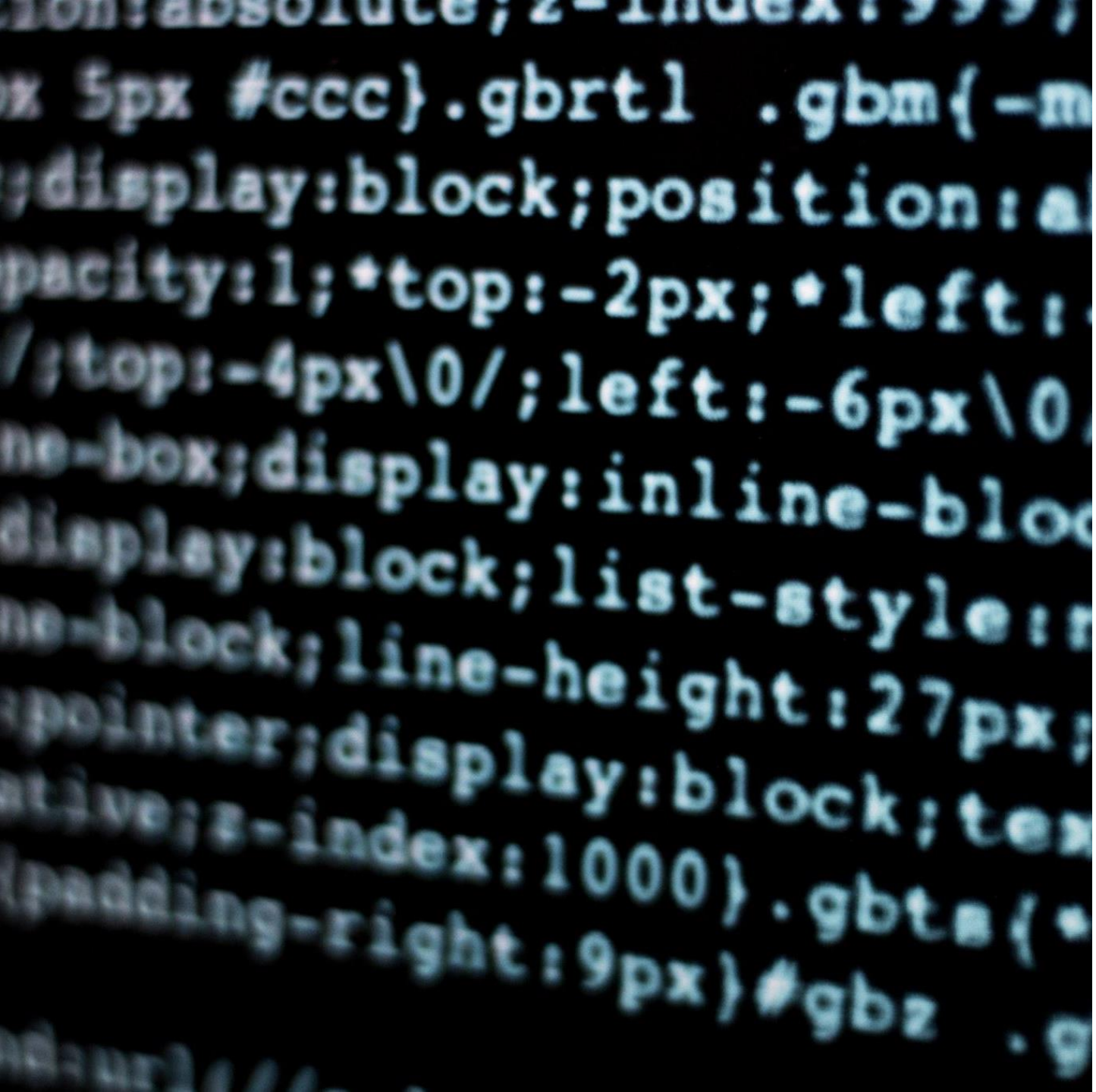
UTN.BA FACULTAD REGIONAL BUENOS AIRES

Arquitectura y Sistemas Operativos

Tecnicatura Universitaria en Programación



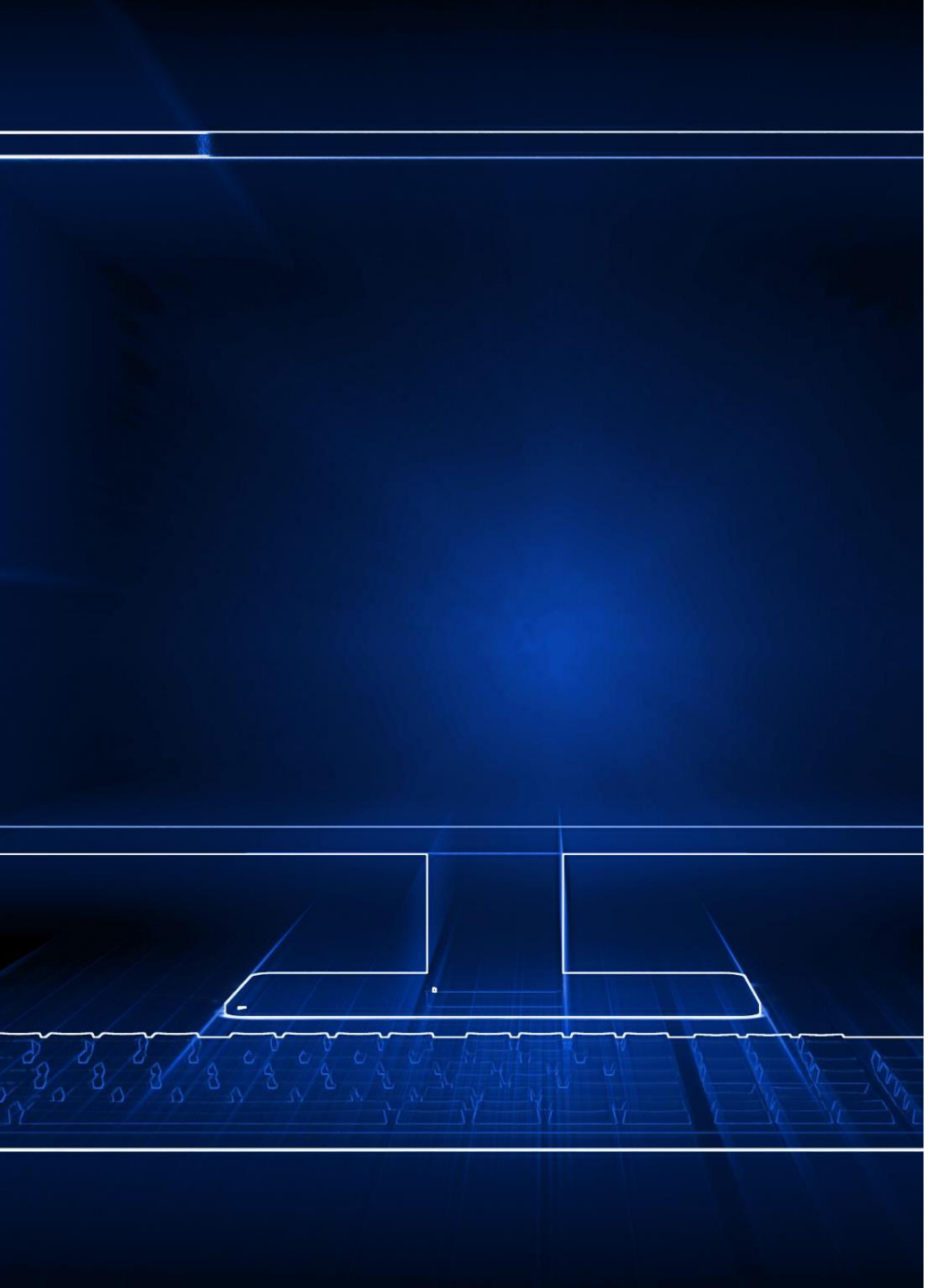
Fundamentos del interbloqueo

Unidad 5

Agenda

1. Definiciones y condiciones para el interbloqueo
2. Prevención del interbloqueo
