Facultad de Ingeniería en Electricidad y Computación

Programación de Sistemas CCPG1051

Federico Domínguez, PhD.

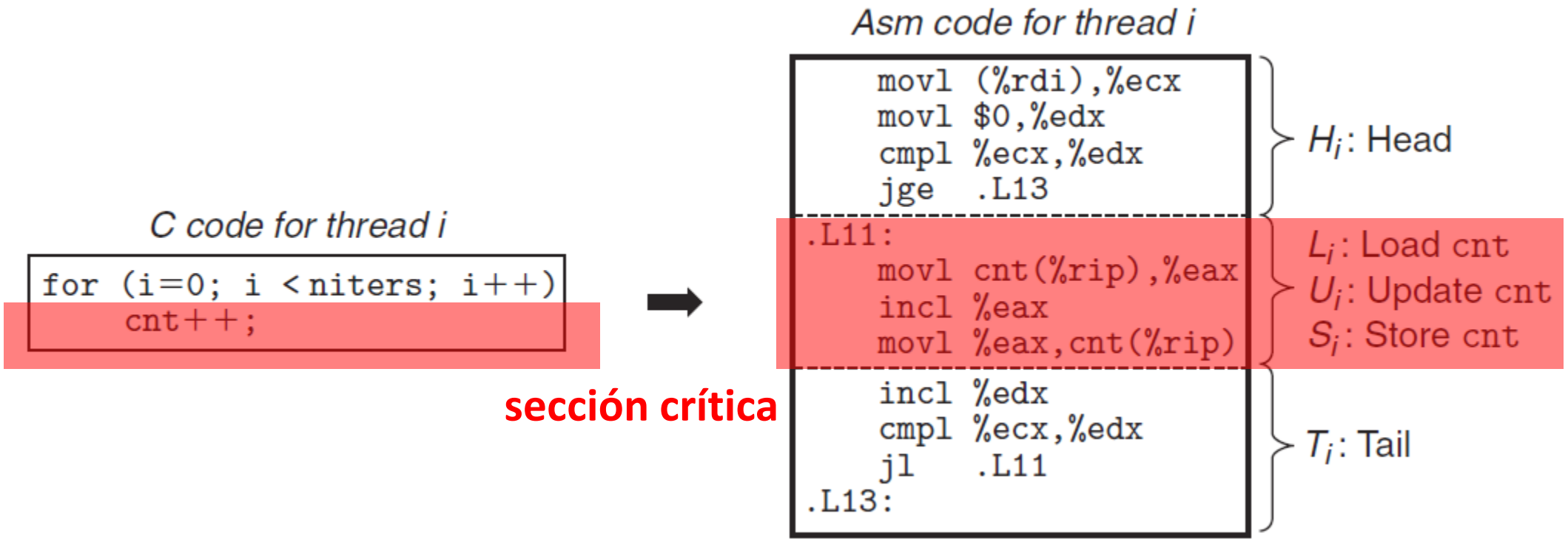
Unidad 6 – Sesión 8: Progress graphs y Deadlocks

Agenda

Progress graphs

Deadlocks

Sección crítica: sección de código que manipula una variable compartida entre varios hilos.



La exclusión mutua en una sección crítica garantiza la ejecución correcta de dos o más hilos.

Step	Thread	Instr	%eax$_1$	%eax$_2$	cnt
1	1	H_1	—	—	0
2	1	L_1	0	—	0
3	1	U_1	1	—	0
4	1	S_1	1	—	1
5	2	H_2	—	—	1
6	2	L_2	—	1	1
7	2	U_2	—	2	1
8	2	S_2	—	2	2
9	2	T_2	—	2	2
10	1	T_1	1	—	2

