El interbloqueo

Situación en la que se encuentran un conjunto de procesos, (al menos dos), tal que cada proceso del conjunto espera la ocurrencia de un evento que sólo puede ser provocado por otro proceso del mismo conjunto.

Los recursos

Tipos de recursos:

1. Recurso apropiable

Se puede tomar del proceso que lo posee sin provocar efectos dañinos

2. Recurso no apropiable

No se puede tomar de su poseedor activo sin provocar un error.

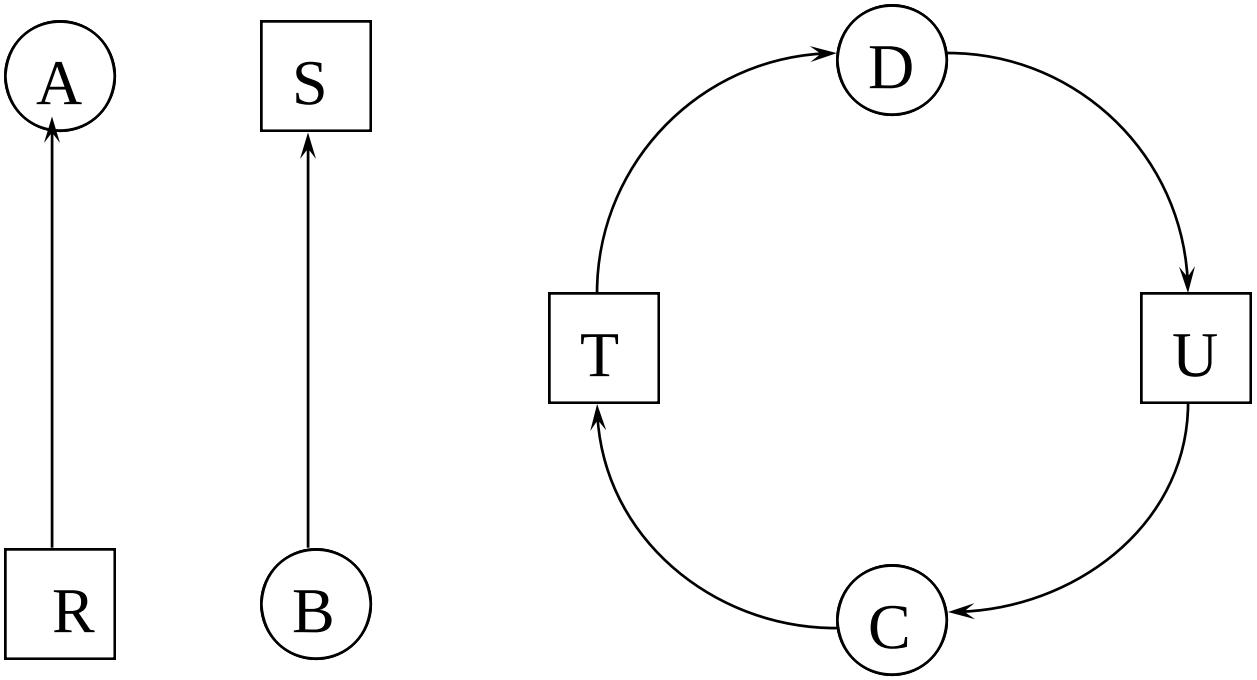
Secuencia de eventos necesarios para utilizar un recurso:

1. Solicitar el recurso
2. Utilizar el recurso
3. Liberar el recurso

Condiciones interbloqueo

1. Condición de exclusión mutua
2. Condición de posesión y espera
3. Condición de no apropiación
4. Condición de espera circular

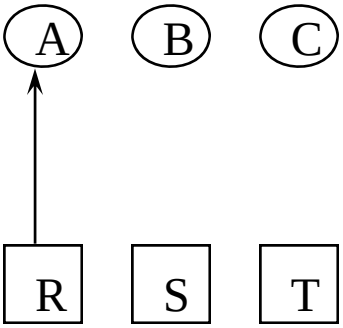
Modelando interbloqueos: el modelo de Holt



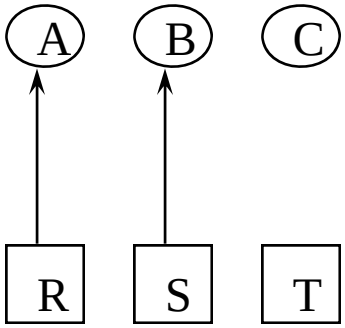
Gráficas de asignación de recursos. (a) Posesión de un recurso. (b) Solicitud de un recurso. (c) Bloqueo.

