Redes de Petri

- Donato, Juan Pablo
- Botalla, Tomas
- Alvarez, Dylan

“MODELING MULTITHREADED APPLICATIONS USING PETRI NETS”

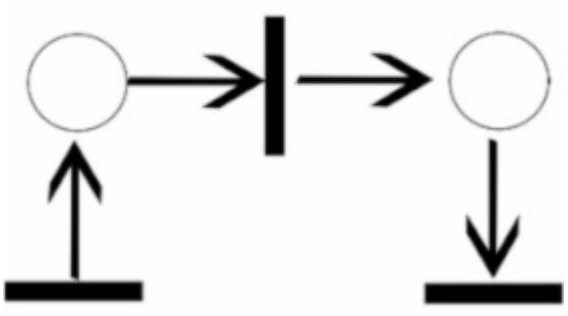
- Autores
  - Kavi, K.
  - Moshtaghi, A.
  - Chen, D.

- Objetivo: Aplicación de modelo de Redes de Petri para modelar problemas de concurrencia. Presentación del software **C2Petri** que modela programas escritos en C como Redes de Petri.

¿Redes de Petri? ¿Que son?

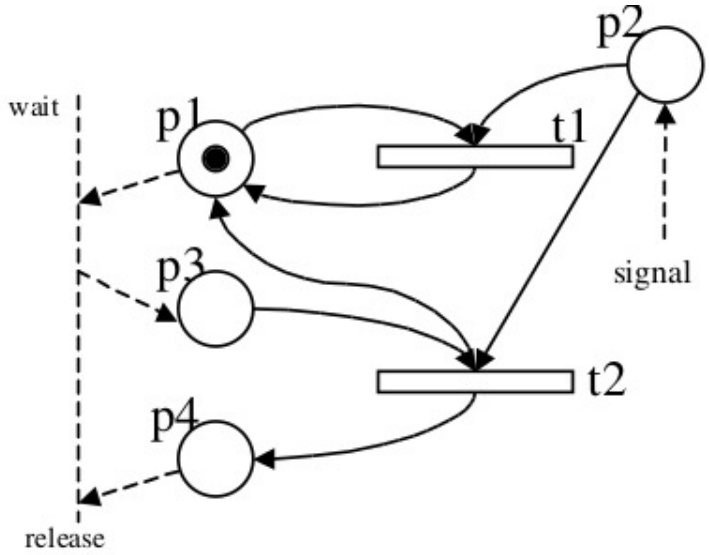
Redes de Petri

- Grafo **dirigido** y **bi-partito**
  - Los **lugares** solo se conectan entre sí a través de **transiciones**
- Se busca **modelar** estados y transiciones de **aplicaciones multithreaded** para **detectar potenciales problemas**
  - **Race conditions**, **lost signals**, **deadlocks**
  - Se mapean nodos del grafo a líneas de código
- Las **redes de Petri fueron muy estudiadas**, llevar el problema a este formato permite **aprovechar eso**



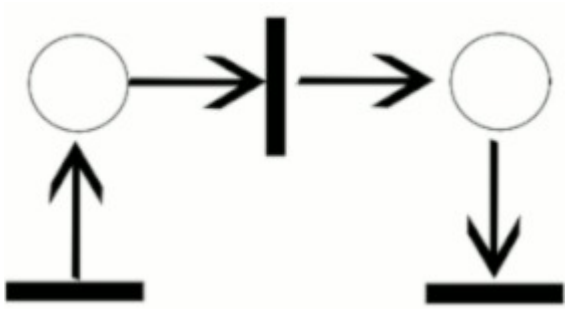
Componentes

- **Lugares P** = { p_1 , p_2 , ... , p_m }
- Vector de **markings M**
  - Cantidad de **tokens** (puntos negros) **en cada** uno de los **m lugares**
  - **M_0** es el vector de markings **(estado) inicial**
- **Transiciones T** = { t_1 , t_2 , ... , t_L }
- **Aristas F** $\subseteq$ (P x T) U (T x P)
  - Solo entre lugares y transiciones (**bipartito**)
- **Pesos W,** F $\longrightarrow$ {1,2,3,...}
  - Peso de cada arista: **cuántos tokens** pasan a la vez