(a) Correct ordering

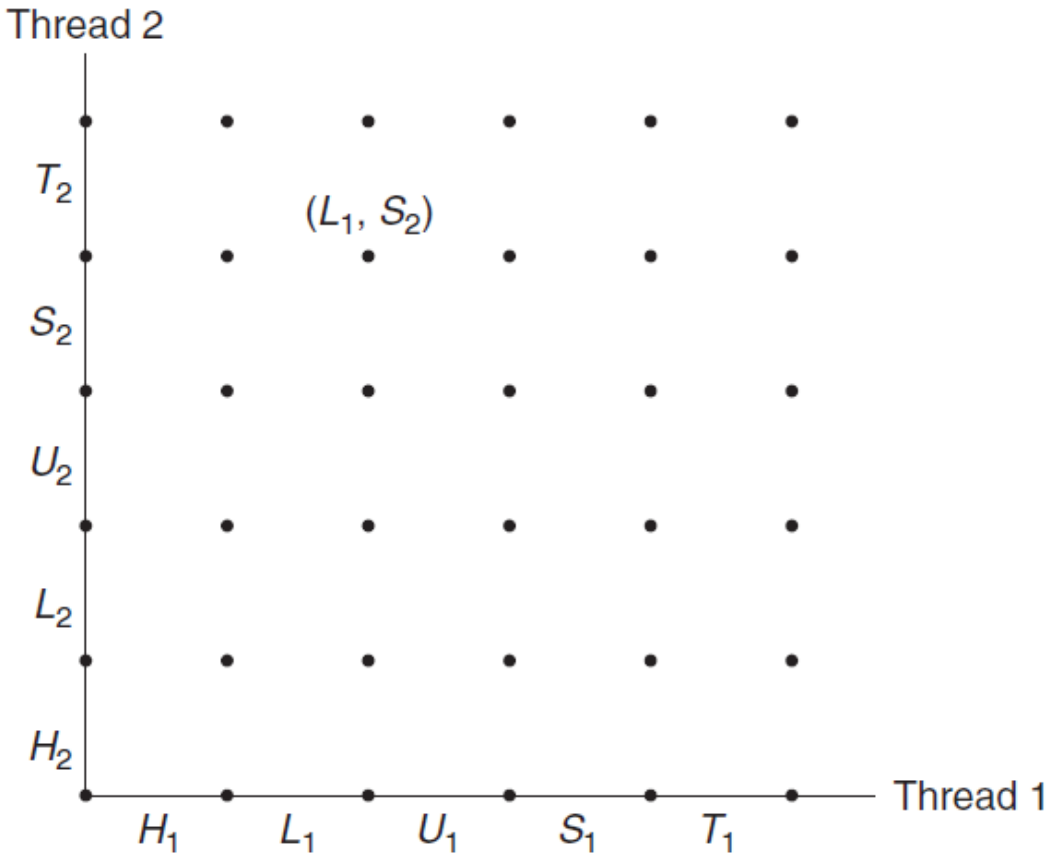
Step	Thread	Instr	%eax$_1$	%eax$_2$	cnt
1	1	H_1	—	—	0
2	1	L_1	0	—	0
3	1	U_1	1	—	0
4	2	H_2	—	—	0
5	2	L_2	—	0	0
6	1	S_1	1	—	1
7	1	T_1	1	—	1
8	2	U_2	—	1	1
9	2	S_2	—	1	1
10	2	T_2	—	1	1

(b) Incorrect ordering

Los "*Progress Graphs*" son una forma útil de visualizar ejecuciones simultáneas de hilos y encontrar posibles problemas.