A	B	C
Solicitud de R	Solicitud de S	Solicitud de T
Solicitud de S	Solicitud de T	Solicitud de R
Liberación de R	Liberación de S	Liberación de T
Liberación de S	Liberación de T	Liberación de R

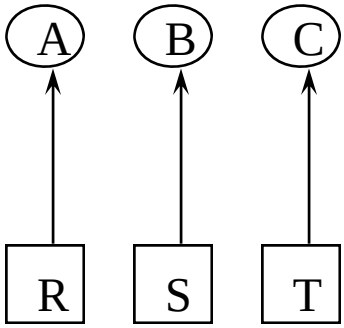
1 A Solicitud R
2 B Solicitud S
3 C Solicitud T
4 A Solicitud S
5 B Solicitud T
6 C Solicitud R
bloqueo



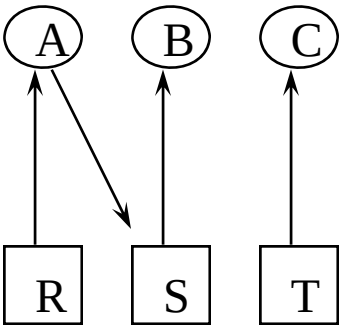
(1)



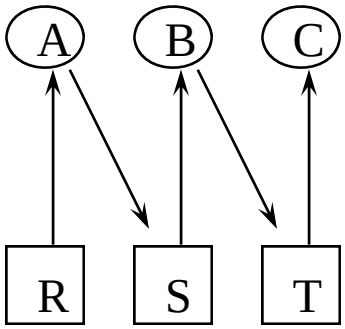
(2)



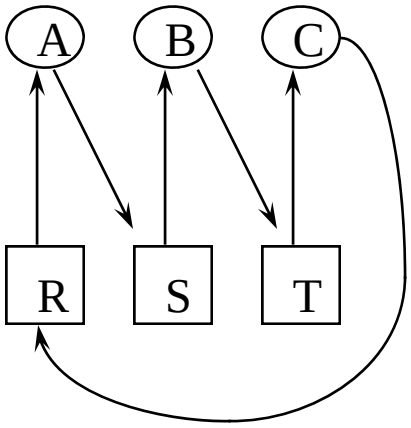
(3)



(4)