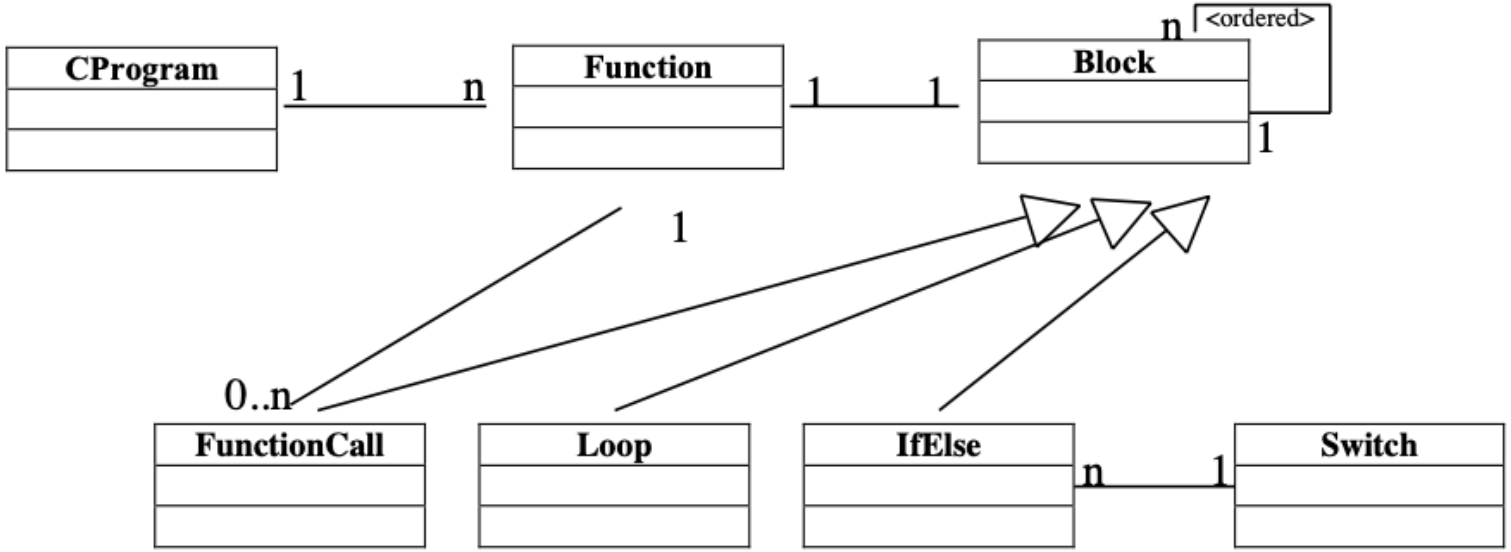
Transiciones o cambios de estado en Redes de Petri

- Una **transición** se **activa** cuando todos los **lugares que entran** a la misma llegan a la **cantidad de tokens** correspondiente al **peso de la arista**
- Una transición **activa puede o no ejecutarse**
  - **Dependerá del evento** que representa
- Cuando se ejecuta, se **remueven** los tokens de los **lugares de entrada** y se **colocan en la salida** tokens **según el peso de las aristas**
  - Pueden aparecer o desaparecer tokens del



¿Cómo generar la red completa de un programa?

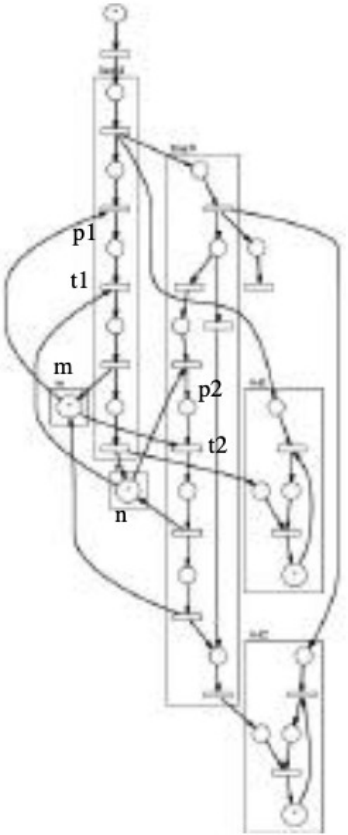
Modelo de un programa completo



- No se contemplan funciones recursivas ni arrays o punteros de thread types porque no se puede crear la red de Petri de forma dinámica.
- El paper propone seleccionar un par de combinaciones posibles en este caso y generar N redes de Petri

