Introducción a los Sistemas Operativos

Interbloqueos

Armando Arce, arce@itcr.ac.cr

Tecnológico de Costa Rica

Interbloqueos

En un entorno de multiprogramación, varios procesos pueden competir por un número finito de recursos.

- Un proceso solicita recursos y, si los recursos no están disponibles en este momento, el proceso pasa al estado de espera.
- Es posible que, algunas veces, un proceso esperando pueda nunca cambiar estado, porque los recursos que está solicitado estén ocupados por otros procesos que a su vez estén esperando otros recursos.
- Cuando se produce una situación como esta, se dice que ocurrió un interbloqueo.

Modelo de sistemas

Un proceso debe solicitar cada recurso antes de utilizarlo y debe liberarlo después de usado.

- Un proceso puede solicitar tantos recursos como necesite para llevar a cabo las tareas que tenga asignadas.
- El número de recursos solicitados no puede exceder el total de recursos disponibles en el sistema.

Modelo de sistemas

Una tabla del sistema registra si cada recurso está libre o ha sido asignado; para cada recurso asignado, la tabla también registra el proceso al que está asignado actualmente.

- Si un proceso solicita un recurso que en ese momento está asignado a otro proceso, puede añadirse a la cola de procesos en espera para ese recurso.
- En un interbloqueo, los procesos nunca terminan de ejecutarse y los recursos del sistema están ocupados, lo que impide que se inicien otros trabajos.

Condiciones necesarias

La situación de interbloqueo puede surgir si se dan simultáneamente las cuatro condiciones siguientes en un sistema:

1. Exclusión mutua: al menos un recurso debe estar en modo no compartido; es decir, sólo un proceso puede usarlo cada vez. Si otro proceso solicita el recurso, el proceso solicitante tendrá que esperar hasta que el recurso sea liberado.
2. Retención y espera: un proceso debe estar reteniendo al menos un recurso y esperando para adquirir otros recursos adicionales que actualmente estén retenidos por otros procesos.

Condiciones necesarias

3. Sin desalojo: los recursos no puede ser desalojados; es decir, un recurso sólo puede ser liberado voluntariamente por el proceso que le retiene, después de que dicho proceso haya completado su tarea.
4. Espera circular: debe existir un conjunto de procesos en espera, tal que uno de los procesos esté esperando a un recurso requerido por otro, y el otro esté esperando un recurso requerido por alguno mas.

Grafo de asignación de recursos

Los interbloqueos pueden definirse de forma más precisa mediante un grafo dirigido, que se llama grafo de asignación de recursos del sistema.

- Este grafo consta de un conjunto de vértices y de un conjunto de aristas.
- El conjunto de vértices se divide en dos tipos diferentes, el conjunto formado por todos los procesos activos del sistema, y el conjunto formado por todos los recursos del sistema.

Grafo de asignación de recursos

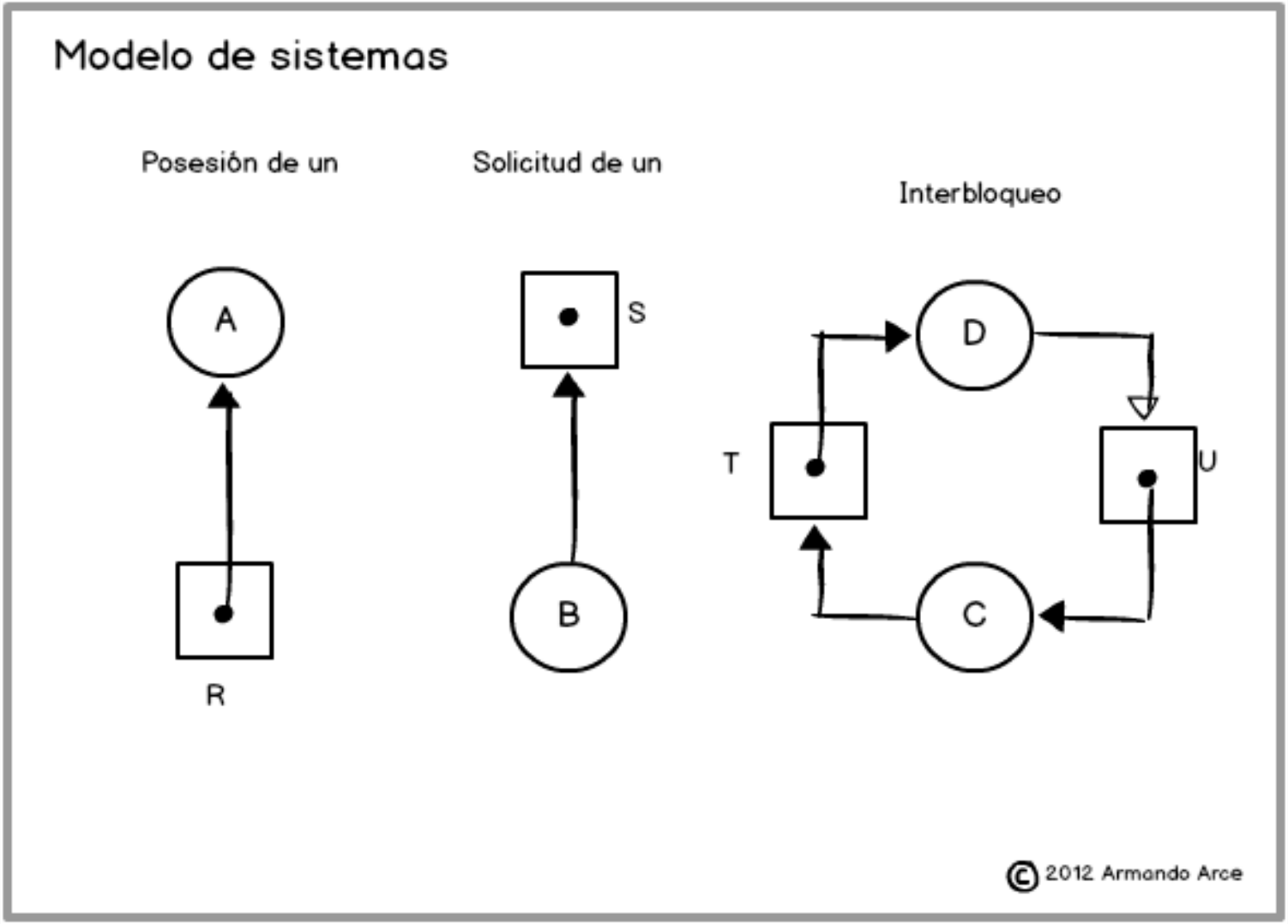

Figure 1:

Grafo de asignación de recursos

Una arista dirigida desde un proceso a un recurso significa que el proceso ha solicitado una instancia del recurso y que actualmente está esperando dicho recurso.

- Una arista dirigida desde el recurso al proceso quiere decir que se ha solicitado una instancia de ese recurso al proceso.
- La arista dirigida desde el proceso al recurso se denomina arista de solicitud, mientras que la lista dirigida del recurso al proceso se denomina arista de asignación.

Grafo de asignación de recursos

Dada la definición de un grafo de asignación de recursos, se puede demostrar que, si el grafo no tienen ningún ciclo, entonces ningún proceso del sistema está en interbloqueo. Si el grafo contiene un ciclo, entonces puede existir un interbloqueo.

