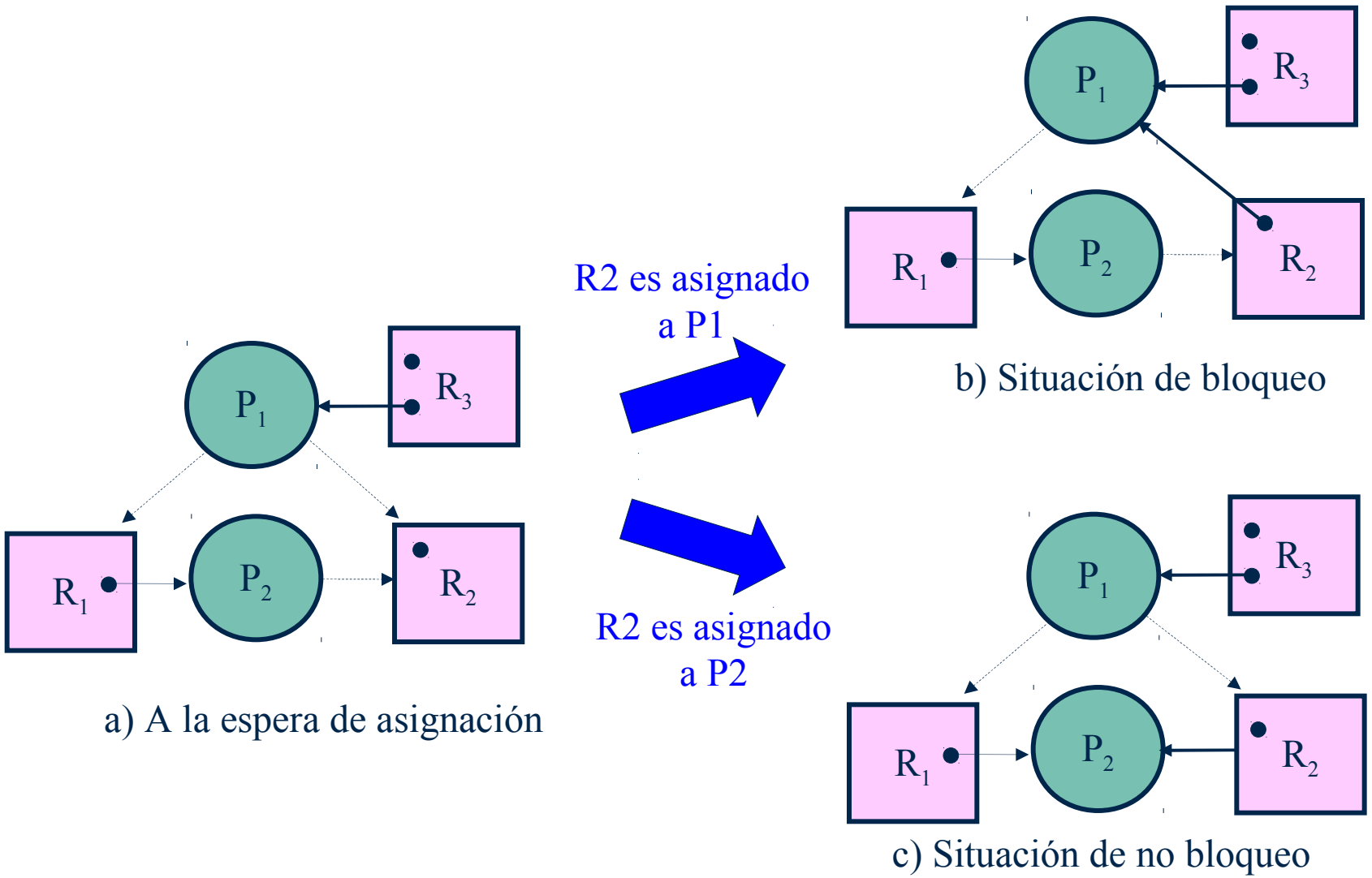


Métodos de detección de recursos.

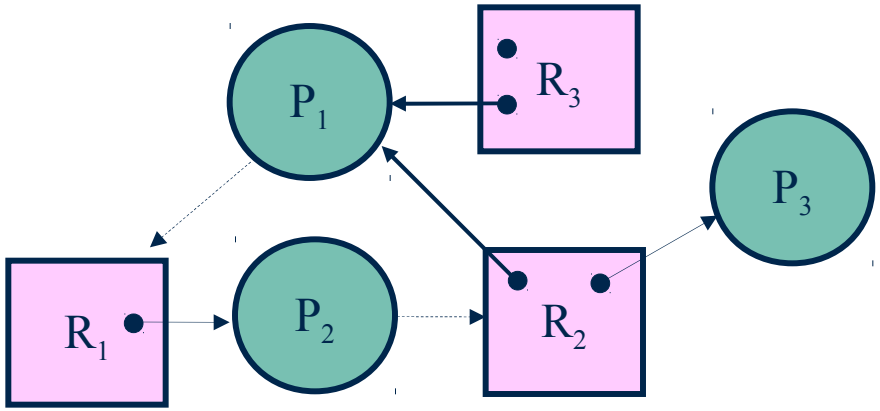
Grafo de reserva de recursos.