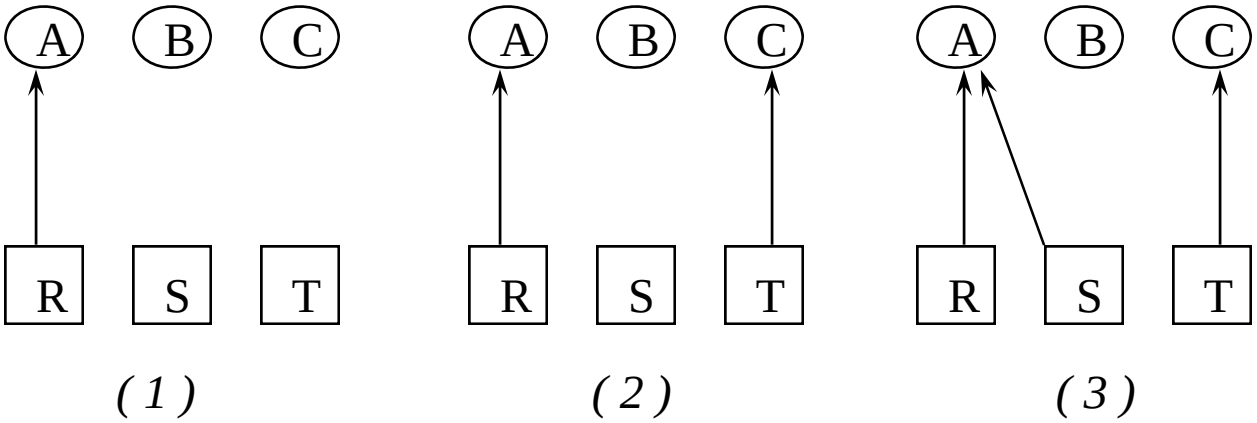


(5)

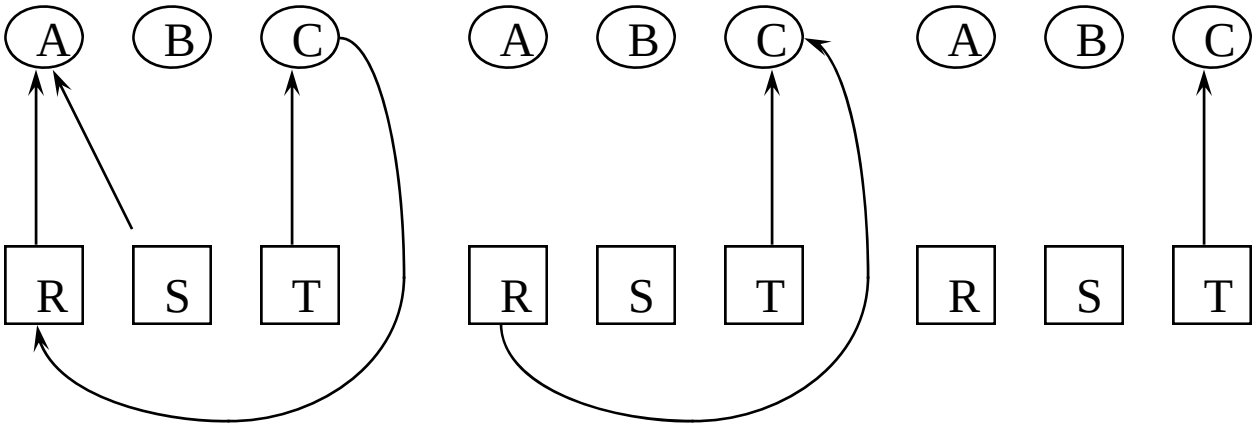


(6)

1. A Solicitud R
2. C Solicitud T
3. A Solicitud S
4. C Solicitud R
5. A libera R
6. A libera S
   no existe
   bloqueo



(1) *(2)* *(3)*



(4) *(5)* *(6)*

Areas de investigación del Interbloqueo

Prevención del interbloqueo

Condicionar un sistema para quitar cualquier posibilidad de ocurrencia de interbloqueo.

Evitar el interbloqueo

No precondiciona al usuario a quitar todas las posibilidades de interbloqueo.

Detectar el interbloqueo

Determinar si un interbloqueo se dio o no.

Recuperarse del interbloqueo

Limpiar un sistema de interbloqueos, una vez que fueron detectados.