3. Predicción del interbloqueo
4. Detección del interbloqueo

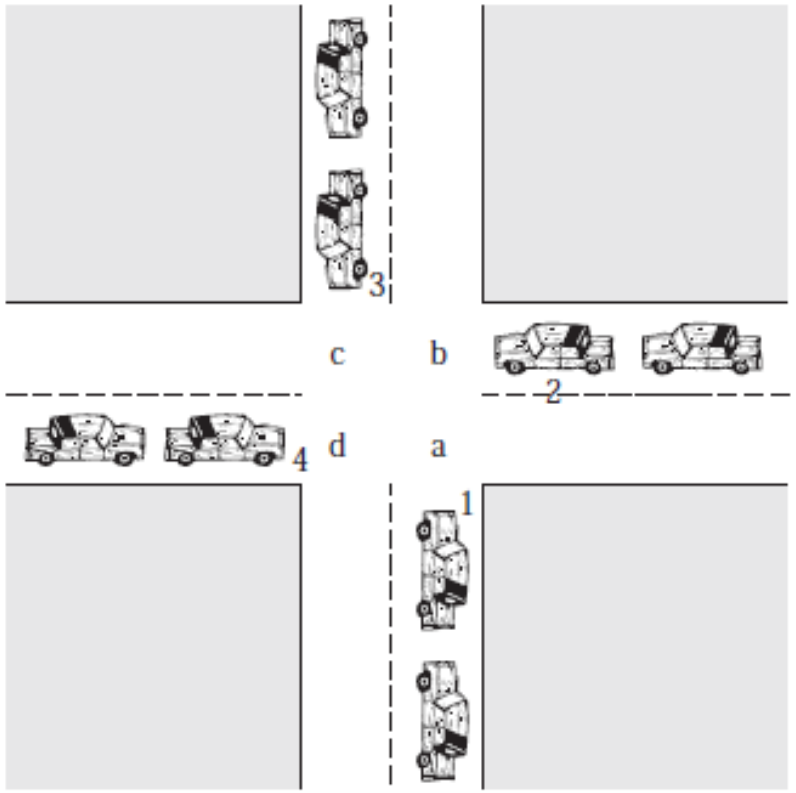
1. Definiciones y condiciones para el interbloqueo



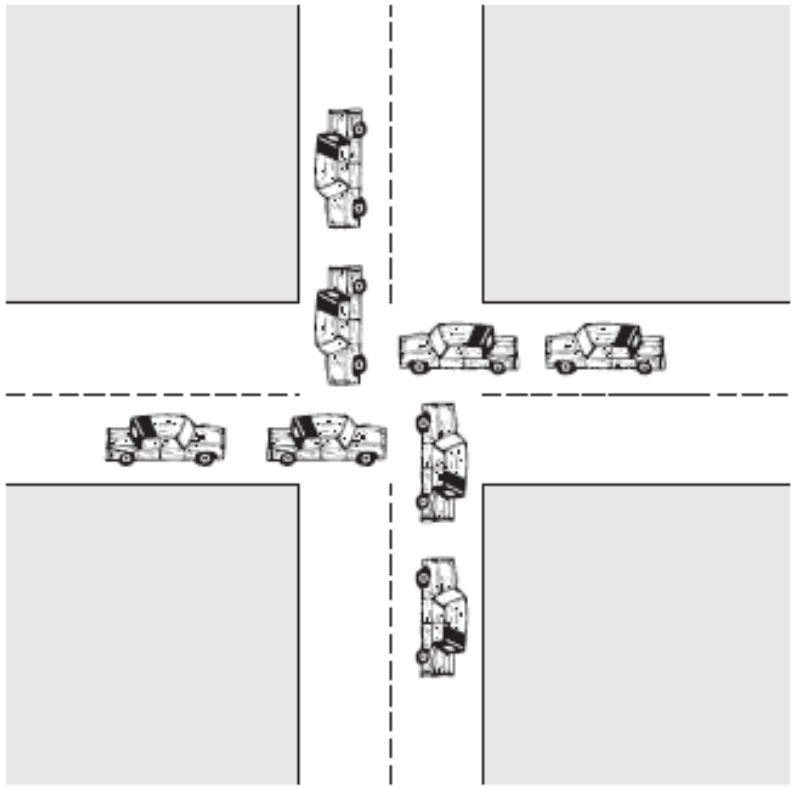
Interbloqueo

También llamado Deadlock, es el **bloqueo permanente** de un conjunto de procesos que compiten por recursos del sistema o se comunican entre sí.