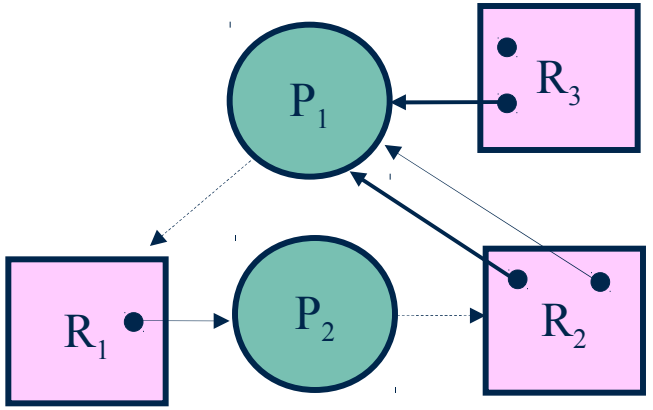
Ejemplos de grafos de asignación de recursos



Ejemplos con recursos con replicas

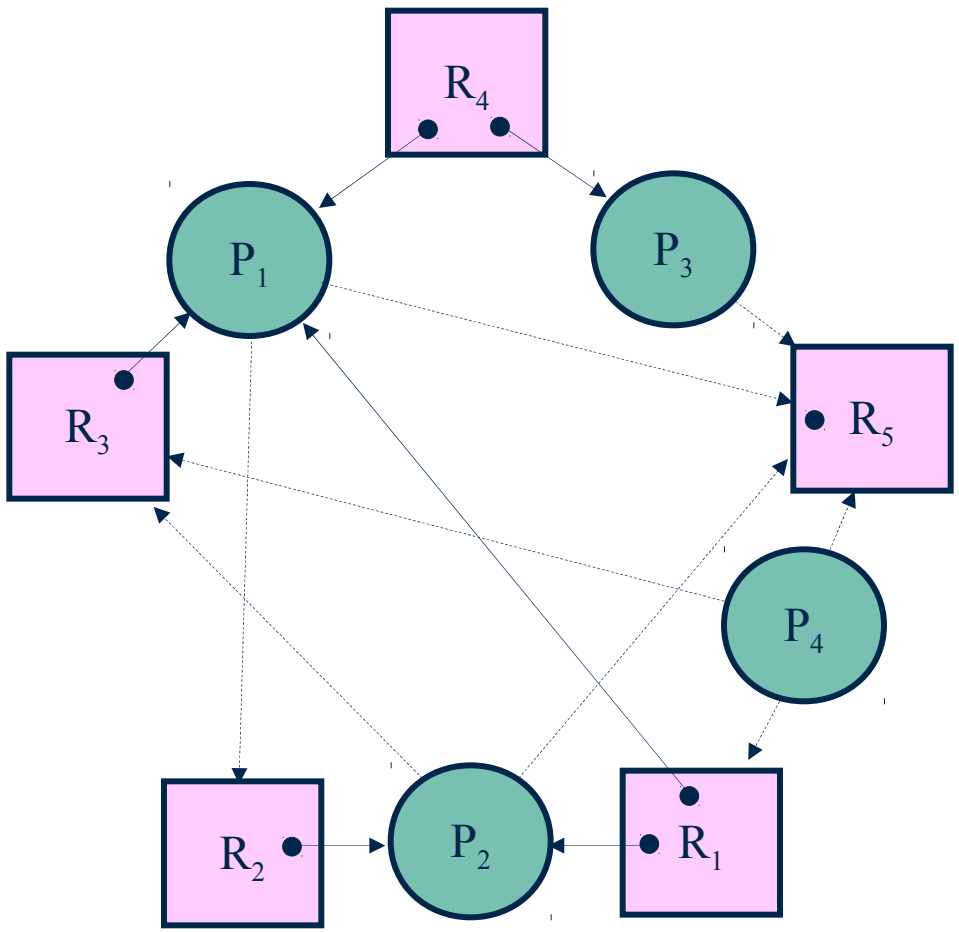


(aunque hay bucle, no hay bloqueo



(Hay bucle, y hay bloqueo

Estructura de datos para la gestión de grafos de asignación.