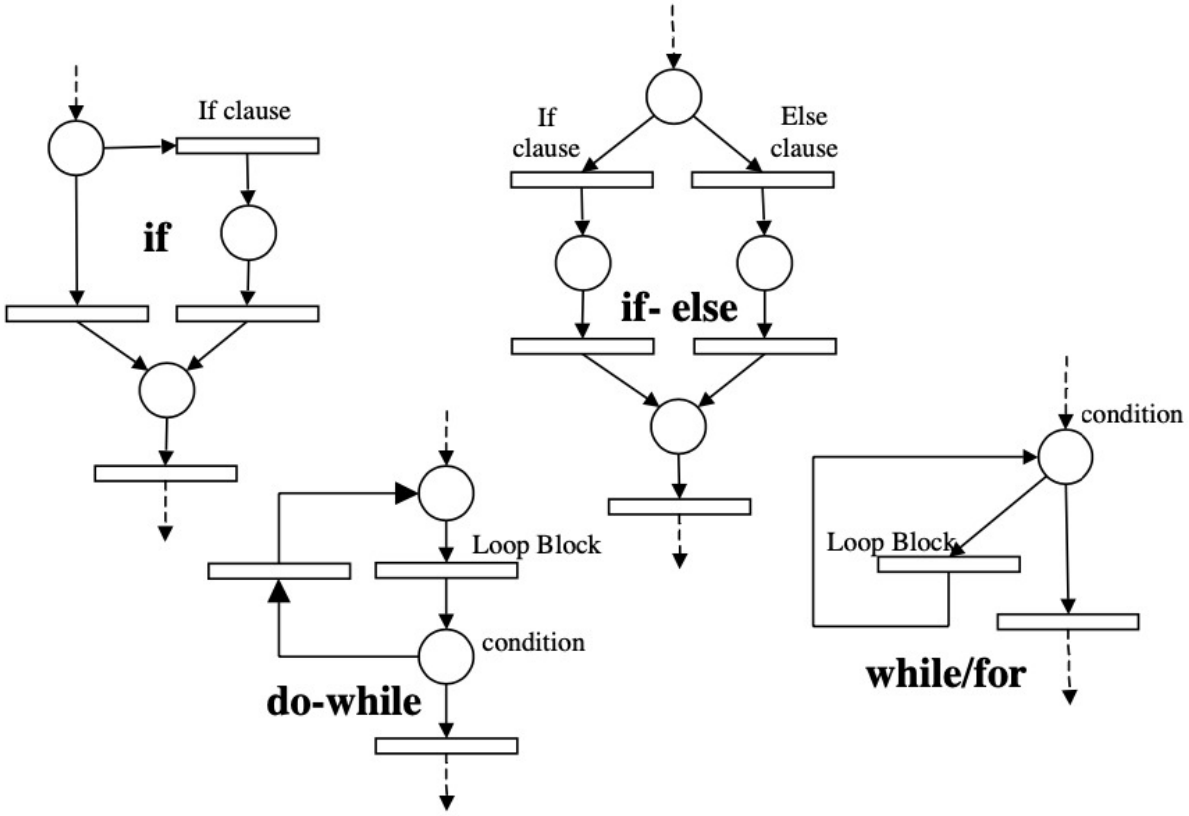
Modelo de un programa completo

- Se toman las instrucciones de **control de flujo**, **invocación a función** y manejo de la **librería pthread**
- Se las modela como **componentes interconectados**
  - Loop, Function...
- Se **traduce** ese modelo a una **red de Petri**
- Se evalúan las **posibles secuencias** de **ejecución de transiciones**
- Se **reportan los marking** asociados con **warnings y errores**