Prevención del Interbloqueo

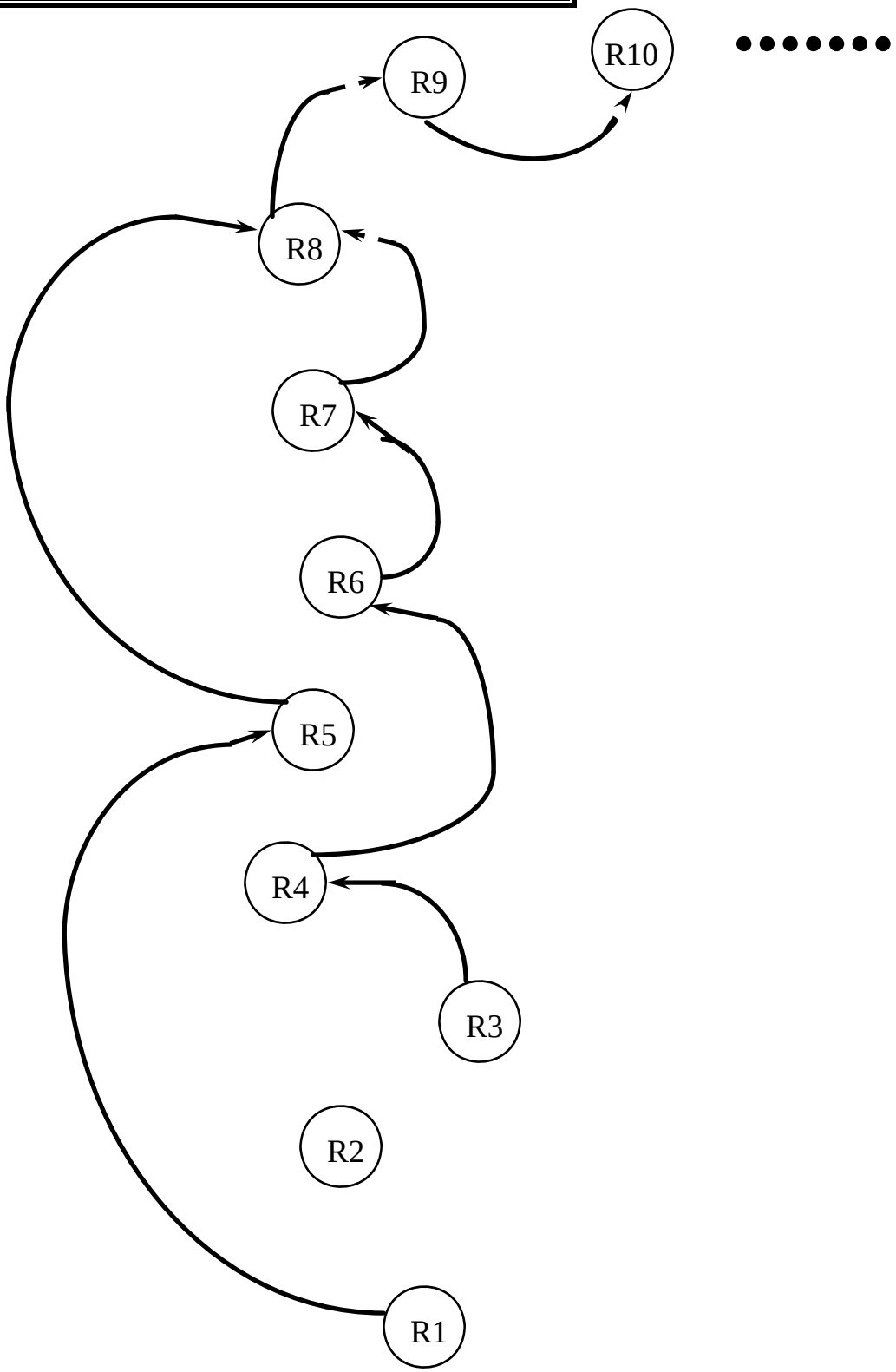
Havender (Hv68)

+ *Negación de la condición de no apropiación*
  Cada proceso debe de hacer todas sus requisiciones de recursos y no puede continuar hasta que todo le haya sido otorgado.

+ *Negación de la condición de posesión y espera*
  Si a un proceso que retiene recursos asignados se le negó un recurso, éste debe de liberar todos los que tenía. Si es necesario los puede pedir después.

+ *Negación de la espera circular*
  Imponer un orden lineal de los tipos de recursos en todos los procesos. Por ejemplo, si un proceso tiene asignados recursos de un determinado tipo, éste sólo puede pedir recursos de tipos posteriores en el ordenamiento.