Ejemplo



(a) Posible interbloqueo


(b) Interbloqueo

Condición: todos llegan aproximadamente al mismo tiempo a la intersección

Auto 1 necesita a y b
Auto 2 necesita b y c
Auto 3 necesita c y d
Auto 4 necesita d y a

Opción 1: todos ceden el derecho de paso al de la derecha

Opción 2: ninguno respeta la norma de tránsito y avanzan cuidadosamente accediendo a un cuadrante, pero el segundo ya está ocupado

¿Por qué ocurre el interbloqueo?

La **ocurrencia** o aparición de un deadlock depende tanto de la dinámica de ejecución como de los detalles de implementación de los programas.

Todos los interbloqueos involucran necesidades (recursos) que afectan a 2 o más procesos.

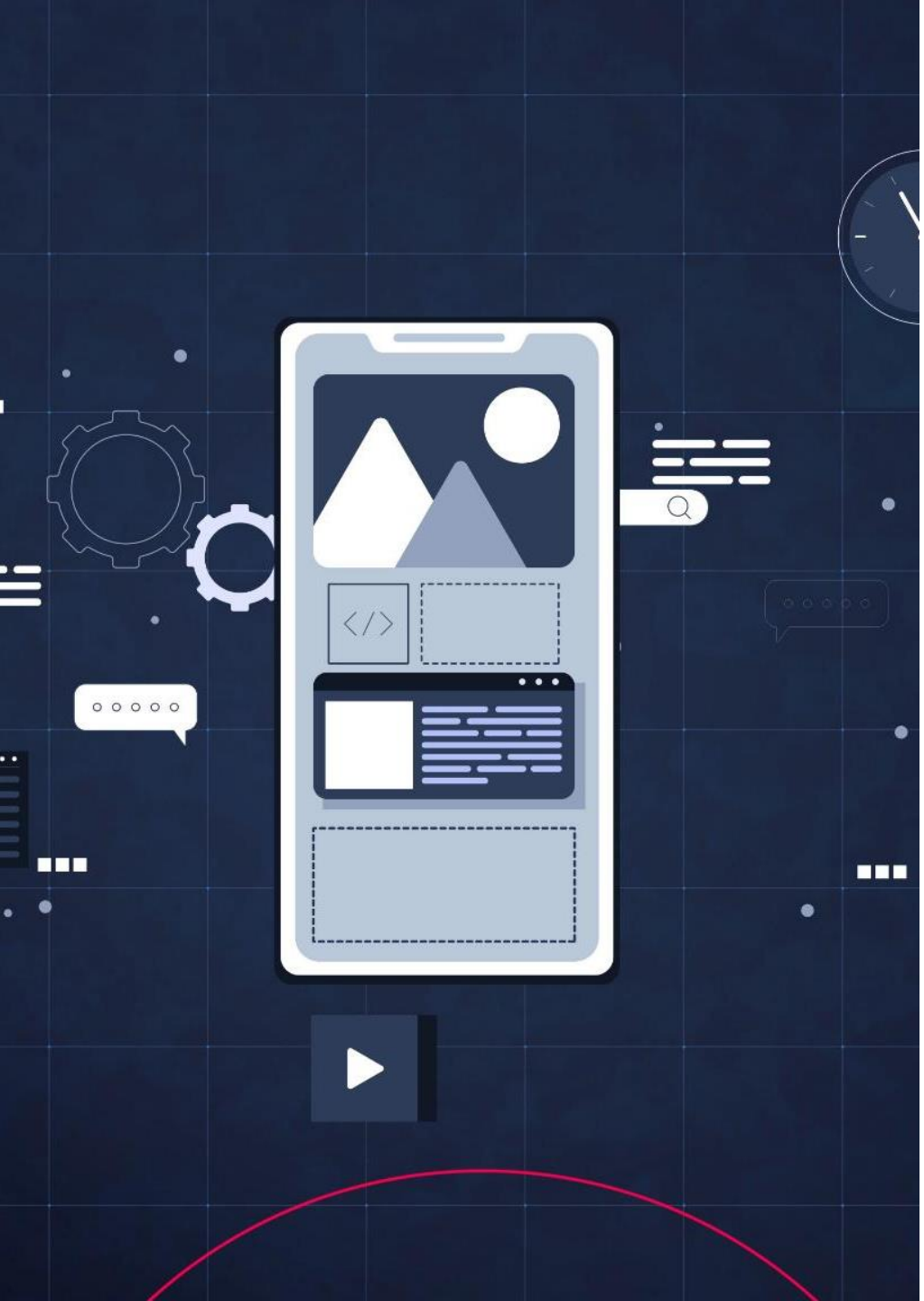
Tipos de recursos

Reutilizables

- Sólo lo puede utilizar **de forma segura** un proceso en cada momento
- No se destruye con su uso
- Son el procesador, disco, memoria, archivos, etc.
- Existe un deadlock si cada proceso retiene un recurso y solicita otro