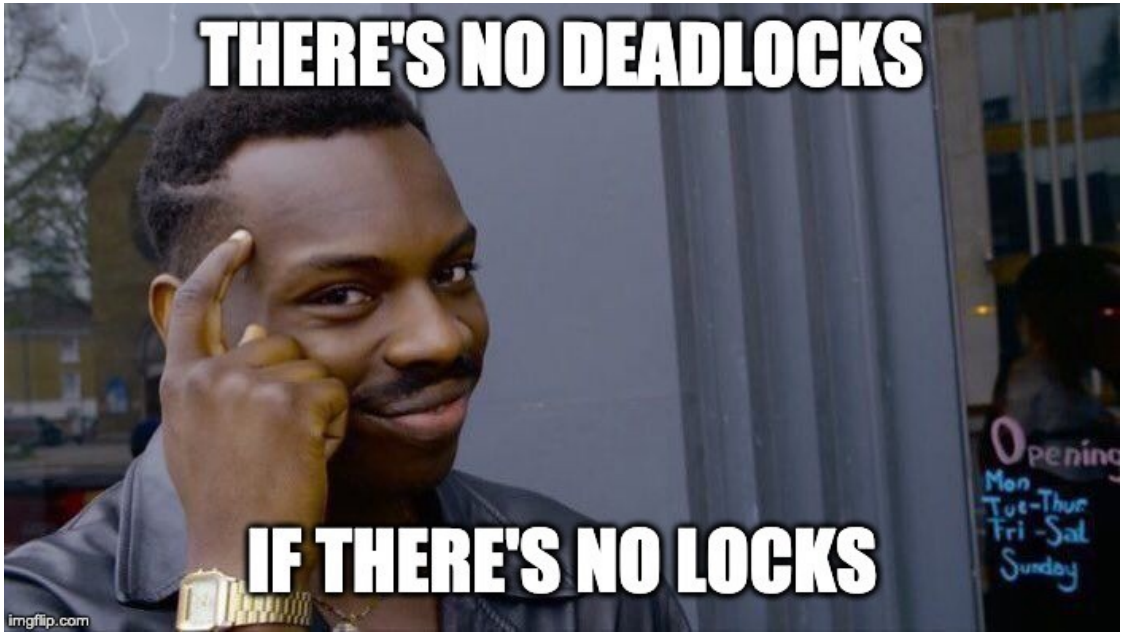


Modelo de un programa completo

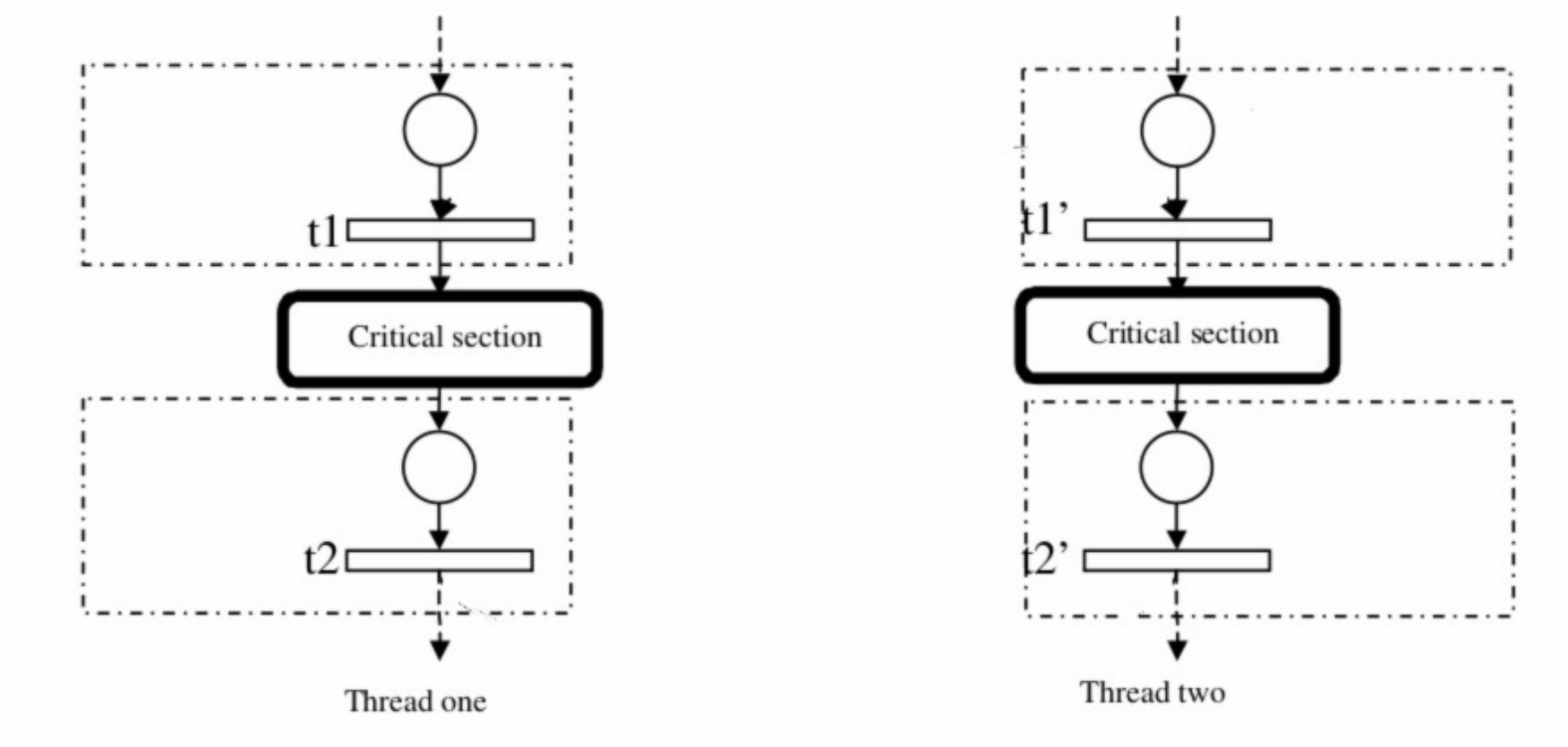
- Se arranca con un **lugar que contiene un token** (**program start**)
- Se termina con una **transición** (**program end**)
- Un **marking M** corresponde a un **deadlock** cuando:
  - Se puede llegar a M
  - **No hay** un **camino** de M **a otro estado**
  - M **no es** el estado ***program***



Veamos un ejemplo de aplicación con un problema de concurrencia!