Ejemplo Ordenamiento Lineal Recursos



Resumen metodos prevención interbloqueo

Exclusión Mutua

Realizar un spooling general

Posesión y espera

Solicitar todos los recursos al principio

No apropiación

Retirar los recursos

Espera circular

Ordenar los recursos en forma numérica

Evitando el interbloqueo el algoritmo del banquero

Si las condiciones necesarias para que se produzca un interbloqueo no se pueden eliminar

=> tener cuidado con la asignación de recursos

Propuesto por Dijkstra.

Banquero:

Involucra un banquero que realiza prestamos y recibe pagos de una determinada fuente de capital.

Para autorizar un préstamo es necesario que el cliente sea soluble y que no deje en banca rota al banco.

Objetivo algoritmo:

Dejar al sistema en un estado seguro después de asignar recursos

Edo. Seguro: la situación total de los recursos es tal que los usuarios podrán terminar su trabajo

Edo. Inseguro: estado que me puede llevar a un interbloqueo

Ejemplo Estado Seguro e Inseguro

Ejemplo de estado seguro:

Estado I

	Préstamo Actual		Necesidad Máxima
Usuario 1	1		4
Usuario 2	4		6
Usuario 3	5		8
Disponibles		2	

Ejemplo de estado inseguro

Estado II

	Préstamo Actual		Necesidad Máxima
Usuario 1	8		10
Usuario 2	2		5
Usuario 3	1		3
Disponibles		1	

Paso de transición de edo. seguro a inseguro

Estado III

	Préstamo Actual	Necesidad Máxima
Usuario 1	1	4
Usuario 2	4	6
Usuario 3	5	8
Disponibles	2	

Estado IV

	Préstamo Actual	Necesidad Máxima
Usuario 1	1	4
Usuario 2	4	6
Usuario 3	6	8
Disponibles	1	

El algoritmo del banquero

- Varios ejemplares de recursos del mismo tipo
- Algoritmo puede extenderse a recursos de diferentes tipos
- Consideremos la asignación de una cantidad t de recursos (p.e. unidades de cinta) entre un número u de usuarios:
  - cada usuario especifica el número de cintas que necesitará durante la ejecución de su trabajo en el sistema
  - el sistema aceptará la petición de un usuario si la necesidad máxima de ese usuario no es mayor que la cantidad de recursos disponibles
  - un usuario puede obtener o liberar unidades de cinta una a una
  - un usuario puede esperar para obtener una cinta adicional, sin embargo la espera es finita
  - el usuario debe garantizar al sistema que las unidades de cinta serán utilizadas y liberadas en un tiempo finito
- El algoritmo permite la asignación de unidades de cinta a los usuarios solamente cuando la asignación permita/conduzca a estados seguros y no a estados inseguros

Estructuras datos algoritmo banquero

- Necesario representar el estado del sistema, para determinar si se provocará un estado seguro o inseguro
- Muchas estructuras datos deben actualizarse para implementar el algoritmo del banquero
- Variables necesarias:
  - t: número máximo de recursos
  - n: número de procesos
  - prestamo(i): cuantos recursos tiene asignado P_i
  - max(i):número máximo de recursos solicitados por P_i
  - necesidad(i): el resto de los recursos que P_i puede solicitar
  - peticion(i): P_i requiere tantos procesos como peticion(i) almacene
  - dispo: número de recursos disponibles

$$\text{dispo} = t - \sum_{i=n}^{n} \text{prestamo}(i)$$

El algoritmo del Banquero

1. Si (peticion(i) $\leq$ necesidad(i))

   ir al paso 2

   sino

   error --> proceso excedió su necesidad máxima

2. Si (peticion(i) $\leq$ dispo)

   ir al paso 3

   sino

   Pi debe esperar ya que no hay recursos disponibles

3. El sistema asigna los recursos pedidos al proceso P_i modificando el estado del sistema de la siguiente forma:

```
dispo = dispo - petición(i);
prestamo(i) = prestamo(i) + petición(i);
necesidad(i) = necesidad(i) - petición(i);
```

Si el estado después de la asignación es seguro:

=> la transacción es completada y P_i obtiene sus recursos.

Si el estado es inseguro:

=> entonces P_i debe esperar por petición(i) y el antiguo estado de asignación de recursos es reestablecido.

