Tema 3: Planificación de recursos

1. Caracterización del interbloqueo
2. Modelación del interbloqueo
3. Métodos para tratar el interbloqueo

Caracterización del interbloqueo

- **Recurso**: cualquier elemento del sistema que pueda ser utilizado por un proceso (software o hardware)
- **Tipos de recursos**:
  - Apropiables: aquellos que se pueden tomar del proceso que lo posee sin producir efectos dañinos
  - No apropiables
- Los **interbloqueos** se suelen relacionar con los recursos no apropiables. Los interbloqueos sobre recursos apropiables se pueden resolver fácilmente con una reasignación de recursos de un proceso a otro (CPU, memoria, ...)

Caracterización del interbloqueo (y II)

- Secuencia de sucesos para usar un recurso no apropiable:
    1. Pedir el recurso (si no está libre, el SO puede bloquear al proceso o devolverle un código de error)
    2. Usar el recurso
    3. Liberar el recurso

- **Definición de interbloqueo** (*deadlock*): Un conjunto de procesos está interbloqueado si cada proceso del conjunto está esperando un suceso que sólo otro proceso del conjunto puede producir

Condiciones de Coffman

- Para que se dé un interbloqueo, han de cumplirse simultáneamente las cuatro condiciones siguientes :

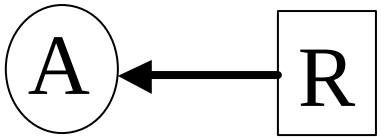
1. **Exclusión mutua**: un recurso o está asignado a un proceso o está disponible
2. **Mantener y esperar**: cada proceso continúa manteniendo los recursos asignados con anterioridad mientras espera adquirir otros
3. **No apropiación**: los recursos asignados a un proceso sólo pueden volver al sistema cuando el proceso voluntariamente los libere
4. **Espera circular**: los procesos forman una cadena circular donde cada proceso solicita un recurso que tiene asignado el siguiente miembro de la cadena

Tema 3: Planificación de recursos

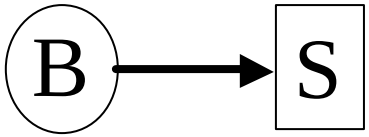
1. Caracterización del interbloqueo
2. Modelación del interbloqueo
3. Métodos para tratar el interbloqueo

Modelación del interbloqueo

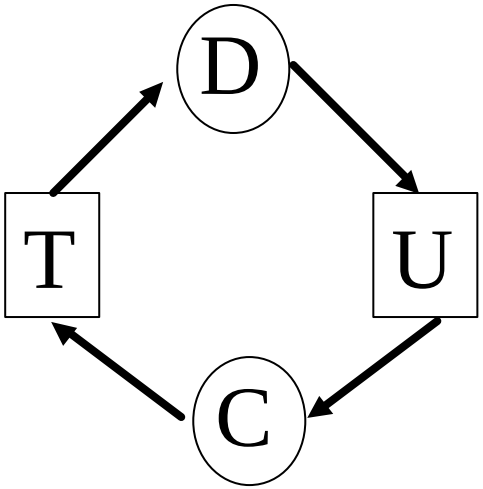
- Holt (1972) mostró una forma de representar las cuatro condiciones mediante un grafo dirigido: **grafo de asignación de recursos**
- Nodos:
  - Círculos ↝ Procesos
  - Rectángulos ↝ Recursos
- Arcos:



A tiene asignado el recurso R



B solicita (y espera) el recurso S



Interbloqueo
ciclo C-T-D-U-C

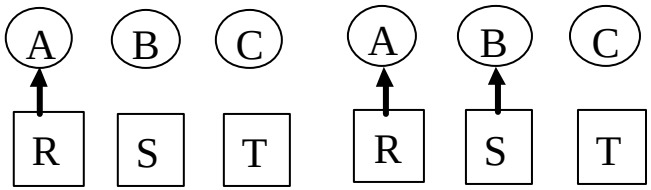
Ejemplo del uso de grafos de asignación de recursos