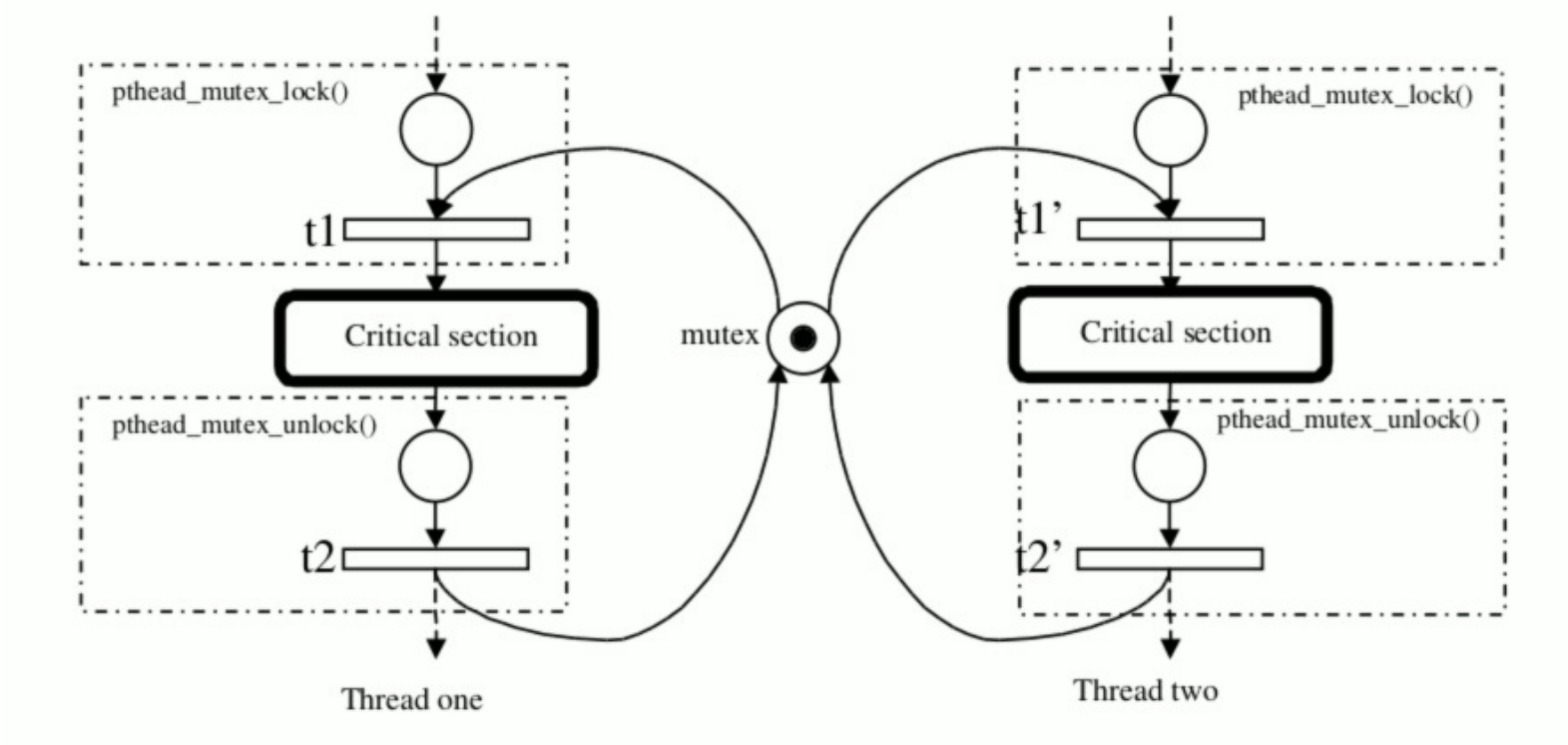


Race Condition



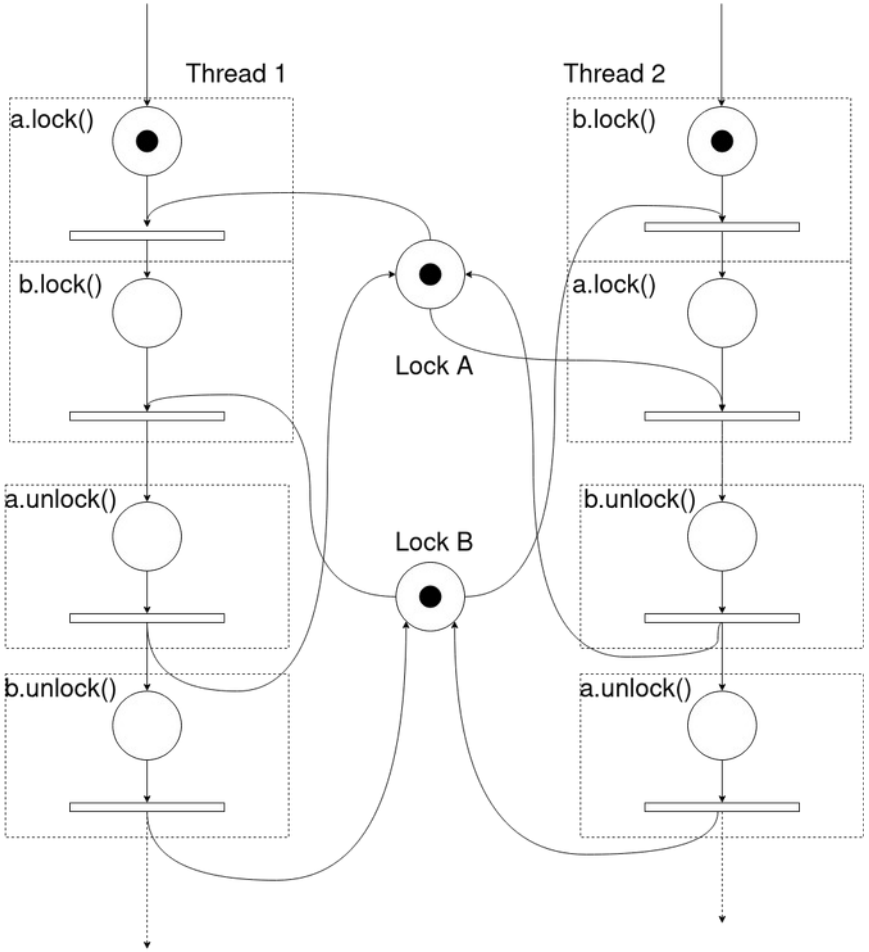
- Output del programa depende de la secuencia de los eventos
- Resultado no determinístico