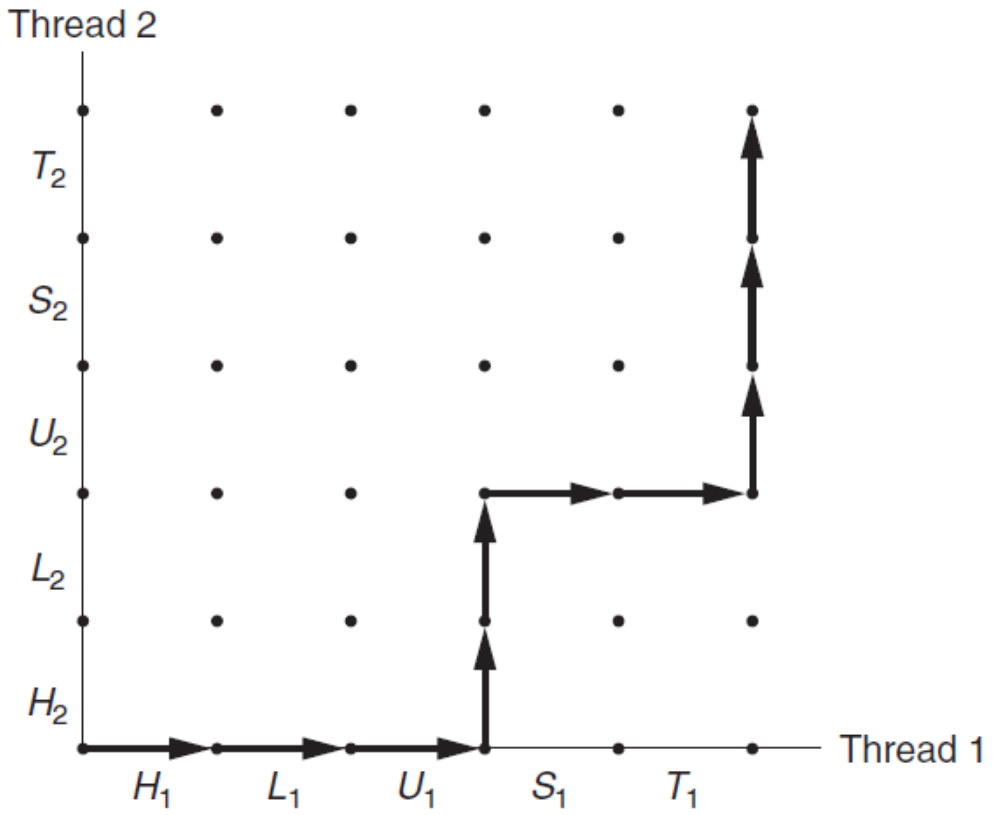
Cada eje representa la ejecución de un hilo.

Cada punto representa una instrucción.



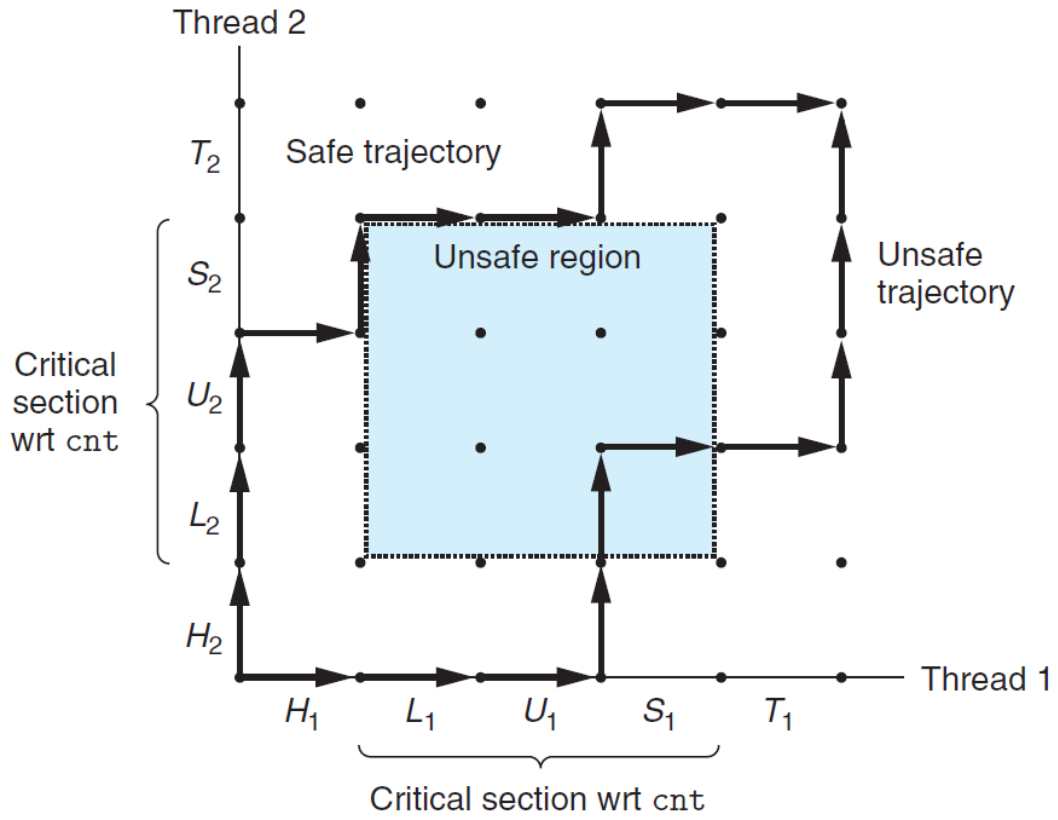
Una trayectoria es la ejecución combinada de los hilos en el gráfico.