Grafo de asignación de recursos

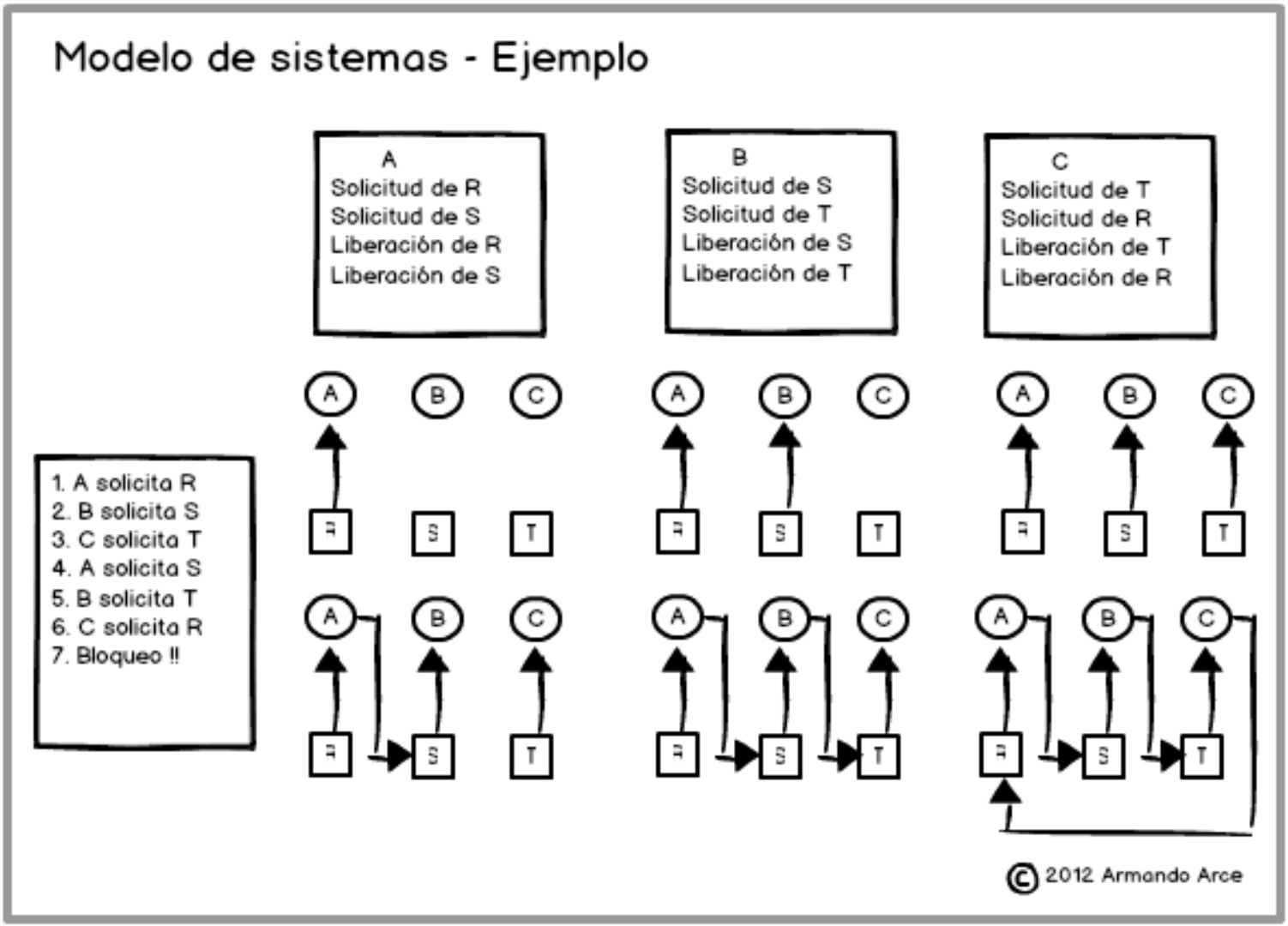

Figure 2:

Métodos para tratar interbloqueos

Para garantizar que nunca se produzcan interbloqueos, el sistema puede emplear un esquema de prevención de interbloqueos o un esquema de evasión de interbloqueos.

- La prevención de interbloqueos proporciona un conjunto de métodos para asegurar que al menos una de las condiciones necesarias no pueda cumplirse.
- Éstos métodos evitan los interbloqueos restringiendo el modo en que pueden realizarse las solicitudes.

Métodos para tratar interbloqueos

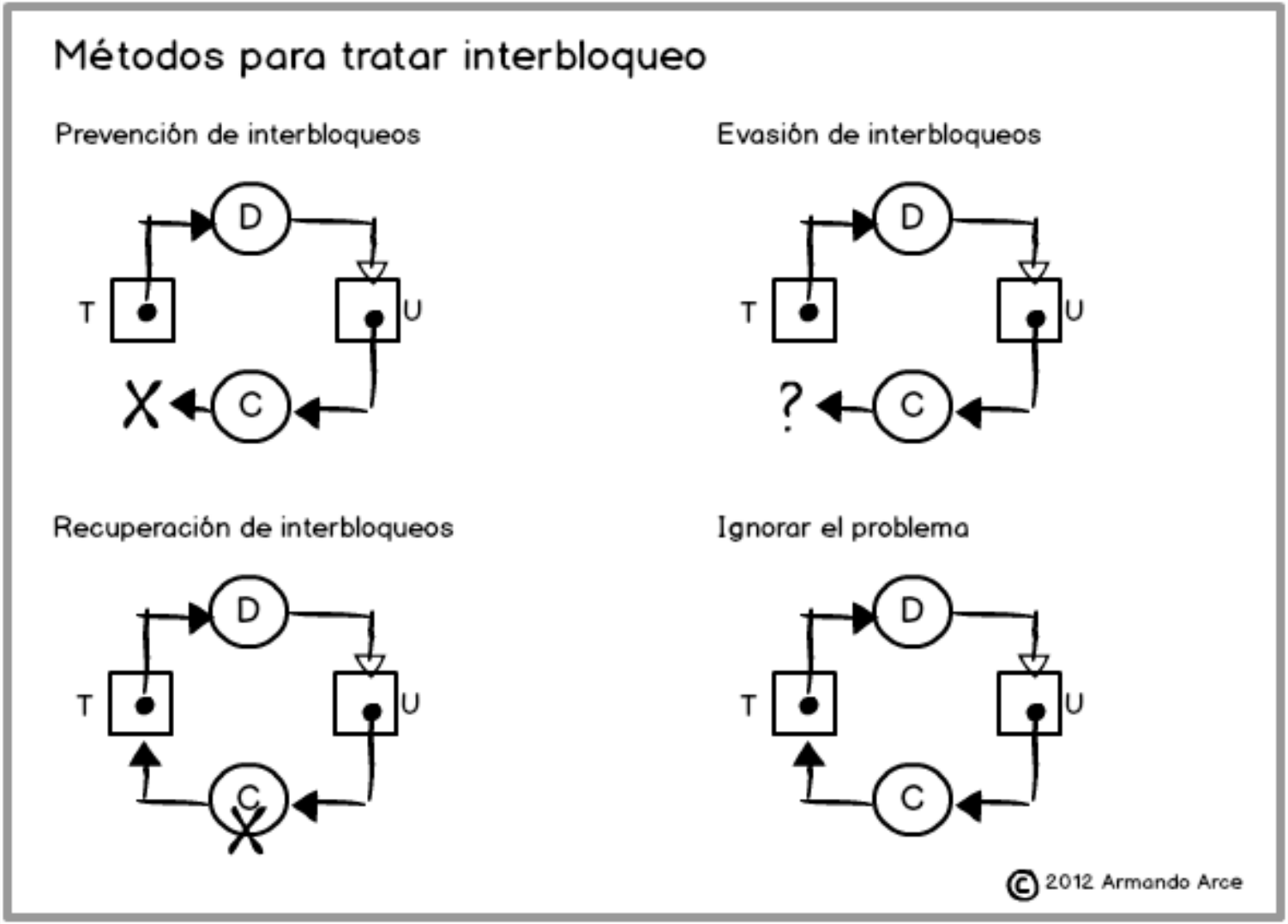

Figure 3:

Métodos para tratar interbloqueos

La evasión de interbloqueos requiere que se proporcione de antemano al sistema operativo información adicional sobre qué recursos solicitará y utilizará un proceso durante su tiempo de vida.