Consumibles

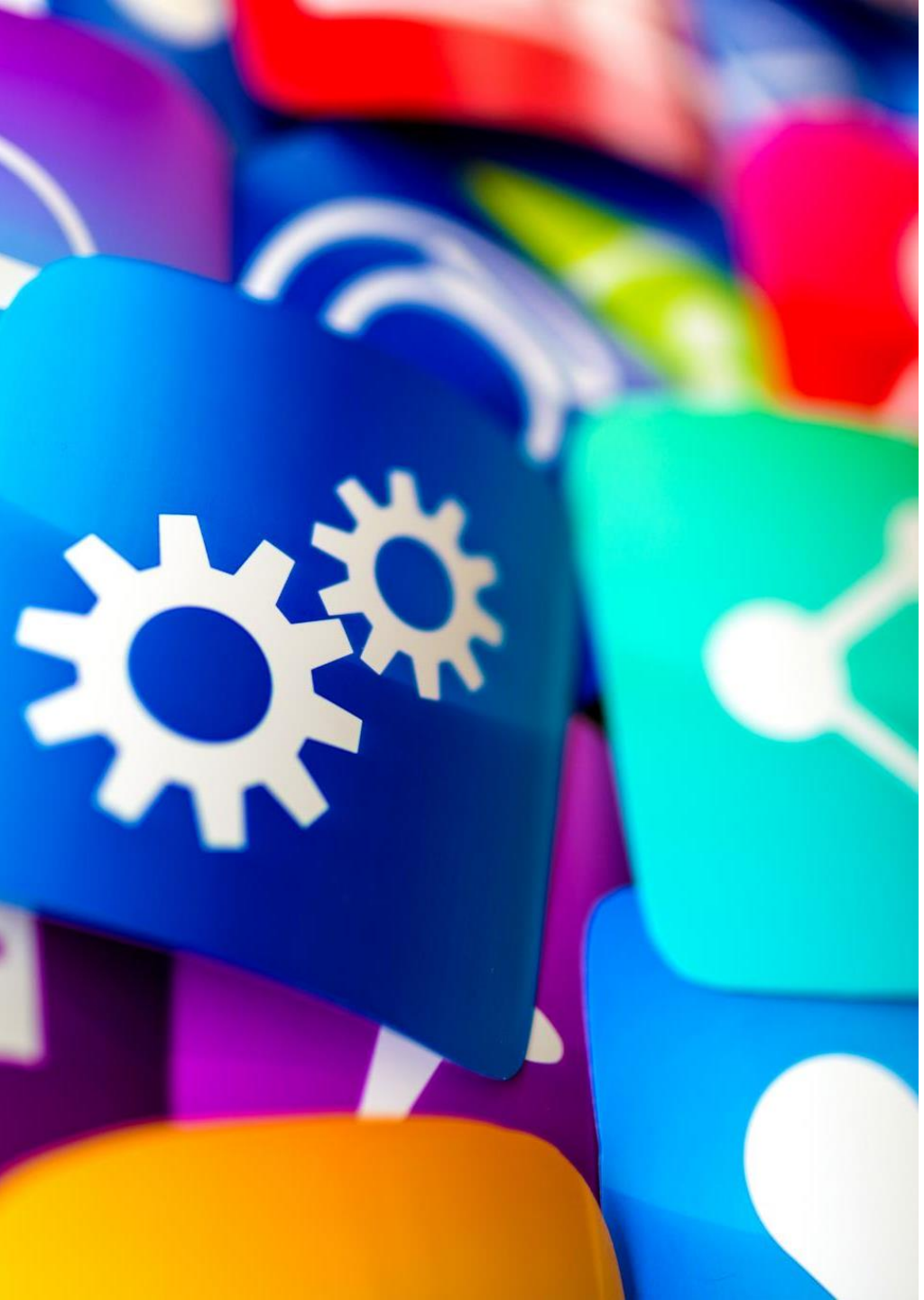
- Puede crearse (producirse) y eliminarse (consumirse) infinitas veces
- Se destruye con su uso
- Son Interrupciones, señales, mensajes, etc.
- Existe deadlock si el proceso se bloquea a la espera de un mensaje.
- Otras causas involucran combinaciones de eventos



Ejemplo recursos reutilizables

Memoria disponible: 200 Kbytes

Proceso A	Proceso B	
solicita 80 Kbytes	solicita 70 Kbytes	Memoria libre 50 Kbytes
.	.	
.	.	
.	.	
solicita otros 60 Kbytes	solicita otros 80 Kbytes	Interbloqueo



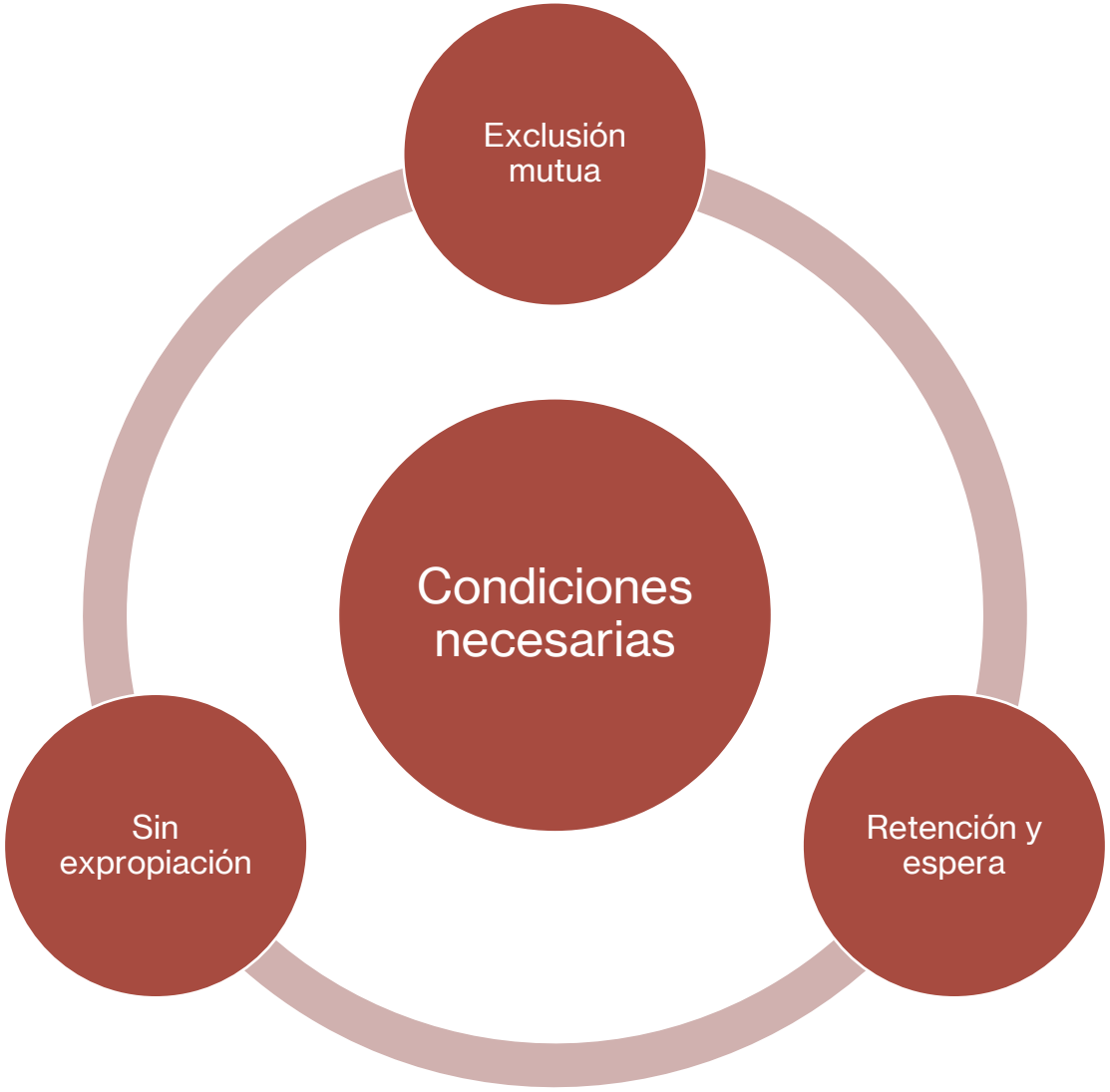
Ejemplo recurso consumible

Proceso A	Proceso B
receive(mensaje_de_B) . send(B, mensaje)	receive(mensaje_de_A) . send(A, mensaje)

Si el receive es bloqueante, puede darse una secuencia de ejecución donde A se bloquee esperando un mensaje de B y a su vez B se bloquee esperando un mensaje de A.