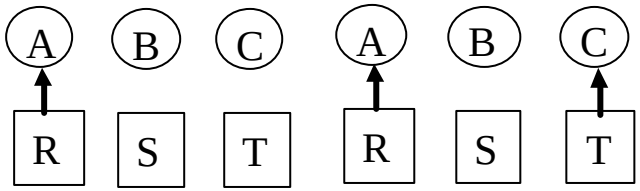
Interbloqueo

No Interbloqueo

A

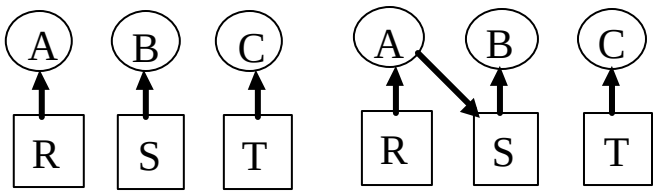
Pedir R
Pedir S
Liberar R
Liberar S

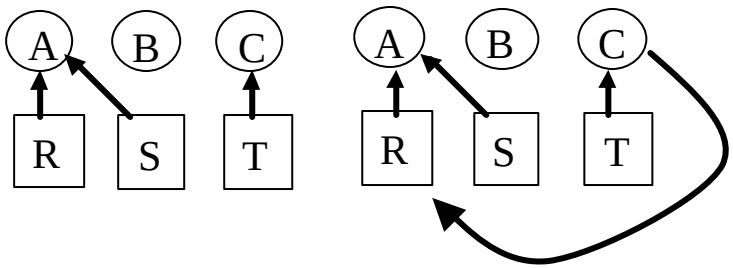




B

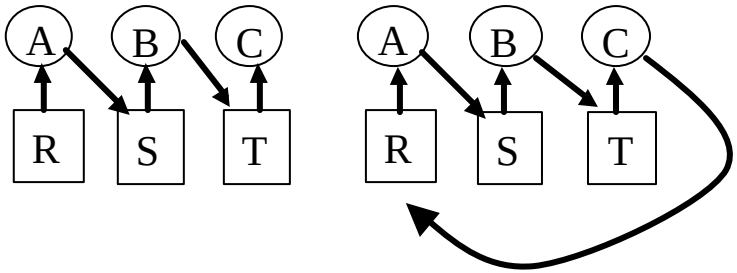
Pedir S
Pedir T
Liberar S
Liberar T

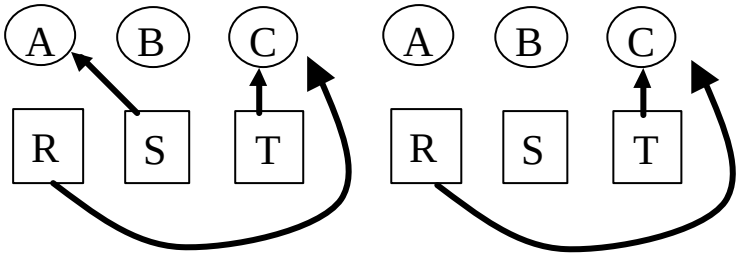




C

Pedir T
Pedir R
Liberar T
Liberar R





Tema 3: Planificación de recursos

1. Caracterización del interbloqueo
2. Modelación del interbloqueo
3. Métodos para tratar el interbloqueo

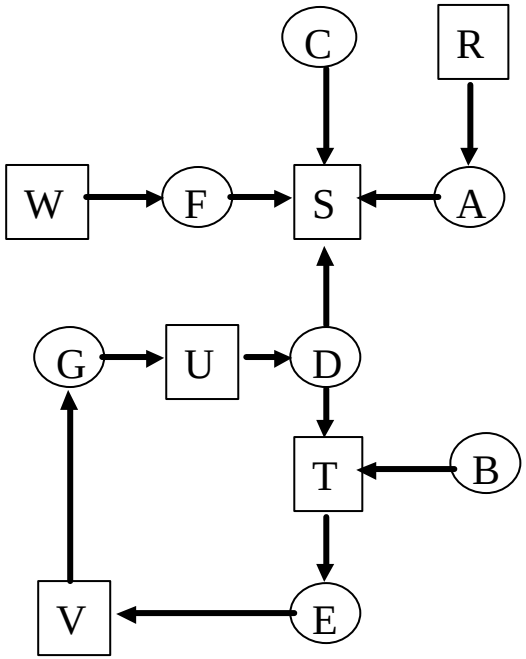
Métodos para tratar el interbloqueo

- **Ignorar el problema**: Algoritmo del avestruz
- **Detección y eliminación**: Deja que ocurra el interbloqueo, lo detecta y, entonces, lo elimina
- **Evitar el interbloqueo**: Trata de controlar al sistema para que nunca ocurran los interbloqueos, realizando una cuidadosa asignación de recursos
- **Prevención del interbloqueo**: Intenta eliminar alguna de las condiciones de Coffman

Detección y eliminación

- Detección (para un recurso de cada tipo): usa un algoritmo para detectar ciclos en el grafo de recursos

1 Para todo nodo hacer:
2 Inicializar una lista L a vacío y poner todos los arcos como no marcados
3 Añadir el nodo a L ¿existe ya?
    si: hay un ciclo. Terminar
    no: ir a 4
4 Comprobar para este nodo si hay arcos no marcados que salgan de él
    si: ir a 5
    no:ir a 6
5 Elegir uno, marcarlo, ir al nodo al que apunta y volver a 3
6 ¿Es el nodo inicial?
    si: fin, no hay ciclos
    no: eliminar nodo actual de L , ir nodo anterior y volver a 4