- Con estos datos adicionales, se puede decir, para cada solicitud, si el proceso tiene que esperar o no.
- Para decidir si la solicitud actual puede satisfacerse o debe retardarse, el sistema necesita considerar que recursos hay disponibles en ese momento, qué recursos están asignados a cada proceso y las futuras solicitudes y liberaciones de cada proceso.

Métodos para tratar interbloqueos

Si un sistema no emplea un algoritmo de prevención de interbloqueos ni un algoritmo de evasión, entonces puede producirse una situación de interbloqueo.

- En este tipo de entorno, el sistema puede proporcionar un algoritmo que examine el estado del mismo para determinar si se ha producido en interbloqueo (detección) y otro algoritmo para recuperarse (recuperación) de dicho interbloqueo, si realmente se ha producido.

Métodos para tratar interbloqueos

Si un sistema no garantiza que nunca se producirá un interbloqueo, y no proporciona un mecanismo para la detección y recuperación de interbloqueos, entonces puede darse la situación de que el sistema esté en estado de interbloqueo y no tenga forma ni siquiera de saberlo.

Métodos para tratar interbloqueos

En este caso, el interbloqueo no detectado dará lugar a un deterioro del rendimiento del sistema, ya que habrá recursos retenidos por procesos que no pueden ejecutarse y, a medida que se realicen nuevas solicitudes de recursos, cada vez más procesos entrarán en estado de interbloqueo.

- Finalmente, el sistema dejará de funcionar y tendrá que reiniciarse manualmente.

Métodos para tratar interbloqueos

Aunque este método de ignorar el problema (algoritmo del avestruz) no parece un sistema viable para abordar el interbloqueo, se usa en la mayoría de los sistemas operativos.

- En muchos sistemas, los interbloqueos se producen de forma bastante infrecuente, por ejemplo una vez al año; por tanto, este método es más barato que los métodos de prevención, de evasión o de detección y recuperación, que deben utilizarse constantemente.

Prevención de interbloqueos

Para que se produzca un interbloqueo deben cumplirse las cuatro condiciones necesarias. Asegurando que una de las cuatro condiciones no se cumpla, podemos prevenir la aparición de interbloqueos.

Prevención de interbloqueos

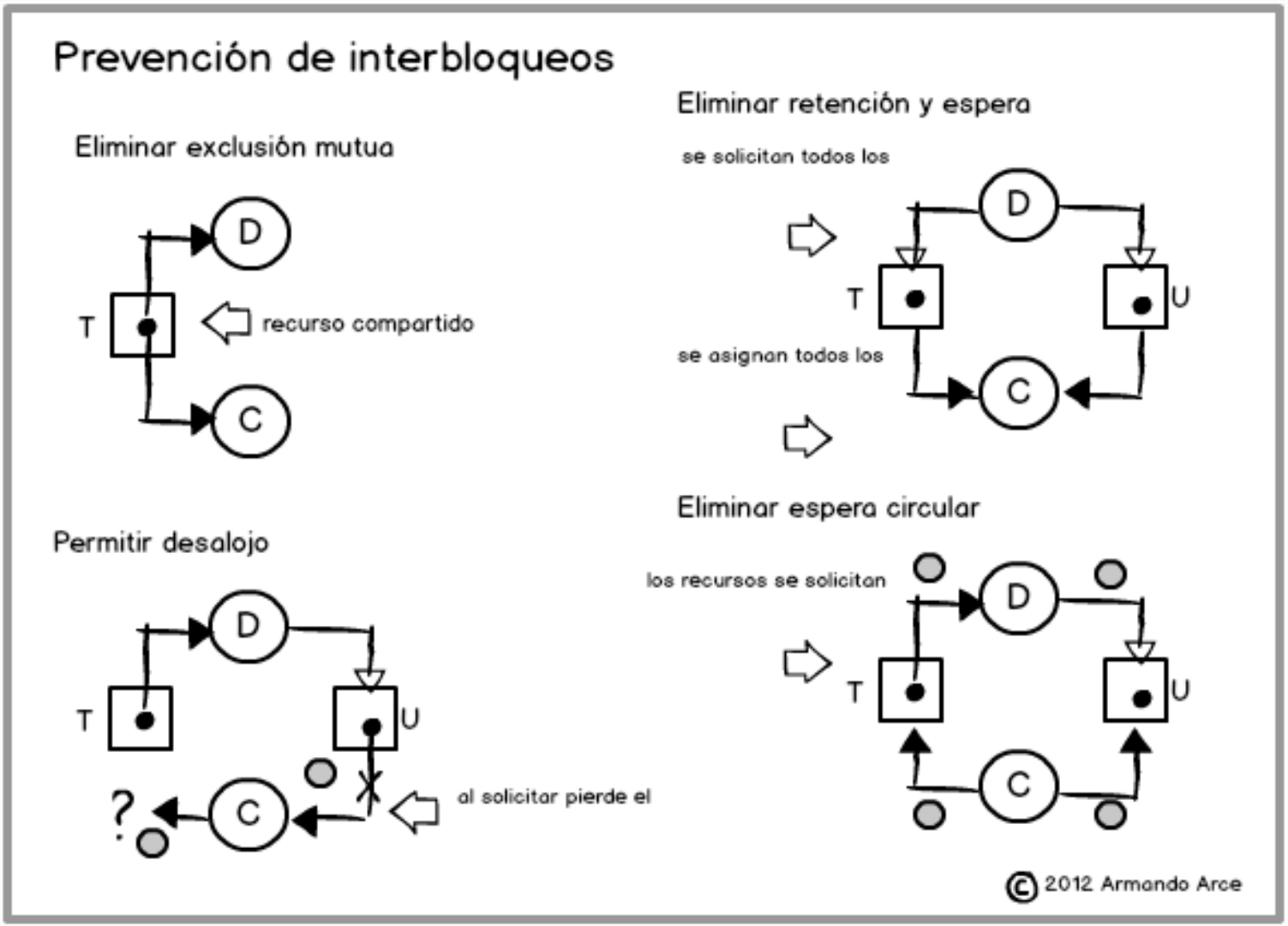

Figure 4:

Eliminar exclusión mutua

La condición de exclusión mutua se aplica a los recursos que no pueden ser compartidos.

- Por el contrario, los recursos que sí pueden compartirse no requieren acceso mutuamente excluyente y, por tanto no pueden verse implicados en un interbloqueo.
- La ventaja aquí es que un proceso no necesita esperar nunca para acceder a un recurso compartible. Sin embargo, algunos recursos son intrínsecamente no compartibles.

Eliminar retención y espera

Se debe garantizar que, cuando un proceso solicite un recurso, el proceso no esté reteniendo ningún otro recurso.

- Un posible protocolo consiste en exigir que cada proceso solicite todos sus recursos (y que los recursos se le asignen) antes de comenzar su ejecución.
- La desventaja es que la tasa de utilización de recursos puede ser baja, dado que los recursos pueden asignarse pero no utilizarse durante un período largo de tiempo.

Eliminar retención y espera

- Una segunda desventaja es que puede producirse el fenómeno de inanición ya que un proceso que solicite varios recursos muy solicitados puede esperar de forma indefinida, debido a que al menos uno de los recursos que necesita está siempre asignado a otro proceso.

Permitir el desalojo

Para impedir el desalojo, se puede usar el protocolo siguiente.

- Si un proceso está reteniendo varios recursos y solicita otro recurso que no se le puede asignar de forma inmediata, entonces todos los recursos actualmente retenidos se desalojan.
- El proceso sólo se reiniciará cuando pueda recuperar sus antiguos recursos, junto con los nuevos que está solicitando.
- A menudo, este protocolo se aplica a tipos de recursos cuyo estado puede guardarse y restaurarse luego, como registros del CPU y el espacio en memoria. Sin embargo, generalmente este método no se puede aplicar a recursos como impresoras y unidades de cinta.