Lock / Unlock de un Mutex

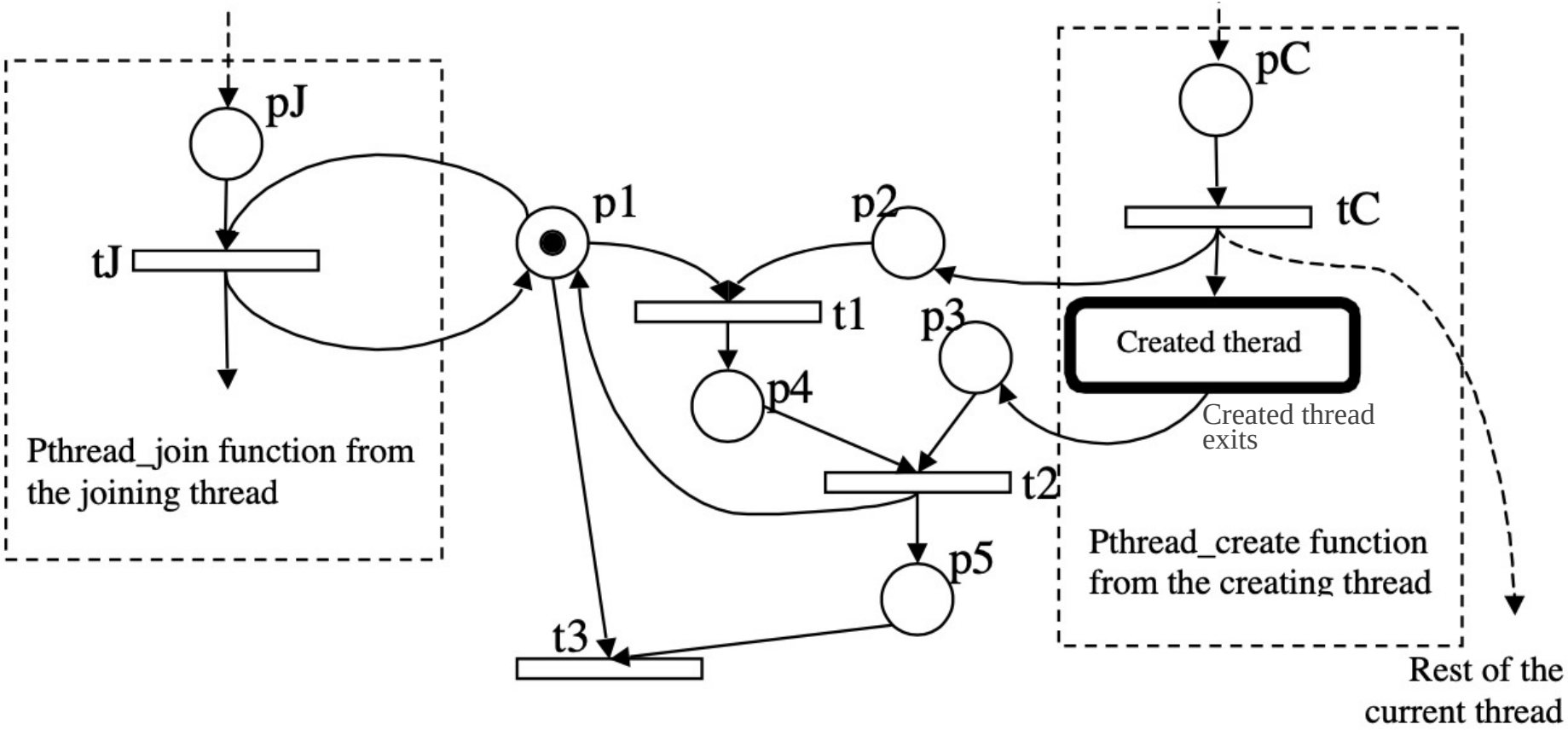


- El resultado ya no dependerá del orden de aparición de los eventos!