Detección y eliminación (y II)

- Detección (varios recursos de un tipo). Se utiliza la siguiente información:

  E: Vector de recursos existentes: Cada elemento representa el nº de recursos de un tipo que hay en el sistema

  D: Vector de recursos disponibles

  A: Matriz de recursos asignados: A_{ij} ≝ unidades que el proceso número i tiene asignadas del recurso j

  S: Matriz de recursos solicitados: S_{ij} ≝ unidades que el proceso número i tiene solicitadas del recurso j

- **Invariante**:

$$\boxed{\sum_{i=1,n} A_{ij} + D_j = E_j}$$

donde ***n***= nº procesos

Detección y eliminación (y III)

- Algoritmo:

1 Partir con todos los procesos sin marcar
2 Si hay algún proceso no marcado que cumpla que su fila en S es menor o igual que D
    - Añadir su fila en A a D y marcarlo
    - Volver a 2
3 ¿Hay procesos no marcados?
    Si: Interbloqueo
    No: no existe interbloqueo

Detección y eliminación (y IV)

- Ejemplo: Tenemos un sistema con tres procesos y los siguientes recursos: 4 Impresoras, 2 Unidades de cinta, 3 Plotters y 1 CD-ROM

E = (4 2 3 1)

D = (2 1 0 0)

Matriz A

	I	U	P	CD
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

Recursos asignados

Matriz S

	I	U	P	CD
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

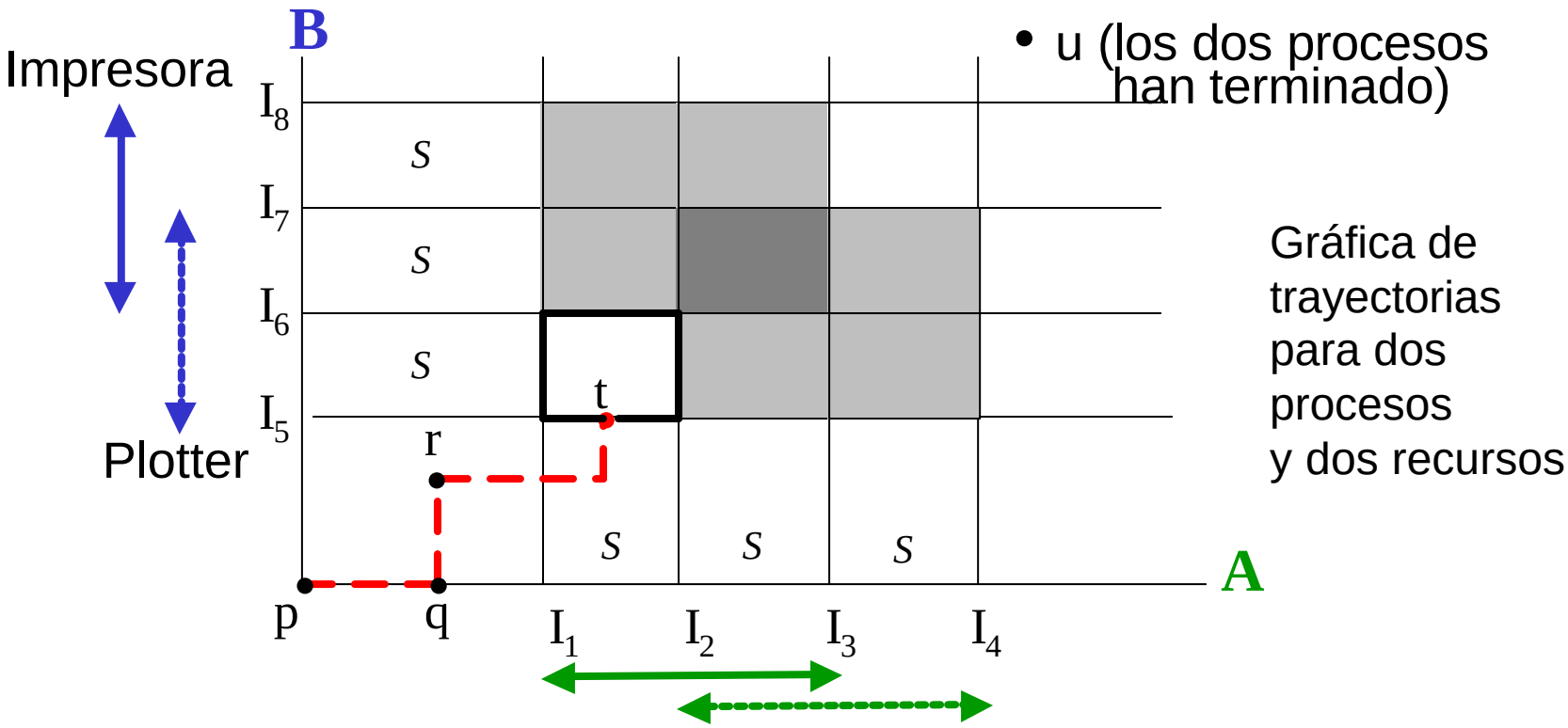
Recursos solicitados

Detección y eliminación (y V)