Algoritmo Detección de Estado del Sistema

Un algoritmo para verificar si el sistema de detección se encuentra en estado seguro o inseguro es el siguiente:

1. Declaración e inicialización de variables:
   trabajo = dispo;
   terminado[1...n] inicializado en falso;

2. Encontrar i de forma que:

   (a) terminado [i] = falso;

   (b) necesidad(i) $\leq$ trabajo

si i no existe ==> ir a 4

si no ==> ir a 3

3. trabajo = trabajo + prestamo(i);
terminado[i] = verdadero;
ir a 2;

4. si terminado[i] = verdadero (para toda i)
   ==> sistema está en un estado seguro

   sino

   el sistema esta en un estado inseguro

Ejemplo algoritmo

Considere el siguiente escenario:

Se cuenta con 4 recursos disponibles, y el resto esta distribuido de la siguiente forma:

Proceso	RA *prestamo(i)*	Nec Max.	Necesidad *necesidad(i)*
P_1	1	4	3
P_2	2	6	4
P_3	5	8	3

Ahora bien el proceso P_2 solicita 2 recursos, por lo que la tabla anterior se actualiza:

Proceso	RA *prestamo(i)*	Nec Max.	Necesidad *necesidad(i)*
P_1	1	4	3
P_2	4	6	2
P_3	5	8	3

dispo = 4 -2 = 2

Ahora hay que verificar que no se crea un edo. inseguro

Paso 1)
trabajo = dispo = 2

	P_1	P_2	P_3
préstamo(i)	1	4	5
necesidad(i)	3	2	3
terminado(i)	F	F	F

Paso 2)

i tal que (terminado[i] = falso) y (necesidad[i] ≤ trabajo)

==> i = 2

Paso 3)

trabajo = trabajo + prestamo[2] = 2 + 4 = 6
terminado[2] = V

la tabla queda de la siguiente forma:

	P_1	P_2	P_3
préstamo(i)	1	4	5
necesidad(i)	3	2	3
terminado(i)	F	V	F

Paso 2)

i tal que (terminado[i] = falso) y (necesidad[i] ≤ trabajo)

==> i = 1

Paso 3)

trabajo = trabajo + prestamo[1] = 6 + 1 = 7

terminado[1] = V

la tabla queda de la siguiente forma:

	P_1	P_2	P_3
préstamo(i)	1	4	5
necesidad(i)	3	2	3
terminado(i)	V	V	F

Paso 2)

i tal que (terminado[i] = falso) y (necesidad[i] ≤ trabajo)

==> i = 3

Paso 3)

trabajo = trabajo + prestamo[3] = 7 + 5 = 12

terminado[3] = V

la tabla queda de la siguiente forma:

	P_1	P_2	P_3
préstamo(i)	1	4	5
necesidad(i)	3	2	3
terminado(i)	V	V	V

Paso 3) No hay => **Paso 4**: terminado[i]=V para toda i

La detección del interbloqueo

- Sistema no intenta evitar los bloqueos, sino que dejan que aparezcan

- Deja que aparezcan y después lleva a cabo una acción para recuperarse

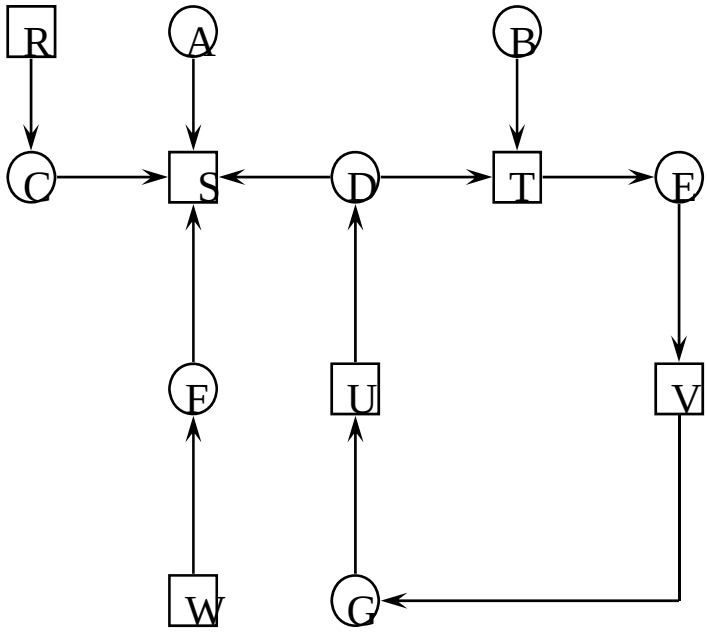
- Existen dos tipos:

1. Detección interbloqueo con un recursos de cada tipo

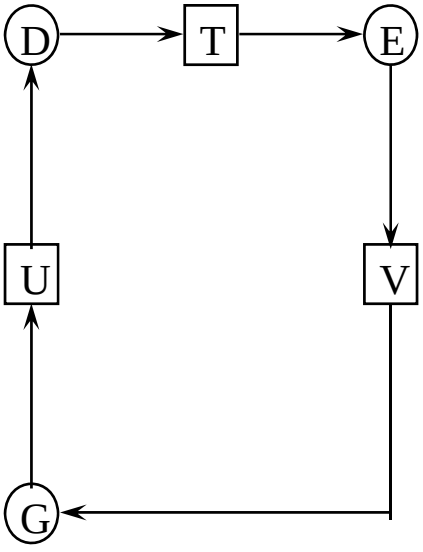
2. Detección interbloqueo de varios recursos de cada tipo

Detección interbloqueo (un recurso de cada tipo)

En el caso de que solo exista un recurso de cada tipo, para detectar un interbloqueo basta con detectar un ciclo en el grafo de asignaciones



(a) Gráfica de recursos



(b) Ciclo extraido de (a)

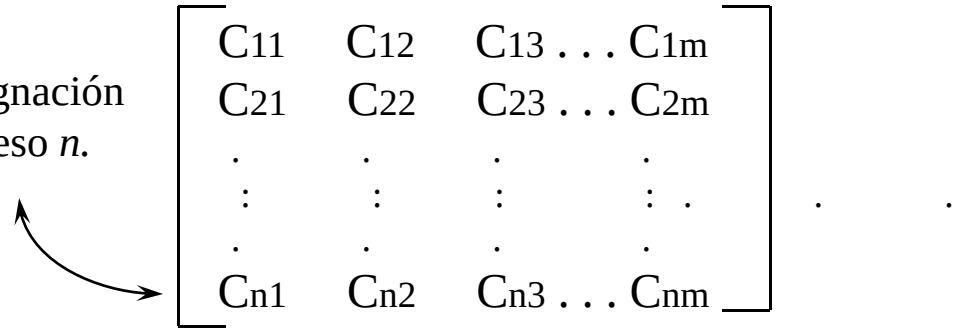
Detección Interbloqueo (varios recursos de cada tipo)

Es necesario utilizar dos matrices: asignación y disponible

Recursos en existencia
(E_1,E_2,E_3,..., E_m)

Matriz de asignación actual

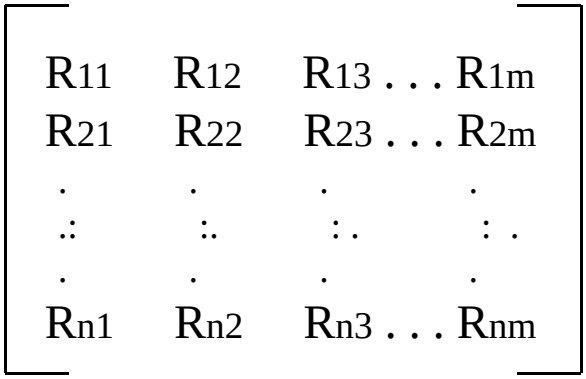
Renglón *n* es la asignación actual para el proceso *n*.