R

2	1	1	2	1

R: Recursos del sistema

R

P

1	0	1	1	0
1	1	0	0	0
0	0	0	1	0
0	0	0	0	0

A: Matriz de asignación

R

P

0	1	0	0	1
0	0	1	0	1
0	0	0	0	1
1	0	1	0	1

B: Matriz de requerimientos

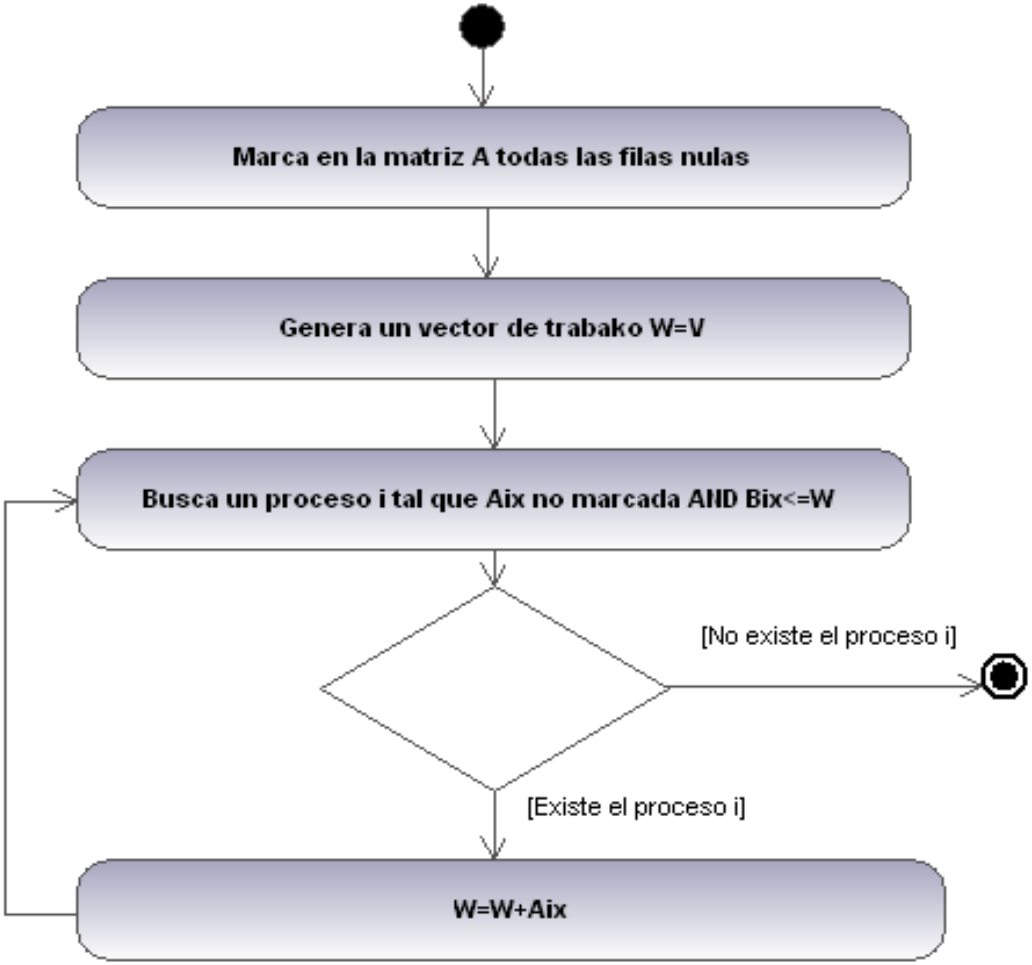
R

0	0	0	0	1

V: Recursos disponibles

Un algoritmo para detección de bloqueos.

- El objetivo del algoritmo es buscar los procesos que no son parte de un bloqueo.



Un algoritmo para detección de bloqueos.

- Se marcan todas las filas de la matriz de asignación A nulas.
  - obvio, puesto que un proceso que no tiene asignados recursos no puede ser parte de un bloqueo.
- Se genera el vector de trabajo W y se inicializa al vector V de recursos disponibles.
- Se busca un proceso i, que corresponda a una fila de A que no este marcada, y que satisfaga que Bi<=W
  - Si no existe tal proceso se termina la búsqueda.
  - Si existe se marca la fila i de la matriz A y se revalua W=W+Ai
- Se vuelve al punto anterior
- Los bloqueos que existan corresponden a los procesos que corresponden a las filas de la matriz A que no están marcadas

Criterios de arbitrar un bloqueo.

- Detectado un bloqueo:
  - Abortar todos los procesos afectados por el bloqueo.
  - Abortar de uno en uno, los procesos que intervienen en el bloqueo, hasta que el mismo desaparece.
  - Liberar desde fuera los recursos afectados, uno a uno hasta que el bloqueo desaparece.

- Criterios de liberación de los recursos:
  - La prioridad de los procesos.
  - El tiempo que cada proceso lleve ya de ejecución.
  - El tiempo que se prevé que aún resta para concluir el proceso.
  - El número de recursos que requieren los procesos.
  - La naturaleza de los recursos que son requeridos por cada proceso.

Algoritmo del banquero.

- Existe un gestor que administra los recursos del sistema.
- Se conocen el número máximo de recursos que puede requerir cada proceso.
- El gestor no entrega un recurso si no tiene garantía de que aún quedan recursos para que al menos uno de los procesos en ejecución pueda terminar.

Proceso	Requiere
A	4
B	6
C	8
Recursos:	12

Estado admitido	
A ←	1
B ←	4
C ←	5
Quedan:	2

Estado no admitido	
A ←	2
B ←	4
C ←	5
Quedan:	1