La causa del interbloqueo es por lo general un problema de diseño de las aplicaciones, aunque son muy difíciles de detectar. La dificultad en la detección reside en que la aparición de un interbloqueo muchas veces depende de la secuencia de ejecución de las instrucciones.

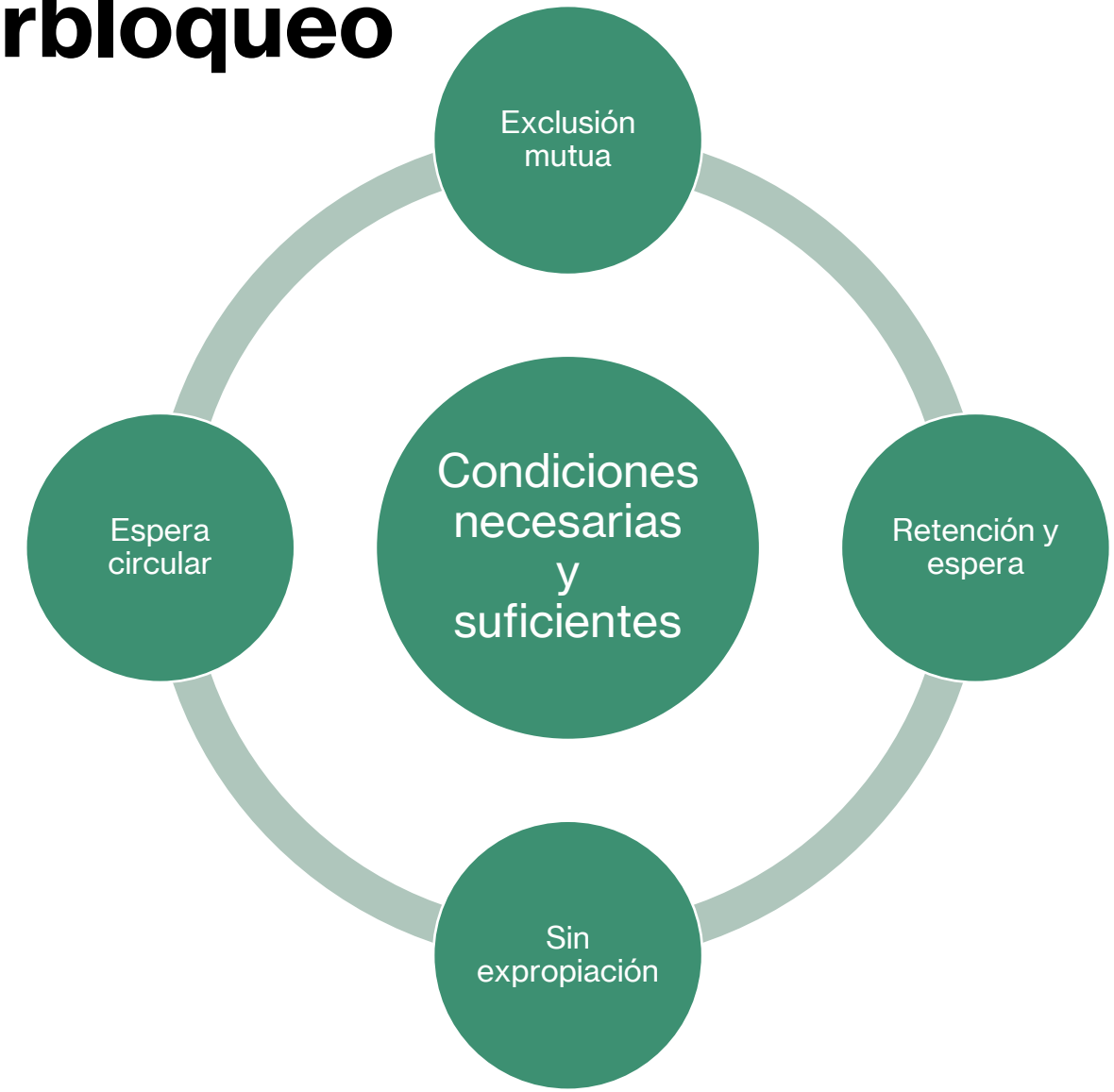
Condiciones necesarias para el interbloqueo



El hecho de que se cumplan estas 3 condiciones (al mismo tiempo), no asegura que exista interbloqueo entre procesos.