Un programa puede seguir varias trayectorias, esto depende de la planificación del kernel y no del programador.



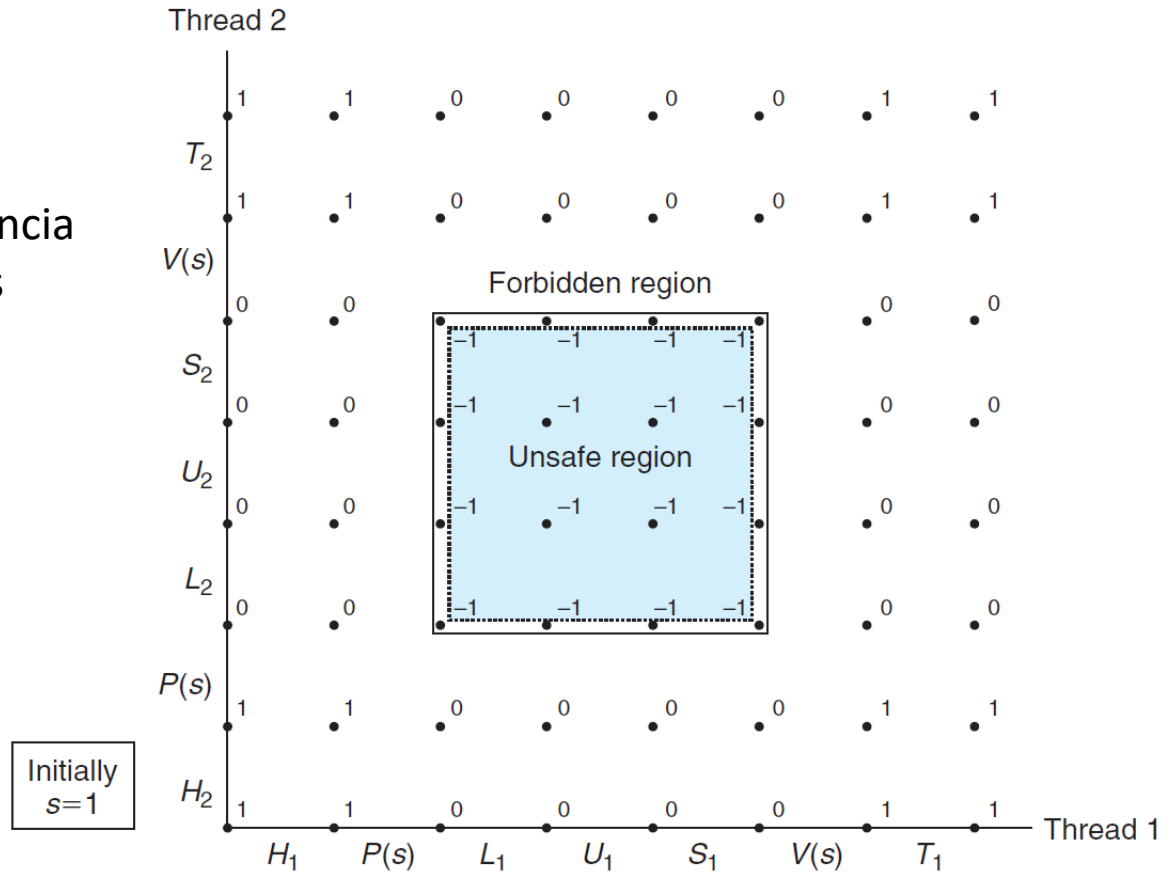
La intersección de secciones críticas en el gráfico son por definición zonas inseguras...