Eliminar espera circular

Una forma de garantizar que la condición de espera circular nunca se produzca es imponer una ordenación total de todos los tipos de recursos y requerir que cada proceso solicite sus recursos en orden creciente de numeración.

- Si un proceso solicita una instancia de un tipo de recurso menor al que ya tiene, debe liberar primero dichos recursos.

Evasión de interbloqueos

Un método alternativo para evitar los interbloqueos consiste en requerir información adicional sobre cómo van a ser utilizados los recursos.

- Conociendo exactamente la secuencia completa de solicitudes y liberaciones de cada proceso, el sistema puede decidir, para cada solicitud, si el proceso debe esperar o non con el fin de evitar u posible interbloqueo futuro.
- Cada solicitud requiere que, para tomar la correspondiente decisión, el sistema considere los recursos actualmente disponibles, los recurso actualmente asignados a cada proceso y las solicitudes y liberaciones futuras de cada proceso.

Evasión de interbloqueos

Solo un tipo de recurso

Proc	Tiene	Máximo
A	2	6
B	1	5
C	2	4
D	5	7

Cada proceso declara la cantidad máxima de recursos que puede necesitar

Varias instancias de cada recurso

		A	B	C
P0	Asig	0	1	0
	Solic	0	0	0
P1	Asig	2	0	0
	Solic	2	0	2
P2	Asig	3	0	3
	Solic	0	0	0
P3	Asig	2	1	1
	Solic	1	0	0
P4	Asig	0	0	2
	Solic	0	0	2
	Disp	0	0	0

El estado de asignación está definido por el número de recursos disponibles y asignados y la demanda máxima de los procesos.

Figure 5:

Estado seguro

Un estado que seguro si el sistema puede asignar recursos a cada proceso, hasta su máximo, en determinado orden sin que eso produzca un interbloqueo. Un sistema está en estado seguro sólo si existe lo que se denomina una secuencia segura.

Estado seguro

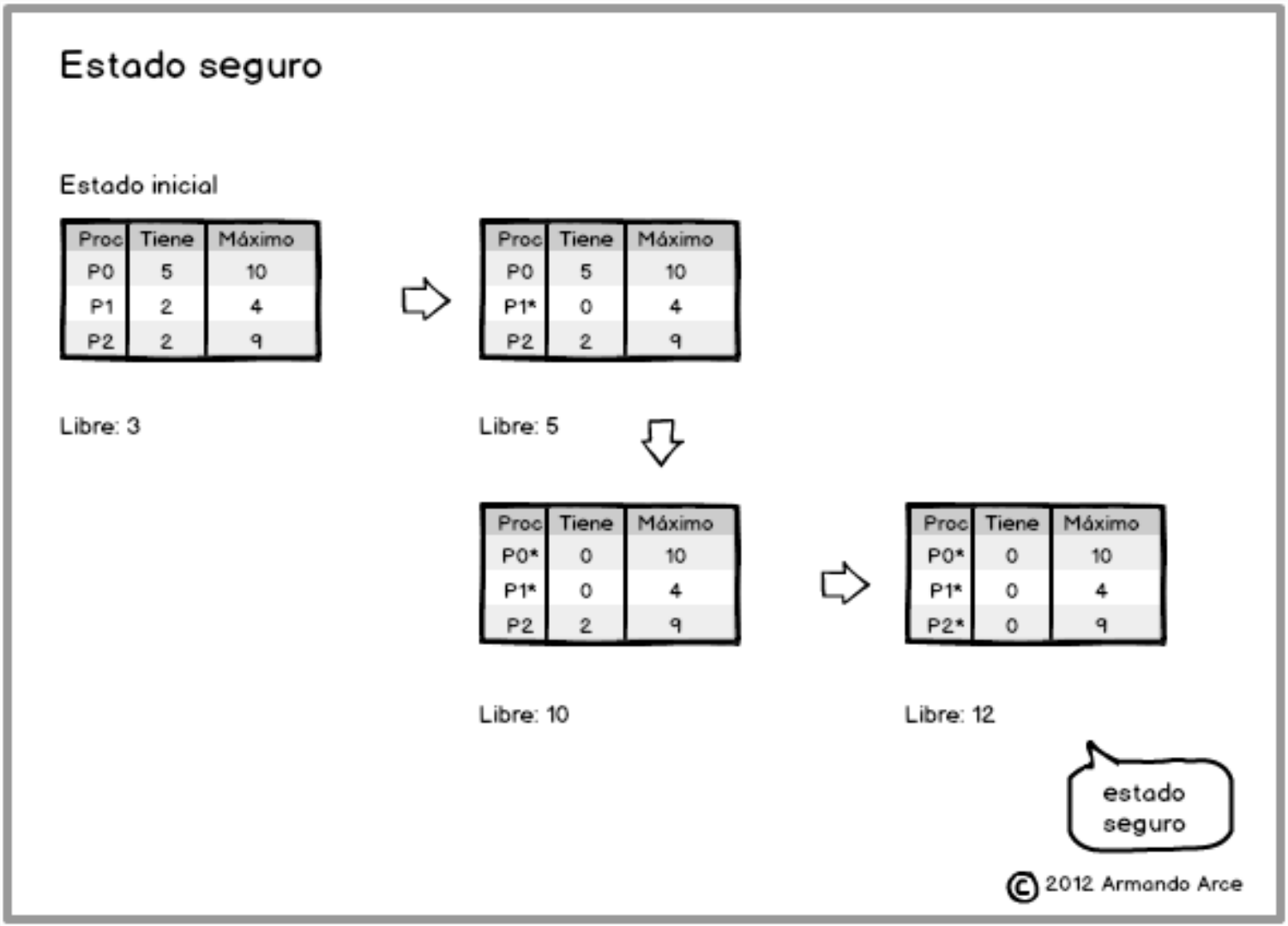

Figure 6:

Estado seguro

Un estado seguro implica que no puede producirse interbloqueo. A la inversa, todo estado de interbloqueo es inseguro.

- Sin embargo, no todos los estados inseguros representan un interbloqueo.
- Un estado inseguro puede llevar a que aparezca un interbloqueo.
- Siempre y cuando el estado sea seguro, el sistema operativo podrá evitar los estados inseguros, y de interbloqueo.
- En un estado inseguro, el sistema operativo no puede impedir que los procesos soliciten recursos de tal modo que se produzca un interbloqueo: el comportamiento de los procesos es el que controla los estados inseguros.