Se dice que existe la posibilidad de que ocurra un interbloqueo

Condiciones necesarias y suficientes para el interbloqueo

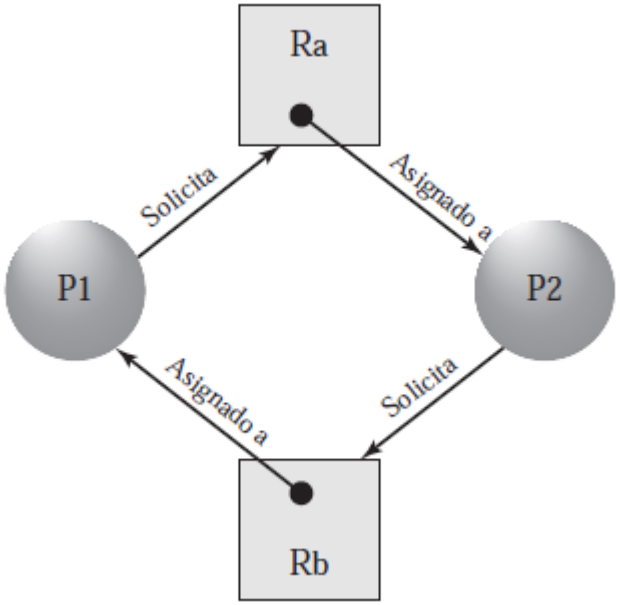


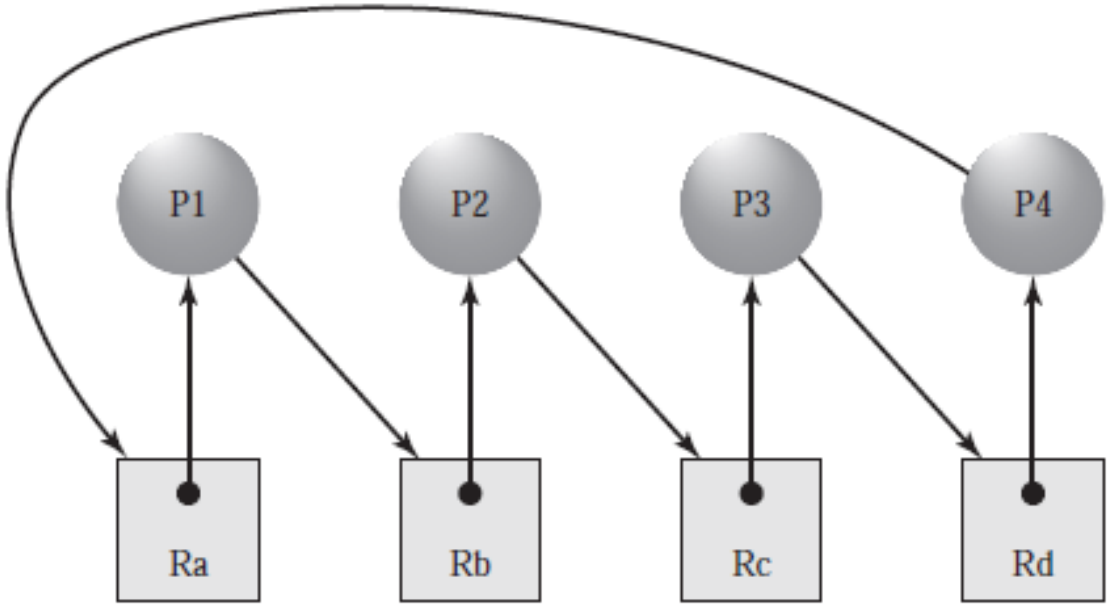
Espera circular

Existe una lista "cerrada" de procesos, de forma tal que cada proceso posee al menos un recurso necesitado por el siguiente proceso de la lista.

Una **espera circular irresoluble** es lo que se denomina interbloqueo

Espera circular





R: Recurso

P: Proceso

2. Prevención del interbloqueo

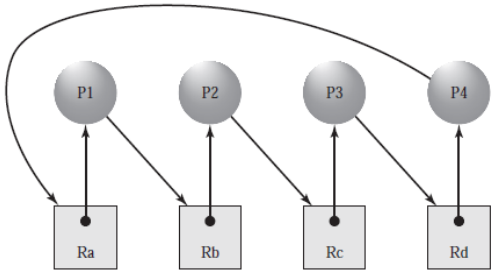
Métodos de prevención

Indirectos: evitar alguna de las 3 condiciones necesarias

- Exclusión mutua: no puede eliminarse. El SO debe proporcionarla
- Retención y espera: solicitar todos los recursos al mismo tiempo
- Sin expropiación: el proceso debe liberar los recursos y solicitarlos otra vez. El SO debe ser capaz de quitar los recursos a un proceso

Directos: evitar la espera circular

- Se define una jerarquía de recursos
- Si un proceso solicita un recurso de tipo A, luego sólo podrá solicitar un recurso de tipo B

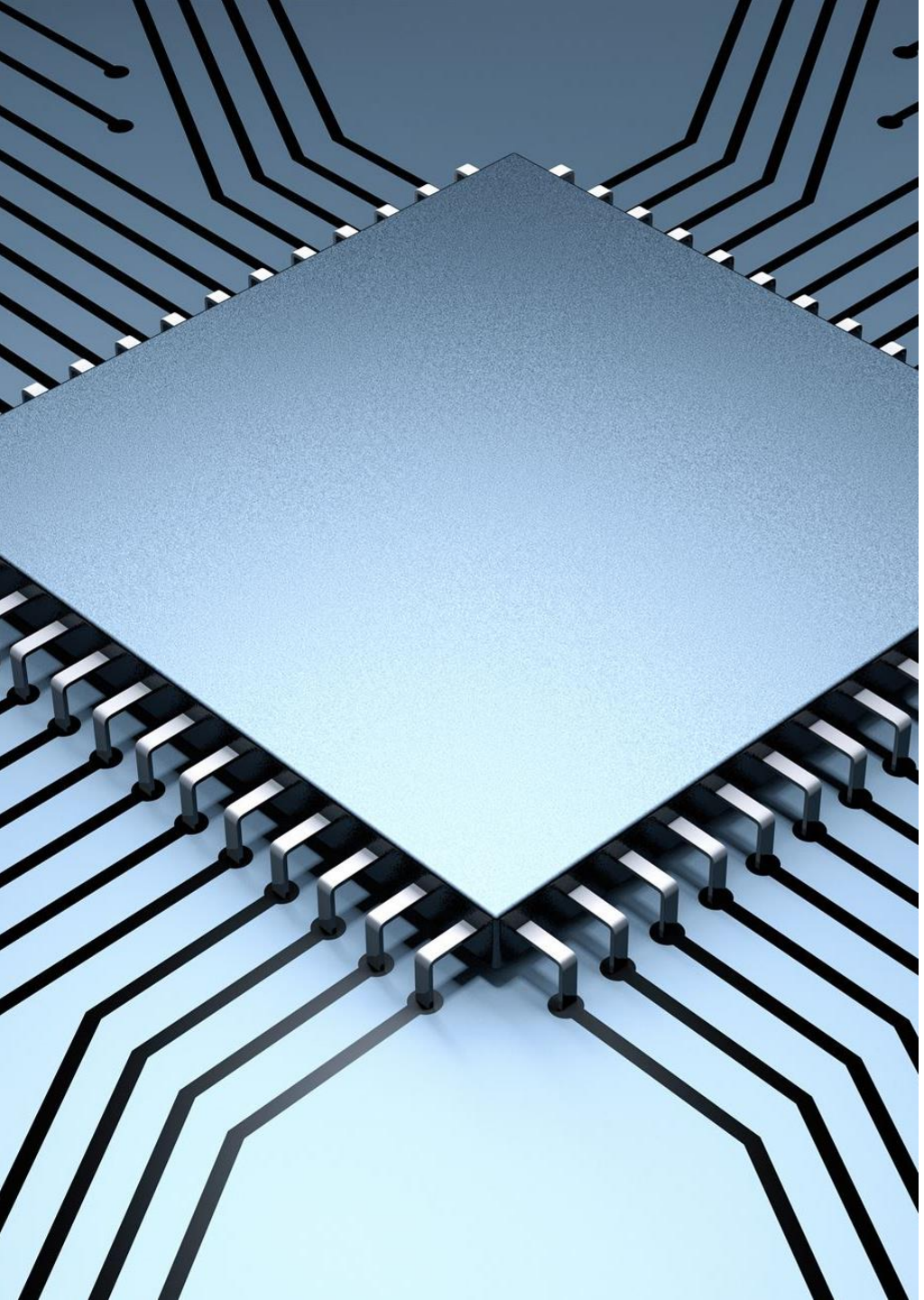


3. Predicción del interbloqueo

Estrategia de predicción

Decidir dinámicamente si la petición de un recurso podrá causar un interbloqueo.