Si una trayectoria entra a una zona insegura, el programa producirá resultados erróneos.

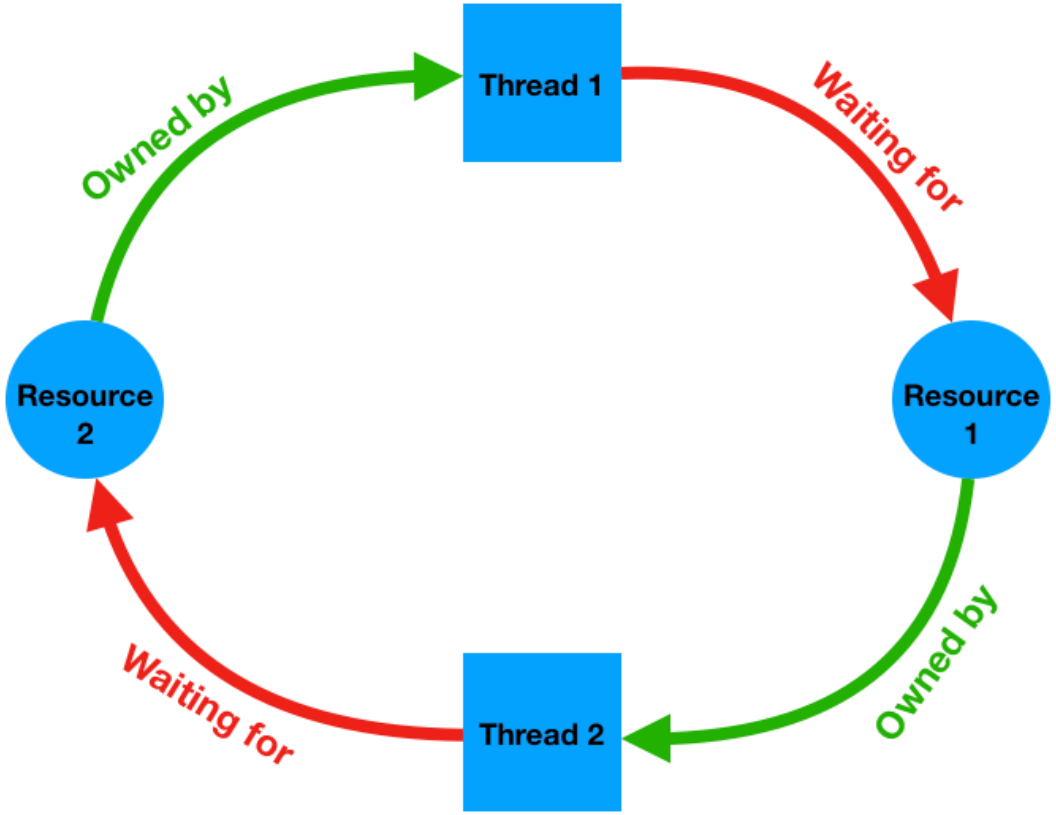


Al usar semáforos de exclusión mutua, las zonas no seguras son imposibles de ingresar debido a que requieren un estado no válido del semáforo (s<0).

El modelamiento de concurrencia con "*Progress Graphs*" solo es válido en sistemas con un procesador.