Recursos disponibles
(A_1,A_2,A_3,..., A_m)

Matriz de solicitudes

El renglón 2 es lo que necesita el proceso 2



Algoritmo de asignación

Cada recurso debe estar asignado o disponible, por lo que:

$$\sum_{i=1}^{m} C_{ij} + A_j = E_j$$

Algoritmo se basa en la comparación de vectores:

$A \leq B$ cada elemento vector A es menor o igual que el elemento correspondiente de B, es decir:

$$A \leq B \Leftrightarrow Ai \leq Bi \quad \forall \quad (0 \leq i \leq m)$$

El algoritmo de detección de bloqueos es el siguiente:

1. Se busca un proceso no marcado P_i, para que el i-ésimo renglón de *R* sea menor o igual que *A*
2. Si se encuentra tal proceso, se suma el i-ésimo renglón de *C* a *A*, se marca el proceso y se regresa al paso 1.
3. Si no existe tal proceso, el algoritmo termina

Al concluir el algoritmo, los procesos no marcados, de existir alguno, están bloqueados.

Ejemplo de aplicación

Recursos en existencia

Unidades de cinta | Ploters | Impresoras | CD ROM

E = (4 2 3 1)

Recursos disponibles

Unidades de cinta | Ploters | Impresoras | CD ROM

A= (2 1 0 0)

Matriz de asignación actual

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

Matriz de solicitudes

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

Recuperación de un interbloqueo

Una vez que se detectó un interbloqueo:

se necesita una forma de recuperarse y lograr que el sistema continúe nuevamente

Existen varias técnicas, (ninguna de ellas interesante):

1. Recuperación mediante la apropiación

2. Recuperación mediante rollback

3. Recuperación mediante la eliminación de procesos

Apropiación

- Selección de la víctima
  - cuales recursos de cuales procesos se deben expropiar
  - orden de expropiación para minimizar el costo
  - factores costo: número de recursos que un proceso tiene asignados, y el tiempo que un proceso bloqueado ha consumido
- Retroceso (rollback)
  - que debe hacerse con el proceso víctima
  - no puede continuar con su ejecución normal
  - se debe regresar el proceso a un estado seguro y reiniciarlo en ese estado
- Inanición (hambruna)
  - como asegurarse de que no habrá inanición
  - como garantizar que no siempre se expropiaran los recursos del mismo proceso
  - un proceso debe ser elegido como víctima un número finito, (chico) de veces
  - solución común: incluir número retrocesos como factor de costo

Eliminación

- Dos opciones:
  - abortar todos los procesos bloqueados
  - abortar un proceso a la vez hasta eliminar el ciclo de bloqueo mutuo
- Terminación forzada de un proceso podría no ser fácil (proceso estaba actualizando un archivo; imprimiendo)
- Factores en la selección de los procesos
  - Prioridad del proceso
  - Tiempo que ha trabajado el proceso y tiempo que le resta antes de llevar a cabo su tarea designada
  - cuantos recursos ha usado el proceso y de que tipo (los recursos se pueden expropiar fácilmente)
  - cuantos recursos adicionales necesita el proceso para terminar su tarea
  - cuantos procesos habrá que abortar
  - los procesos son interactivos o por lotes

Combinando métodos

- Investigadores argumentan que ninguna técnica estrategia por si sola es apropiada para el manejo del deadlock
- Una posibilidad es combinar ciertas técnicas y permitir el uso de la estrategia para cada clase de recursos
- Considerar sistema que maneja cuatro de los siguientes recursos:
  - recursos internos: recursos usados por el sistema como un bloque de control de proceso
  - memoria central: memoria usada por trabajo de un usuario
  - recursos de trabajo: dispositivos asignables (como unidades de cinta) y archivos
  - espacio intercambiable: espacio para cada trabajo de usuario en el almacenamiento auxiliar
- Solución mixta:
  - recursos internos: ordenamiento de recursos
  - memoria principal: expropiación
  - recursos trabajos: evitar, ya que información se puede obtener de las tarjetas de control de trabajos
  - espacio intercambiable: preasignación