La predicción del interbloqueo requiere el conocimiento de las futuras solicitudes de recursos del proceso.



Acciones (dos posibilidades)

- Denegar el inicio de un proceso si la solicitud de recursos puede conducir a un deadlock

- No asignar un recurso a un proceso si esta asignación puede conducir a un deadlock (algoritmo del banquero).

Denegación del inicio del proceso

- *Rj*: Recursos totales del sistema
- *Dj*: Recursos disponibles del sistema
- *Nij*: necesidades del proceso *i* con respecto al recurso *j*
- *Aij*: asignación actual al proceso *i* con respecto al recurso *j*

1. $R_j = D_j + \sum_{i=1}^{n} A_{ij}$, para todo j	Todos los recursos están disponibles o asignados.
2. $N_{ij} \leq R_j$, para todo i, j	Ningún proceso puede necesitar más de la cantidad total de recursos existentes en el sistema.
3. $A_{ij} \leq N_{ij}$, para todo i, j	Ningún proceso tiene asignados más recursos de cualquier tipo que sus necesidades originales de ese recurso.

Se inicia un nuevo proceso P_{n+1} sólo si:

$$R_j \geq N_{(n+1)j} + \sum_{i=1}^{n} N_{ij} \qquad \text{para todo } j$$

⇨ Esta relación especifica que sólo podrá iniciarse un proceso si se pueden satisfacer las necesidades máximas de todos los procesos actuales más las del nuevo proceso.

Algoritmo del banquero

- El sistema puede estar en 2 estados: seguro y no seguro
- El estado del sistema está representado por la actual asignación de recursos
- Un estado seguro es aquel donde existe al menos una secuencia de asignación que no conduce a un deadlock
- Elementos:
  - Vector de recursos totales (R)
  - Vector de recursos disponibles (D)
  - Matriz de solicitudes (necesidad de recursos de cada proceso) (N o S)
  - Matriz de asignación (A)
- Se debe cumplir $N_{ij} - A_{ij} \leq D_j$, para todo j

Ejemplo 1 (a)

$$N_{ij} - A_{ij} \leq D_j, \text{ para todo } j$$

Estado inicial

R: cant. de recursos totales

R = {9, 3, 6}

D = {0,1,1}

Disponibles: D = R - A

Recursos

Procesos

Necesidad (N)

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2
	16	6	11

Asignación (A)

	R1	R2	R3
	1	0	0
	6	1	2
	2	1	1
	0	0	2
	9	2	5

Necesidad - Asignación (N - A)

R1	R2	R3
2	2	2
0	0	1
1	0	3
4	2	0

Ejemplo 1 (b)

$$N_{ij} - A_{ij} \leq D_j, \text{ para todo } j$$

Estado inicial

R = {9, 3, 6}

D = {0,1,1}

Necesidad

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Asignación

R1	R2	R3
1	0	0
6	1	2
2	1	1
0	0	2

Necesidad - Asignación

R1	R2	R3
2	2	2
0	0	1
1	0	3
4	2	0

Le asigna el recurso R3 que necesita P2 para que este finalice y libere los recursos

Ejemplo 1 (c)

Finalizó P2 y liberó los recursos que se asignan a P1

R = {9, 3, 6}

D = {6,2,3}

Necesidad

	R1	R2	R3
P1	3	2	2
P2	0	0	0
P3	3	1	4
P4	4	2	2

Asignación

R1	R2	R3
1	0	0
0	0	0
2	1	1
0	0	2

Necesidad - Asignación

R1	R2	R3
2	2	2
0	0	0
1	0	3
4	2	0

Le asigna los recursos que necesita P1 para que este finalice y los libere.

Ejemplo 1 (d)

P3 ejecuta hasta completarse

R = {9, 3, 6}

D = {7,2,3}

Necesidad

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	3	1	4
P4	4	2	2

Asignación

R1	R2	R3
0	0	0
0	0	0
2	1	1
0	0	2

Necesidad - Asignación

R1	R2	R3
0	0	0
0	0	0
1	0	3
4	2	0

Aquí tiene la alternativa de elegir primero P3 y luego P4, o viceversa.

Ejemplo 1 (e)

Luego de finalizar P3

R = {9, 3, 6}

D = {9,3,4}

Necesidad

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	2

Asignación

R1	R2	R3
0	0	0
0	0	0
0	0	0
0	0	2

Necesidad - Asignación

R1	R2	R3
0	0	0
0	0	0
0	0	0
4	2	0

Ejemplo 2 (a)

Estado inicial

R = {9, 3, 6}

D = {1,1,2}

Necesidad

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Asignación

R1	R2	R3
1	0	0
5	1	1
2	1	1
0	0	2

Necesidad - Asignación

R1	R2	R3
2	2	2
1	0	2
1	0	3
4	2	0

Ejemplo 2 (b)

P1 solicita una unidad de R1 y de R3

Estado inicial

R = {9, 3, 6}

D = {0,1,1}

Necesidad

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Asignación

R1	R2	R3
2	0	1
5	1	1
2	1	1
0	0	2

Necesidad - Asignación

R1	R2	R3
1	2	1
1	0	2
1	0	3
4	2	0

Estado inseguro: Con los recursos disponibles no puede asignarlos a ningún proceso para cubrir completamente sus necesidades

¡Atención!

Un estado inseguro, no garantiza que exista interbloqueo, sólo indica que existe la posibilidad de que ocurra un interbloqueo.