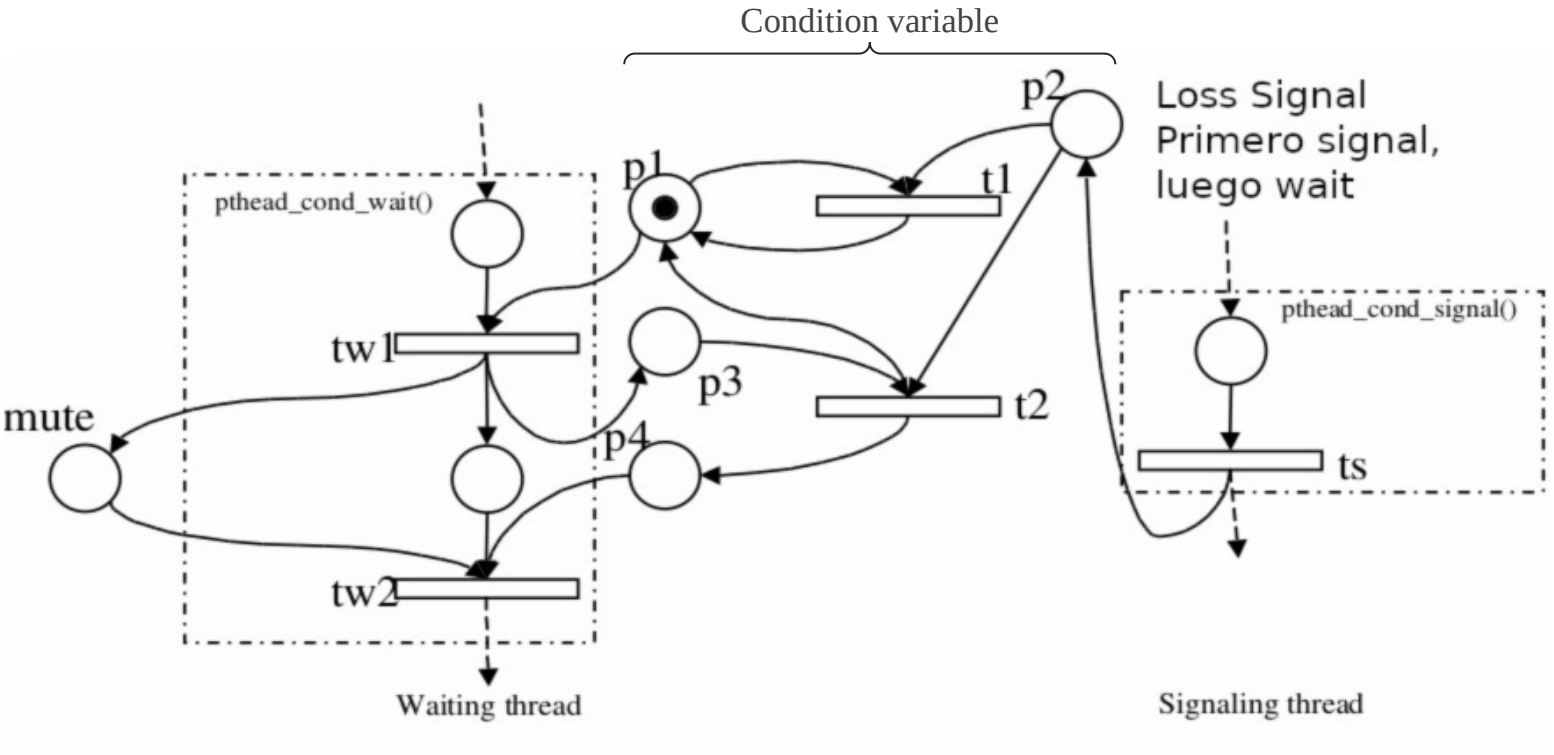
Ejemplo: detección de deadlock



Creación y Join de un thread

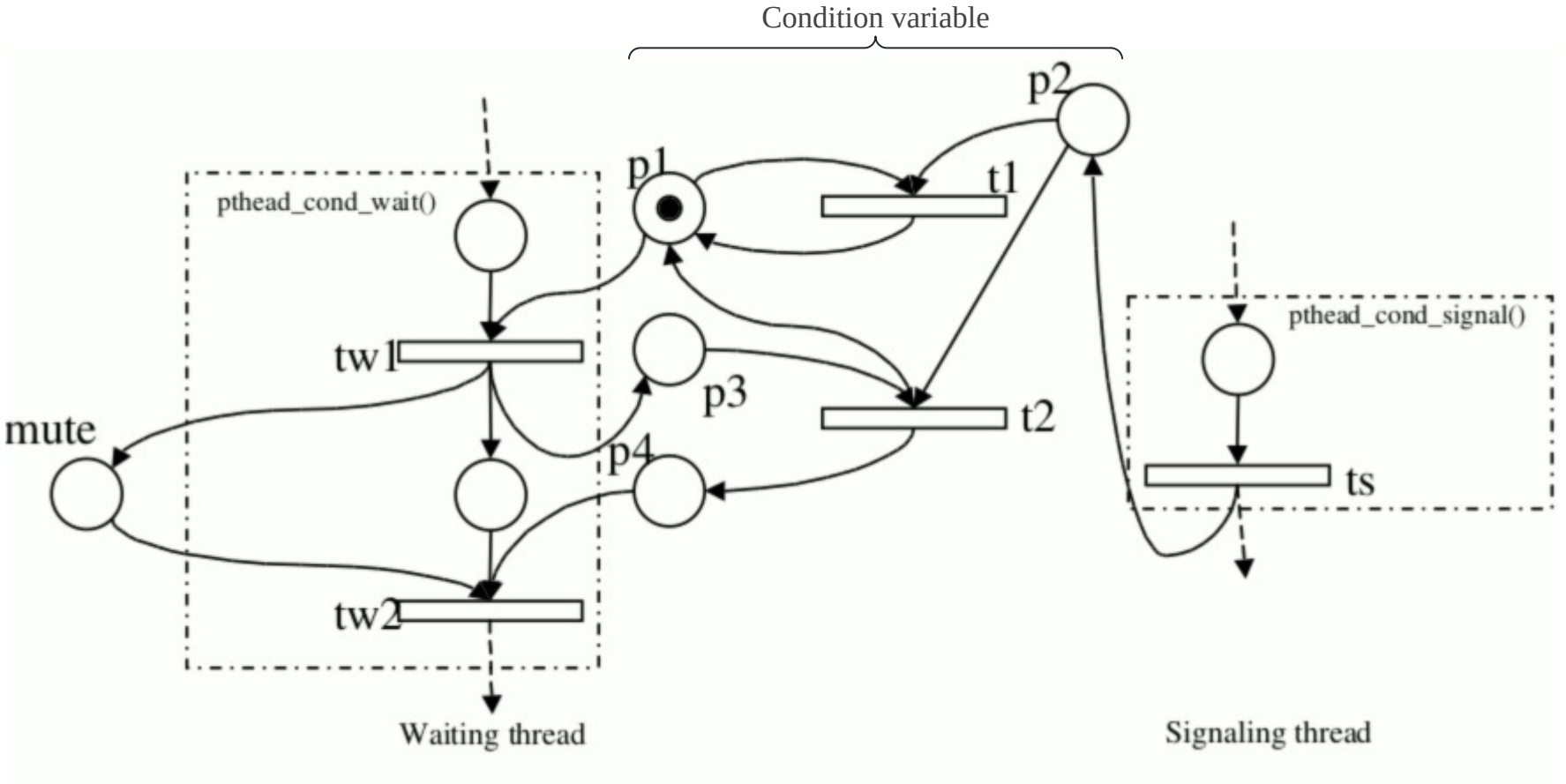


Signal & Wait - Caso Lost Signal



- Si se **puede llegar** a un **marking** (estado) donde el **programa termina sin** haberse **consumido el token p3**, **hay Lost Signal**

Signal & Wait - Caso sin pérdida de señal



¡Gracias!

¿Preguntas?