Estado seguro

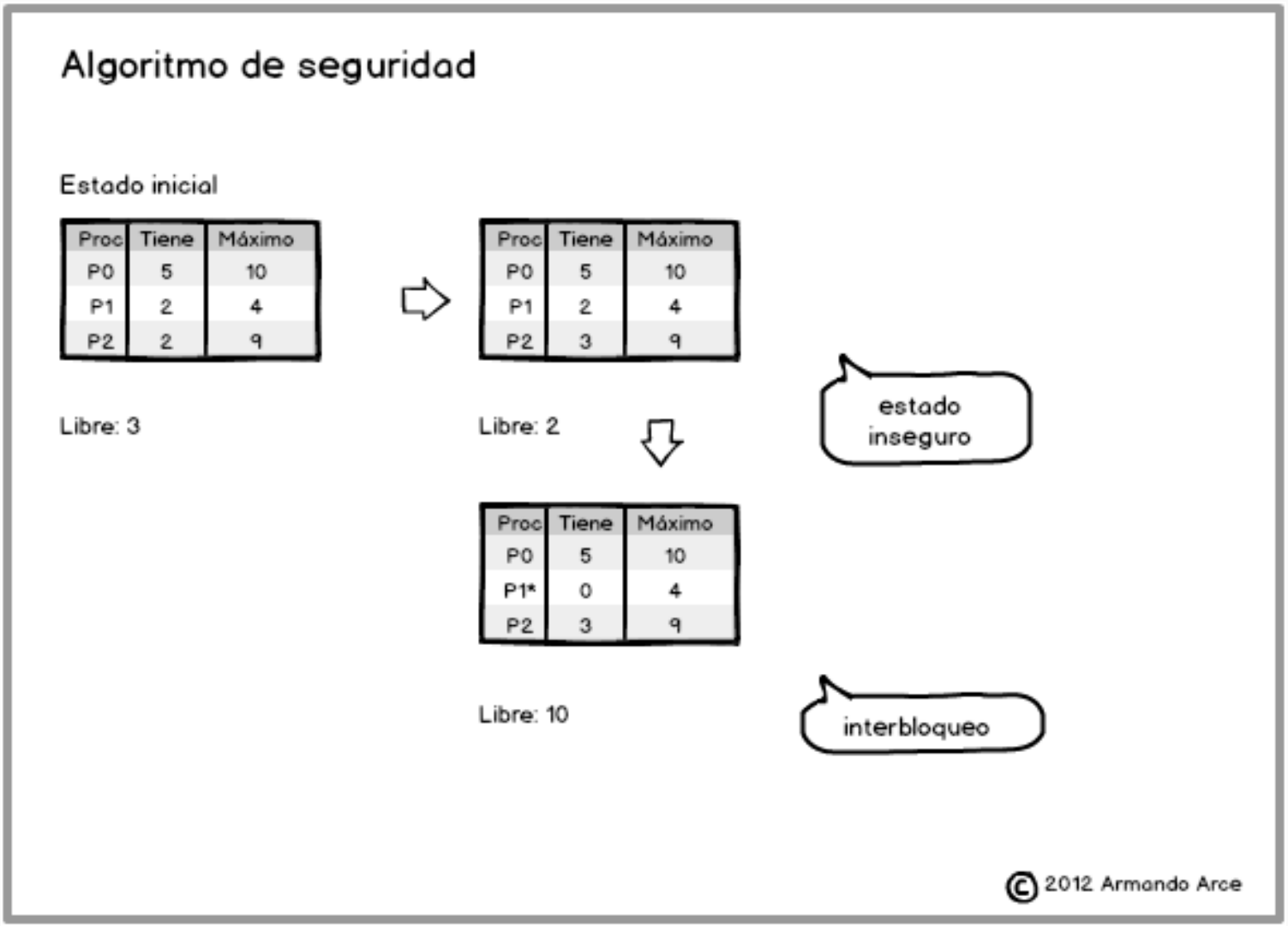

Figure 7:

Estado seguro

Conocido el concepto de estado seguro, se pueden definir algoritmos para evitar los interbloqueo que aseguren que en el sistema nunca se producirá un interbloqueo.

- La idea consiste simplemente en garantizar que el sistema siempre se encuentre en estado seguro.
- Inicialmente, el sistema se encuentra en dicho estado.
- Cuando un proceso solicita un recurso que está disponible en ese momento, el sistema debe decidir si el recuerdo puede asignarse de forma inmediata o el proceso debe esperar.
- La solicitud se concede sólo si la asignación deja el sistema en estado seguro.

Grafo de asignación de recursos

Si el sistema de asignación de recursos consta de una sola instancia por cada tipo de recurso, puede utilizarse una variante del grafo de asignación de recursos para evitar los interbloqueos.

- Además de las aristas de solicitud y de asignación se introduce un nuevo tipo de arista, denominado arista de declaración.
- Una arista de declaración indica que un proceso solicitará un recurso en algún instante futuro.
- Esta arista es similar a una arista de solicitud en lo que respecta a al dirección, pero se representa en el grafo mediante una línea punteada.
- Luego, cuando un proceso libera un recurso, la arista de asignación se reconvierte en una arista de declaración.

Grafo de asignación de recursos

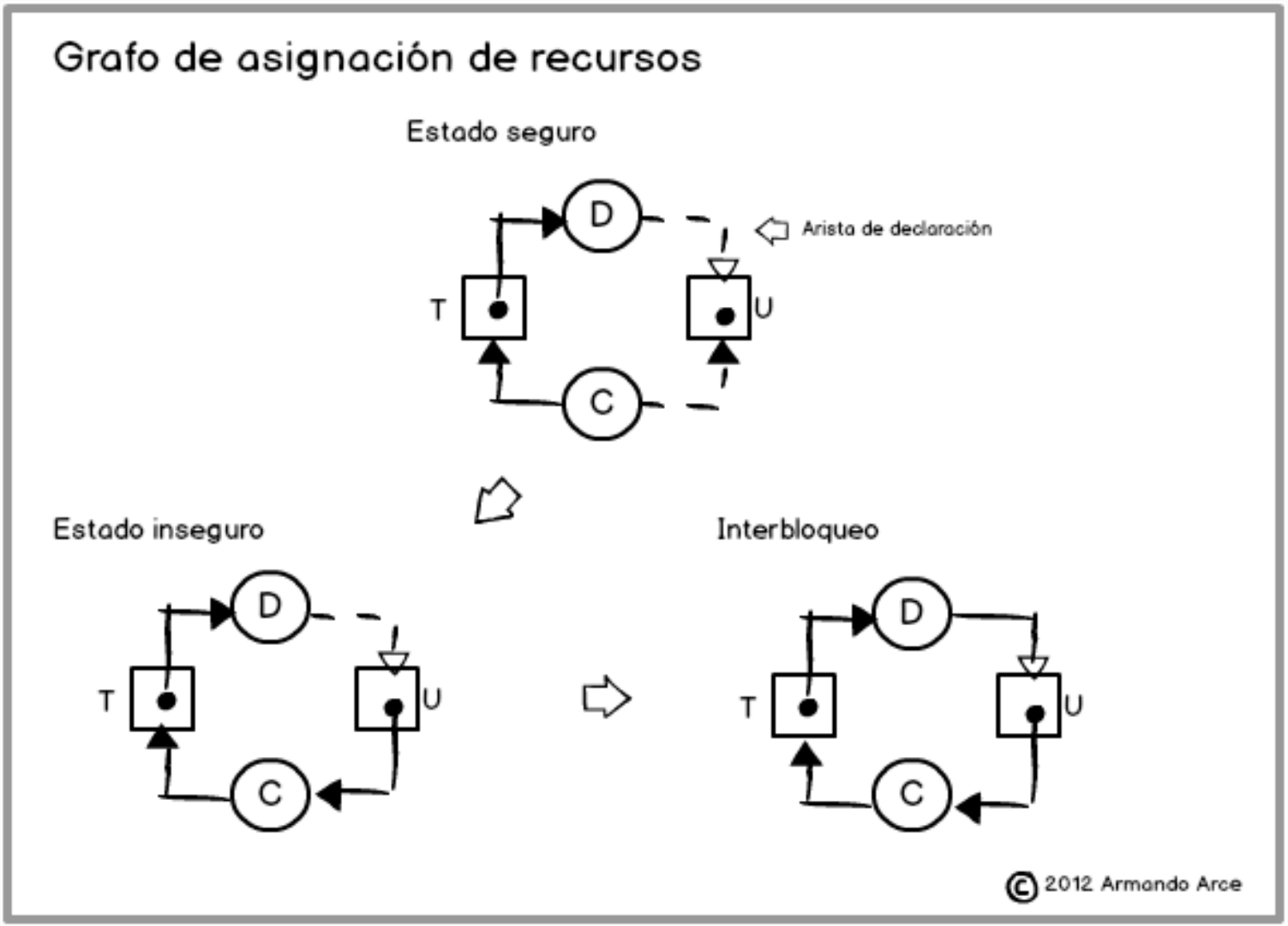

Figure 8:

Grafo de asignación de recursos

Si un proceso solicita un recurso, la solicitud se puede conceder sólo si la conversión de la arista de solicitud en una arista de asignación no da lugar a la formación de un ciclo en el grado de asignación de recursos.