- Opciones para la eliminación del interbloqueo
  - **Terminación** de **procesos**
    - ☞ Abortar todos los procesos implicados
    - ☞ Ir abortando proceso a proceso hasta eliminar el interbloqueo ☜ ¿cuál?
  - **Apropiación** de **recursos** a los procesos hasta que no se dé el interbloqueo. Consideraciones:
    - ☞ Selección de la víctima
    - ☞ Recuperación: guardar estados anteriores de los procesos para poder reiniciarlos desde un punto anterior de su ejecución (antes de poseer el recurso que se le quita)
    - ☞ Inanición

Evitar

- **Estado seguro**: existe una secuencia de asignaciones de recursos que garantiza que todos los procesos terminan. El caso contrario es un **estado inseguro**



Evitar (y II)

- El estado inseguro no garantiza que se producirá un interbloqueo, sólo que es posible que ocurra
- **Algoritmo del Banquero** ✍ Idea: sólo asigna un recurso si el estado del sistema alcanzado es seguro
  - **Problema**: Debe conocer los recursos que necesita cada proceso a priori
  - **Ejemplo** para un único tipo de recurso:

Libres: 10
Estado Inicial

	Tiene	Máx
A	0	6
B	0	5
C	0	4
D	0	7

Libres: 2
Estado Seguro

	Tiene	Máx
A	1	6
B	1	5
C	2	4
D	4	7

Libres: 1
Estado Inseguro y no bloqueado

	Tiene	Máx
A	1	6
B	2	5
C	2	4
D	4	7

Algoritmo del Banquero
Varios tipos de recursos

E: Vector de recursos existentes (6 3 4 2)

D: Vector de recursos disponibles (1 0 2 0)

A: Matriz de recursos asignados: A_{ij} ⇒ unidades que el proceso número i tiene asignadas del recurso j

R: Matriz de recursos que todavía se necesitan: R_{ij} ⇒ unidades que el proceso número i necesita del recurso j

A =

P	*U*	*T*	*I*	*Pf*
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

R =

P	*U*	*T*	*I*	*Pf*
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

P Proceso
U Unidad de cinta
T Tarjeta gráfica
I Impresora
Pf Perforadora

Algoritmo del Banquero
Varios tipos de recursos (y II)

- Construimos el nuevo estado del sistema suponiendo que se acepta la petición y utilizamos el algoritmo para decidir si dicho estado es seguro o no

1 Buscar una fila de R <= D ¿existe?
si: ir a 2
no: ir a 3

2. Suponer que el proceso de la fila elegida solicita todos los recursos y termina. Marcar al proceso como terminado y añadir sus recursos en A a D. Volver a 1.

3 ¿todos los procesos están marcados? ☞ **estado seguro**
¿existen procesos no marcados que no cumplen la condición de 1? ☞ **estado inseguro**

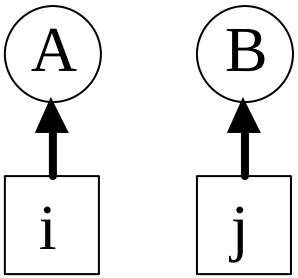
Prevención

- Asegurar que, al menos, una de las condiciones de Coffman no se cumpla
  - Exclusión mutua: usar sistema SPOOL
    - No para todos los tipos de recursos
    - Puede ocurrir interbloqueo
  - Mantener y esperar: evitar que los procesos que tienen asignados recursos esperen por otros
    - Pedir todos los recursos inicialmente
      - ¿Cómo conocen los procesos sus recursos?
      - La asignación de recursos no es óptima
    - Si un proceso solicita un recurso que no está libre: se bloquea y libera todos los recursos asignados. Se activará cuando se le pueda asignar todos los recursos necesarios

Prevención (y II)

- No apropiación: retirarle recursos a los procesos
  - sólo para recursos cuyo estado se puede ser salvado y restaurado posteriormente
- Espera circular:
  - Seguir la regla: un proceso sólo tiene derecho a un único recurso al mismo tiempo ☹ a veces, inaceptable
  - Ordenar los recursos numéricamente. Las peticiones se deben hacer en orden, de acuerdo con el nº de recurso

    Ejemplo: Orden creciente. Un proceso tiene el recurso i, por tanto, sólo puede pedir recursos con número k, k > i



No se puede crear el ciclo, porque sólo uno de ellos podrá solicitar el recurso retenido por el otro proceso

- Si i > j A no puede pedir el recurso j
- Si i < j B no puede pedir el recurso i