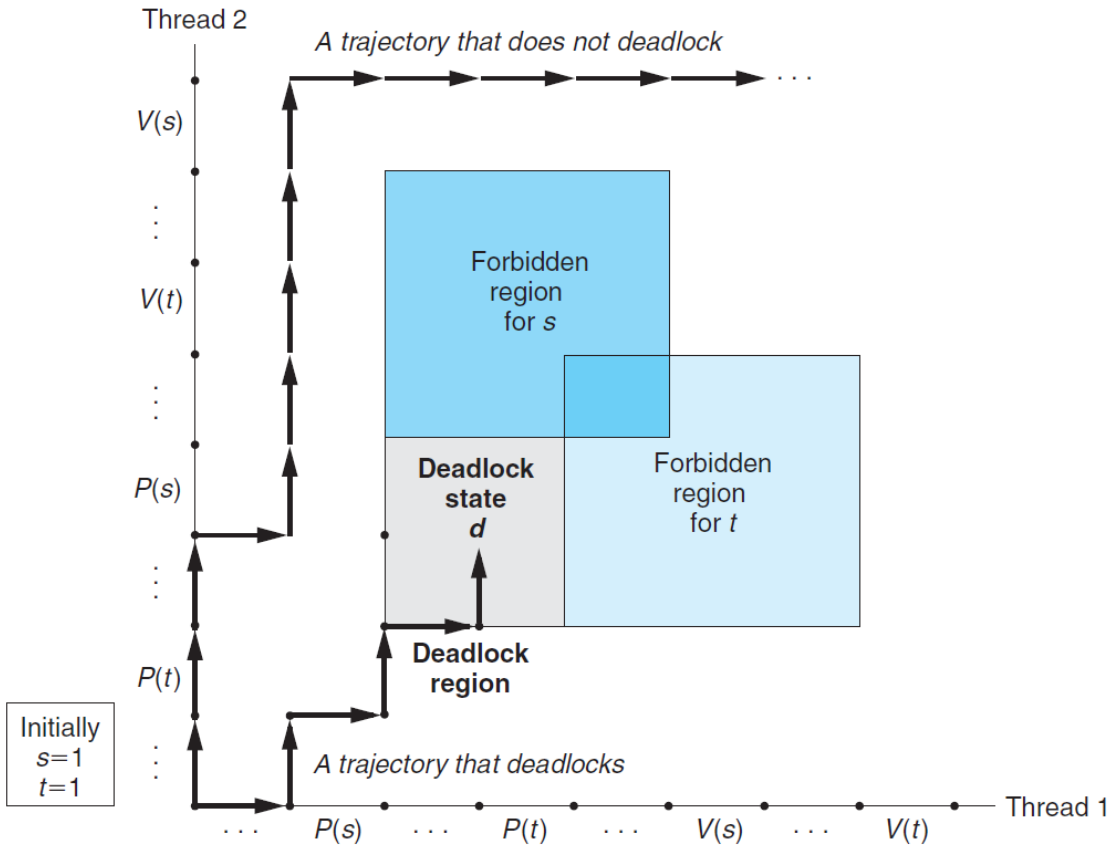
Un **deadlock** es cuando varios hilos se bloquean esperando una condición que nunca será cierta.



El uso de semáforos introduce el riesgo de **deadlocks**.

Usando un "*Progress Graph*" podemos visualizar trayectorias que incurren en **deadlocks**.

En este ejemplo dos hilos usan dos semáforos creando dos zonas no seguras que se intersecan.