- Para detectar si el estado es seguro se utiliza un algoritmo de detección de ciclos.
- Los algoritmos de detección de ciclos en este grafo requieren del orden de n^2 operaciones, donde n es el número de procesos del sistema.

Grafo de asignación de recursos

Si no se crea un ciclo, entonces la asignación del recurso dejará al sistema en un estado seguro. Si se encuentra un ciclo, entonces la asignación colocaría al sistema en un estado inseguro.

- Por tanto, el proceso que hizo la solicitud tendrá que esperar para que su solicitud sea satisfecha.

Algoritmo del banquero

El grafo de asignación de recursos no puede aplicarse a un sistema de asignación de recursos con múltiples ejemplares de cada tipo de recursos.

- El algoritmo de evasión de interbloqueos, llamado “algoritmo del banquero”, puede aplicarse a tal sistema, pero es menos eficiente que el esquema de grafo de asignación de recursos.

Algoritmo del banquero

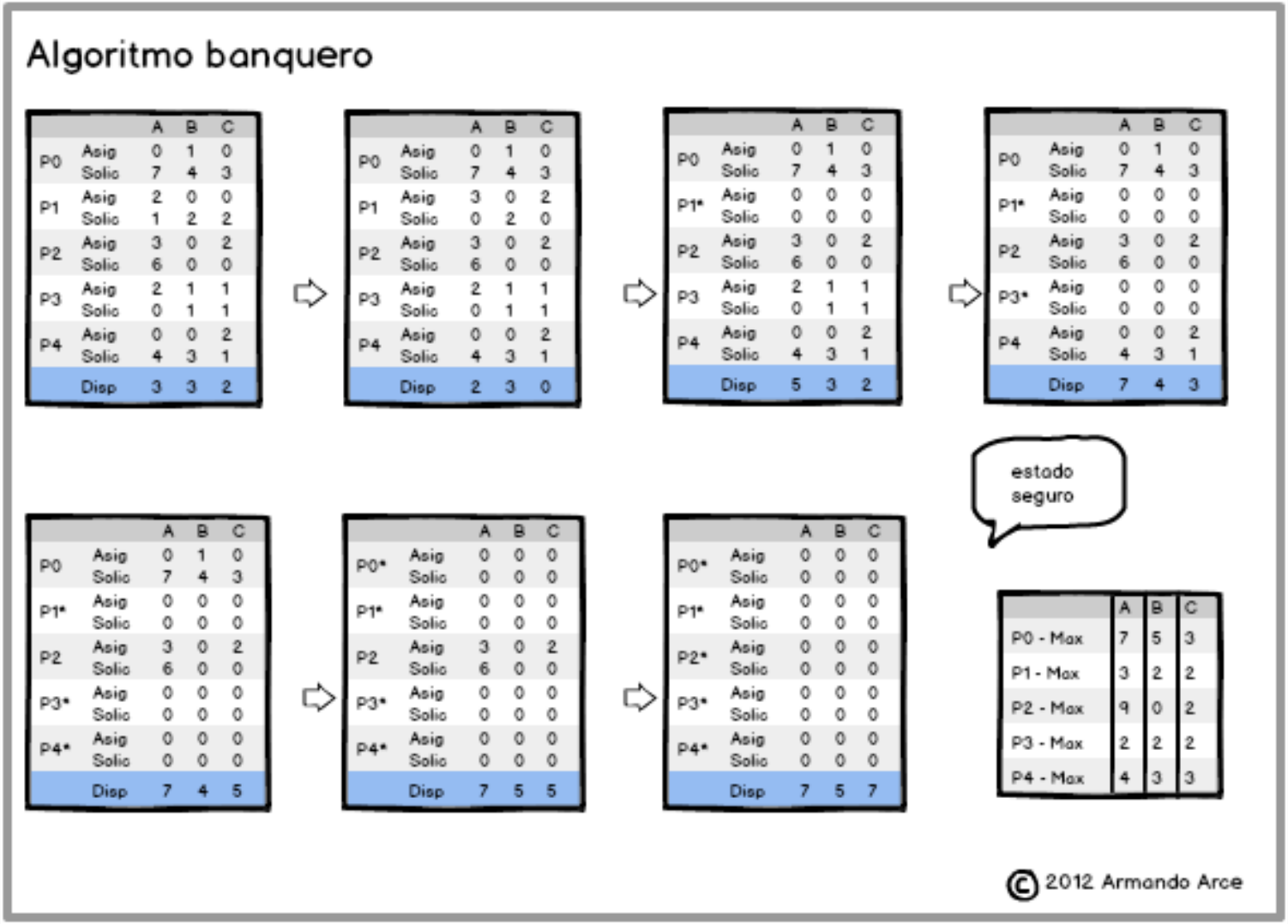

Figure 9:

Algoritmo del banquero

Cuando un proceso nuevo ingresa al sistema, debe declarar el número máximo de ejemplares de cada tipo de recursos que podría necesitar.

- Este número no puede exceder el total de recursos del sistema. Cuando un usuario solicita un conjunto de recursos, el sistema debe determinar si la asignación de esos recursos dejaría el sistema en un estado seguro.
- Si es así, los recursos se asignan; si no, el proceso deberá esperar hasta que otro libere los recursos suficientes.

Algoritmo del banquero

Figure 10:

Algoritmo del banquero

Lo que hace el algoritmo del banquero es “simular” que le asigna los recursos que solicita el proceso, luego con base en ese nuevo estado “simulado” aplica el algoritmo de seguridad para determinar si dicha asignación provocará que el sistema se mantenga en estado seguro.

- Si el estado que resulta de la asignación de recursos es seguro, la transacción se completa y se asignan los recursos al proceso que los solicitó.
- Sin embargo, si el nuevo estado resulta inseguro, el proceso debe esperar, y se restaura el antiguo estado de asignación de recursos.

Detección de interbloqueos

Si un sistema no emplea ni algoritmos de prevención ni de evasión de interbloqueos, entonces puede producirse una situación de interbloqueo del sistema. En este caso, el sistema de proporcionar:

- Un algoritmo que examine el estado del sistema para determinar si se ha producido un interbloqueo
- Un algoritmo para recuperarse del interbloqueo

Hay que resaltar de antemano que los esquemas de detección y recuperación tienen un coste significativo asociado, que incluye no sólo el coste (de tiempo de ejecución), asociado con el mantenimiento de la información necesaria y la ejecución del algoritmo de detección, sino también las potenciales pérdidas inherentes al proceso de recuperación de un interbloqueo.

Una sola instancia de cada tipo de recurso

Si todos los recursos tienen un única instancia, entonces se puede definir un algoritmo de detección de interbloqueos que utilice una variante del grafo de asignación de recursos, denominada grafo de espera.

- Se obtiene este grado a partir del grado de asignación de recursos eliminando los nodos de recursos y colapsando las correspondientes aristas.