Todo depende de la sucesión de eventos al ejecutar un proceso

4. Detección del interbloqueo

Algoritmo de detección del interbloqueo

La comprobación de si hay interbloqueo se puede hacer una vez por cada petición de recurso o, con menos frecuencia, dependiendo de la probabilidad de que ocurra un interbloqueo.

Comprobación por cada petición de recurso:

- Ventaja: detección temprana y el algoritmo es relativamente sencillo debido a que está basado en cambios graduales del estado del sistema.
- Desventaja: estas comprobaciones frecuentes consumen un considerable tiempo del procesador.

Ejemplo

Estado inicial

R = {2, 1, 1, 2, 1}

D = {0, 0, 0, 0, 1}

Solicitudes (S)

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Asignación (A)

R1	R2	R3	R4	R5
1	0	1	1	0
1	1	0	0	0
0	0	0	1	0
0	0	0	0	0

Algoritmo de detección

1. En la matriz A se marcan todas las filas que se encuentren completamente en 0 (cero).
2. Se inicia un vector temporal T asignándole el valor del vector de D.
3. De las filas sin marcar en A, se busca un proceso cuyas solicitudes de recursos sean menores o iguales a T. Si no se encuentra ningún proceso que cumplan tal condición, el algoritmo finaliza.
4. Si se encuentra una fila que cumpla la condición del punto 3, se marca el proceso, se suma la fila correspondiente al proceso en la matriz A al vector T, y vuelve al 3er paso.

Aplicación del algoritmo de detección en el ejemplo

1. Marca en la matriz A, a P4 porque no tiene recursos asignados
2. Establece T = {0, 0, 0, 0, 1}
3. Recorre S buscando una fila que sea menor o igual a T. El proceso P3 satisface dicha condición y lo marca.
4. Actualiza T, sumando el valor de la fila de Asignaciones correspondiente al índice de la fila encontrada en S. En este caso D = {0, 0, 0, 0, 1} + {0, 0, 0, 1, 0} => D = {0, 0, 0, 1, 1}
5. Ya ningún proceso cumple con la condición del punto 3, con lo cual el algoritmo concluye sin marcar P1 y P2, indicando que estos procesos está<u>n en un interbloqueo</u>.

R = {2, 1, 1, 2, 1}

D = {0, 0, 0, 0, 1}

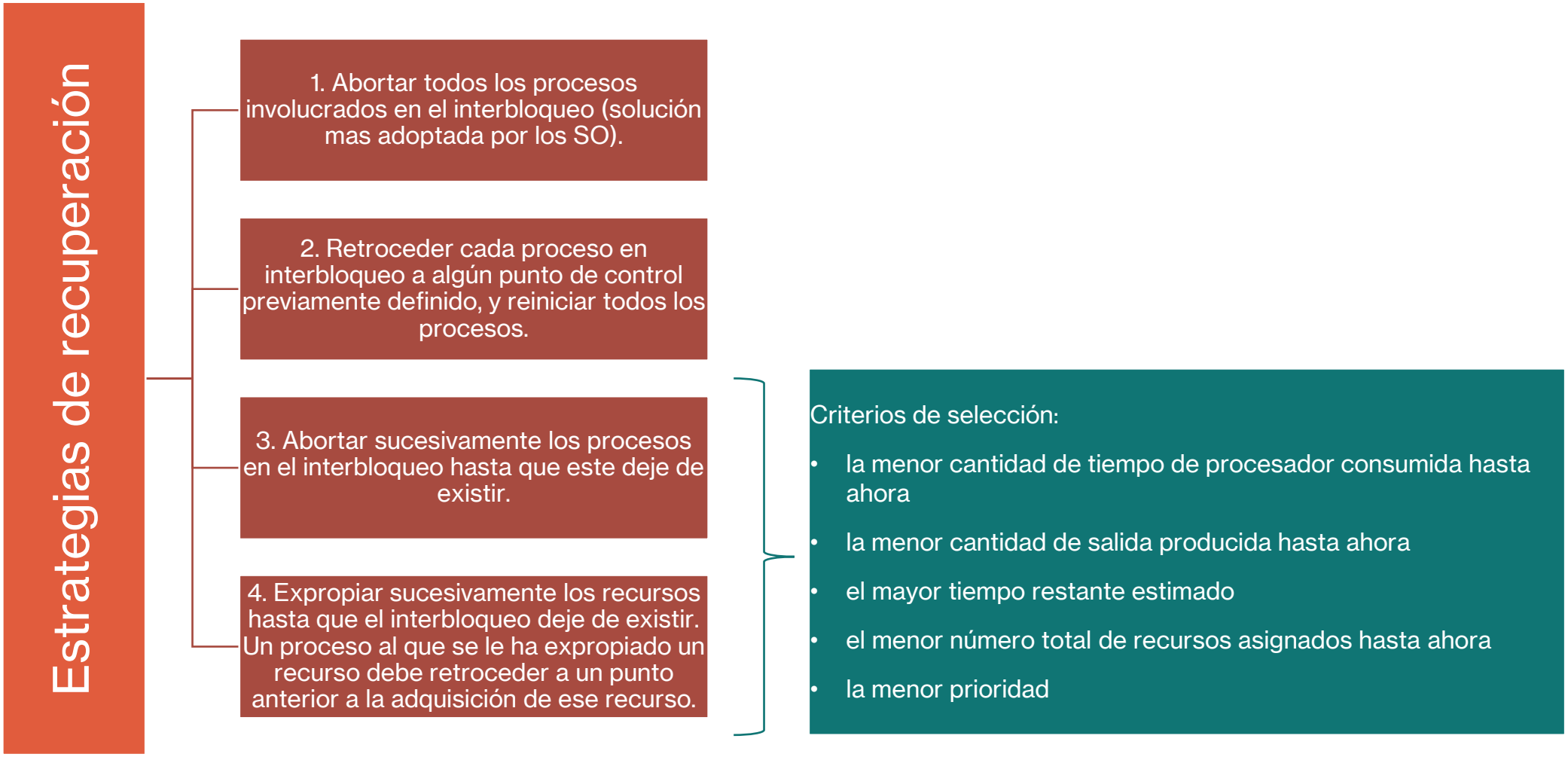
Solicitudes (S)

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Asignación (A)

R1	R2	R3	R4	R5
1	0	1	1	0
1	1	0	0	0
0	0	0	1	0
0	0	0	0	0

Recuperación del interbloqueo



UTN.BA FACULTAD REGIONAL BUENOS AIRES

Muchas gracias