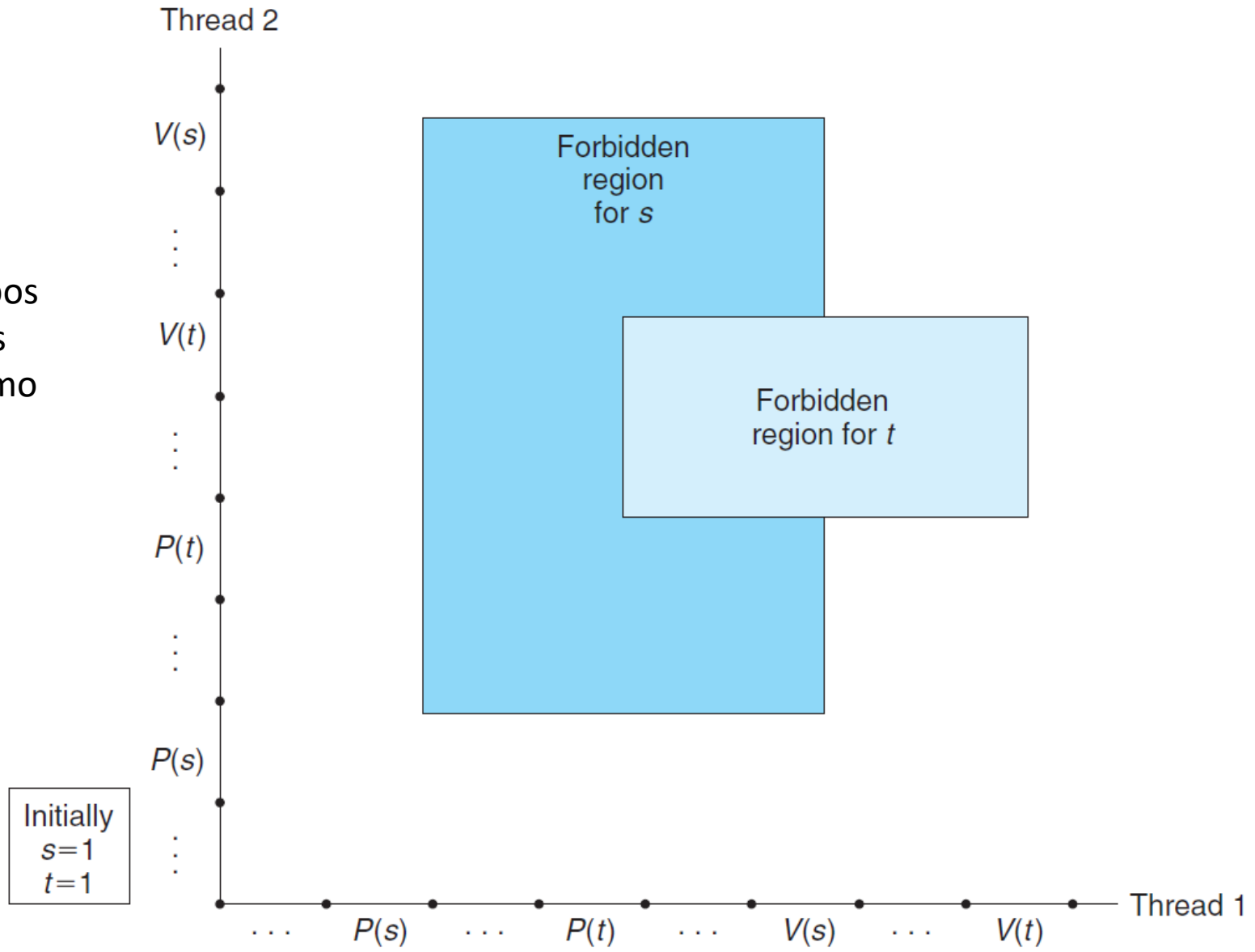
Los **deadlock** son particularmente difíciles de resolver.

Producen programas que se “cuelgan” y en muchas casos la única solución es detener la ejecución del mismo.

Cuando semáforos binarios están siendo usados para exclusión mutua existe una solución simple:

- Para cada par de semáforos binarios *(s , t)* en un programa, cada hilo que usa *s* y *t* de manera simultánea, los debe bloquear en el mismo orden.

En este ejemplo ambos hilos reservan ambos semáforos en el mismo orden.



Referencias

Libro guía Computer Systems: A programmer’s perspective. Sección 12.7