Una sola instancia de cada tipo de recurso

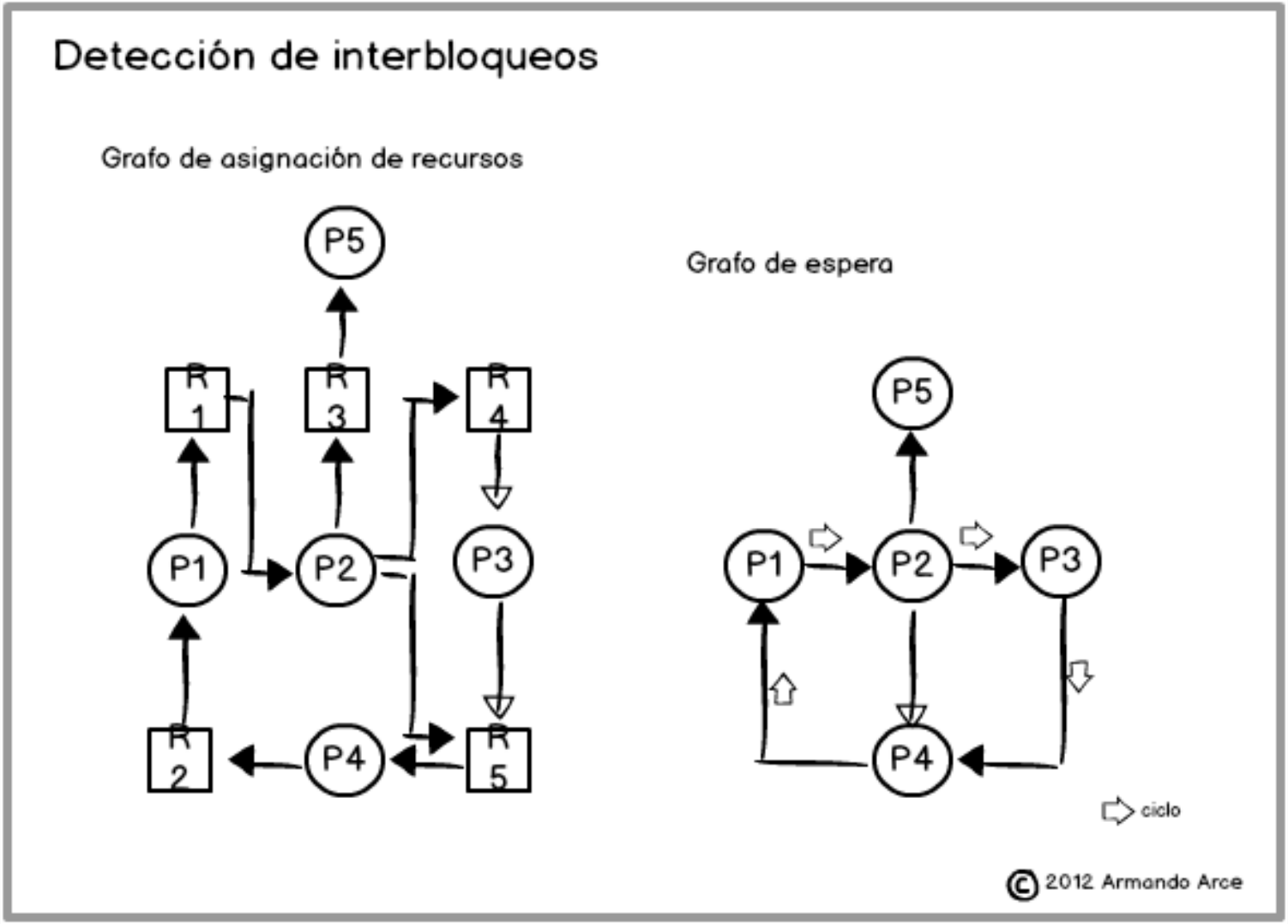

Figure 11:

Una sola instancia de cada tipo de recurso

Como antes, existirá un interbloqueo en el sistema si y sólo si el grafo de espera contiene un ciclo.

- Para detectar los interbloqueos, el sistema necesita mantener el grafo de espera e invocar un algoritmo periódicamente que compruebe si existe un ciclo en el grafo.
- Un algoritmo para detectar un ciclo en un grafo requiere del orden de n^2 operaciones, donde n el número de vértices del grafo (o sea, procesos, en este caso).

Varias instancias de cada tipo de recurso

El esquema del grafo de espera no es aplicable a los sistemas de asignación de recursos con múltiples instancias de cada tipo de recurso.

- Sin embargo, existe un algoritmo de detección de interbloqueos que puede aplicarse en este tipo de sistemas.

Varias instancias de cada tipo de recurso

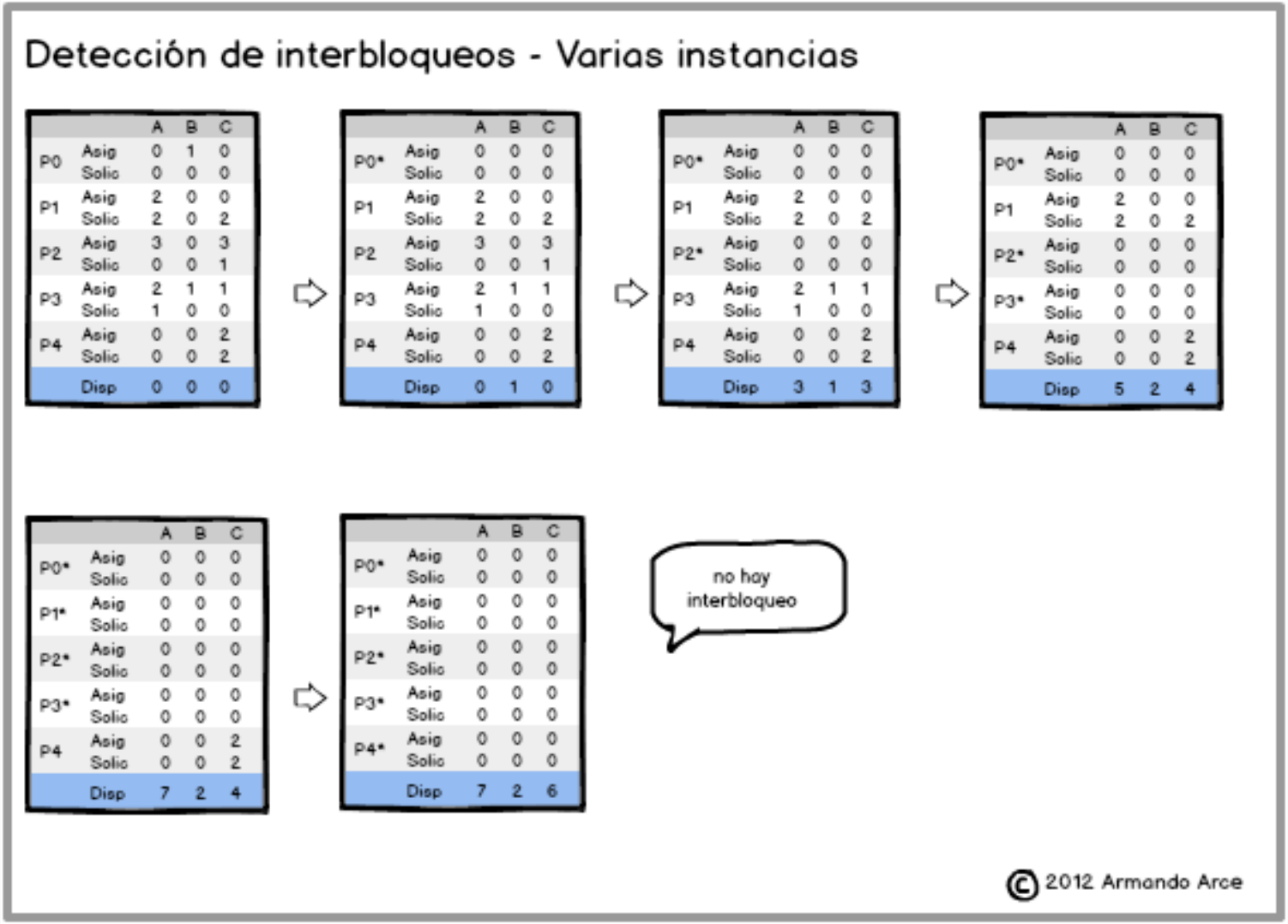

Figure 12:

Varias instancias de cada tipo de recurso

Este método resulta similar al algoritmo del banquero y lo que hace es investigar cada posible secuencia de asignación de los procesos que quedan por completarse.

- Por ello el algoritmo requiere de m x n^2 operaciones (donde m representa la cantidad de recursos y n la cantidad de procesos) para detectar si el sistema se encuentra en estado de interbloqueo.

Uso del algoritmo de detección

Si los interbloqueos son frecuentes, el algoritmo de detección deberá invocarse muy seguido.

- Los recursos asignados a procesos bloqueados estarán ociosos hasta que se pueda romper el interbloqueo.
- Además, el número de procesos implicados en el ciclo de bloqueo mutuo podría crecer.

Uso del algoritmo de detección

Se podría invocar el algoritmo de detección de interbloqueos cada vez que una solicitud de asignación no se puede atender de inmediato.

- En este caso, no sólo se puede identificar el conjunto de procesos que están en interbloqueo, sino también el proceso específico que "causó" el bloqueo mutuo.

Uso del algoritmo de detección

Sin embargo, la invocación del algoritmo de detección de interbloqueos en cada solicitud podría incurrir en un gasto extra de tiempo de cómputo considerable.

- Una alternativa menos costosa es simplemente invocar el algoritmo a intervalos menos frecuentes; por ejemplo, una vez cada hora, o siempre que el grado de utilización de la CPU cae por debajo del 40%.

Recuperación de interbloqueos

Cuando un algoritmo de detección determina que existe un interbloqueo, se dispone de varias alternativas.

- Una posibilidad es informar al operador que ha ocurrido un interbloqueo y dejar que él lo maneje manualmente.
- La otra es dejar que el sistema se recupere del interbloqueo automáticamente. Hay dos opciones para romper el interbloqueo.
- Una solución simplemente aborta uno o más procesos para romper la espera circular.
- La segunda opción es expropiar recursos de uno o más de los procesos bloqueados.

Recuperación de interbloqueos

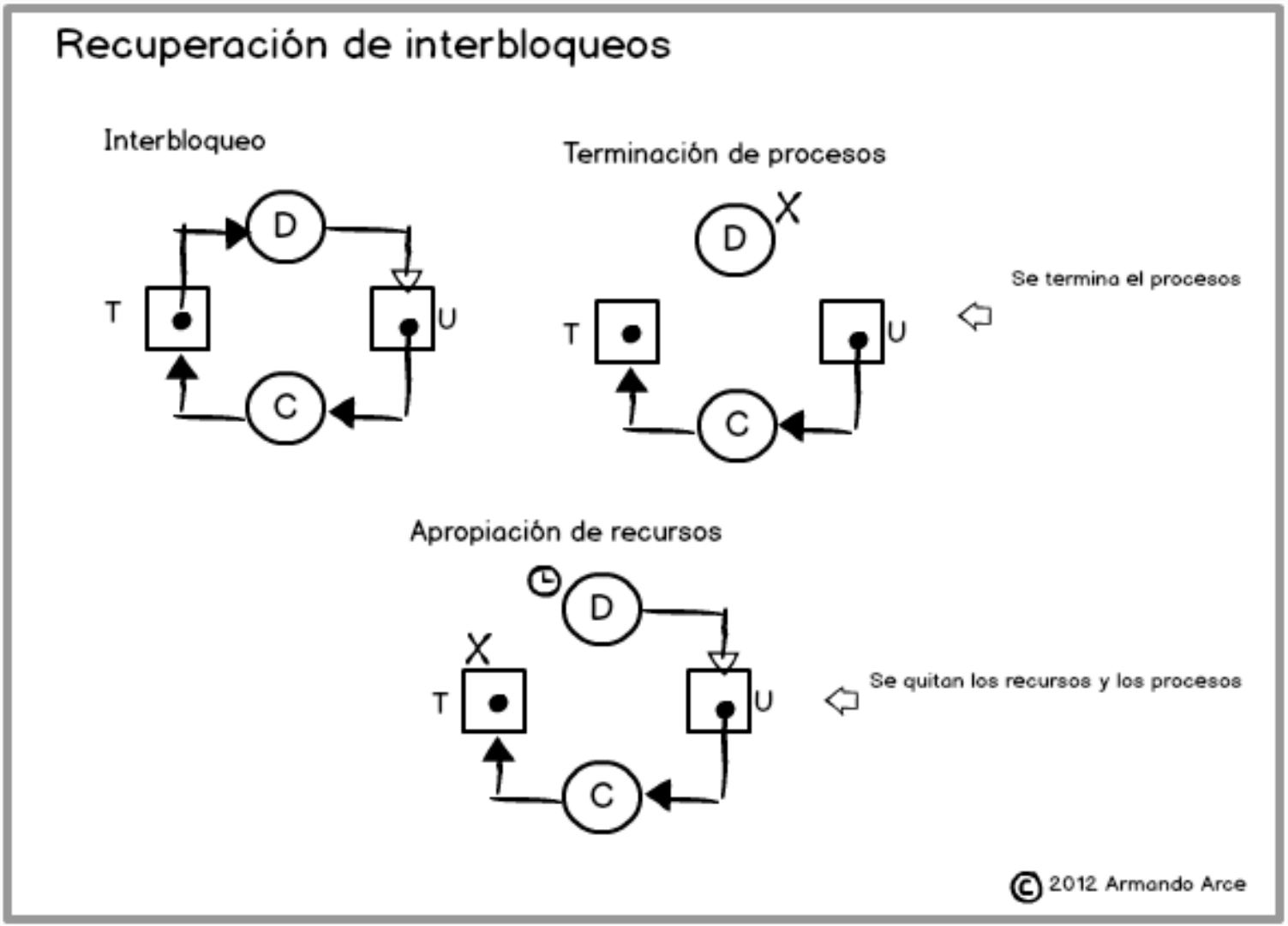

Figure 13:

Terminación de procesos

Para eliminar interbloqueos abortando un proceso, se utiliza uno de dos métodos. En ambos, el sistema recupera todos los recursos asociados a los procesos finalizados.

- Interrumpir todos los procesos bloqueados: Una desventaja es que se pierde todo el trabajo realizado hasta el momento. Además, los recursos pueden quedar en estado inconsistente.

Terminación de procesos

- Interrumpir un proceso cada vez: La desventaja es que luego de detener el proceso hay que hacer la detección del interbloqueo, lo cual requiere tiempo y recursos.

En el último caso se debe determinar cuál proceso se debe abortar para tratar de romper el interbloqueo.

- Muchos factores podrían influir en la selección de los procesos entre ellos: prioridad, tiempo de ejecución, cantidad de recursos utilizados, y si son procesos interactivos o en lotes.

Expropiación de recursos

Para eliminar interbloqueos utilizando expropiación de recursos, se expropia sucesivamente algunos recursos de los procesos y se le dan a otros procesos hasta romper el ciclo de interbloqueo.

Si se necesita expropiación para manejar los interbloqueos, es preciso considerar tres aspectos:

- Selección de la víctima: Se debe determinar el orden de expropiación para minimizar el costo. Los factores de costo podrían incluir parámetros tales como el número de recursos que un proceso bloqueado tiene asignados, y el tiempo que un proceso bloqueado ha consumido durante su ejecución.

Expropiación de recursos

- Retroceso: Se debe retroceder el proceso a algún estado seguro, y reiniciarlo desde ese estado. La solución más sencilla es un retroceso total: se aborta el proceso y luego se reinicia. Sin embargo, es preferible un método menos drástico.
- Inanición: Si la selección de la víctima se basa principalmente en factores de costo, podría suceder que siempre se escoja el mismo proceso como víctima. En consecuencia, dicho proceso nunca terminaría su tarea asignada (inanición). La solución más común es incluir el número de retrocesos en el factor de costo.

Referencia bibliográfica

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap. 7, 9a Edición, John Wiley, 2013.