

# UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales



## Trabajo Práctico Final - "Red de Petri / Agencia de Viajes"

Programación Concurrente

**Grupo:** *Threading Bad*

- RODRIGUEZ, MATEO (43.967.398)
- TRACHTA, AGUSTIN (43.271.890)

# 1 Introducción

El presente informe detalla el trabajo realizado para abordar un problema de programación concurrente mediante el uso de redes de Petri y monitores. La programación concurrente se ha convertido en una disciplina esencial en el desarrollo de sistemas modernos, ya que permite aprovechar la capacidad de procesamiento de múltiples hilos de ejecución para mejorar la eficiencia y el rendimiento de las aplicaciones.

Sin embargo, la programación concurrente también presenta complejidades y desafíos únicos, como condiciones de carrera, bloqueos y problemas de sincronización. En este contexto, las redes de Petri emergen como una poderosa herramienta para modelar y analizar sistemas concurrentes, brindando una representación gráfica y formal de los estados y transiciones del sistema.

El objetivo principal de este informe es presentar el enfoque adoptado para resolver un problema específico de programación concurrente utilizando redes de Petri para modelar el sistema y monitores para gestionar la sincronización y el acceso seguro a los recursos compartidos. A lo largo del informe se hacen referencia a algunas definiciones que pasaremos a detallar ahora:

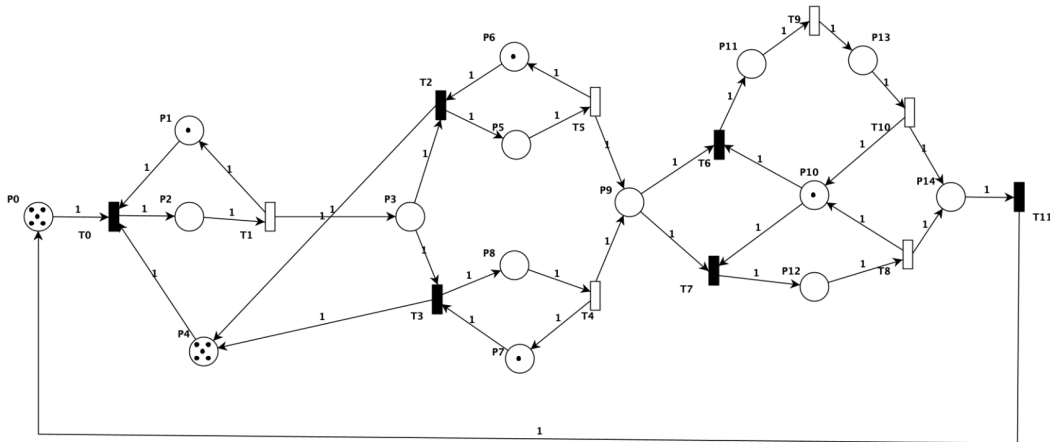
## 2 ¿Qué es una Red de Petri?

Una Red de Petri (RdP) es un modelo matemático y gráfico utilizado para describir y analizar sistemas concurrentes y distribuidos.

## 3 ¿Por qué utilizamos RdP?

Permiten modelar y visualizar casos de la vida real con paralelismo, concurrencia, sincronización e intercambio de recursos. Además, tienen un formalismo matemático y gráfico que nos permite determinar el disparo de una RdP y el siguiente marcado. Las RdP nos permiten separar la lógica de la política. La lógica es lo que puedo hacer, y la política es lo que me conviene hacer.

## 4 Propiedades de la Red



Las siguientes propiedades a mostrar son realizadas en PIPE

### Petri net state space analysis results

<b>Bounded</b>	true
<b>Safe</b>	false
<b>Deadlock</b>	false

#### 4.1 Red Limitada

Una red de Petri es limitada si existe un límite superior para el número de tokens que puede haber en cualquier lugar de la red en cualquier momento. Esto asegura que los recursos representados por los tokens no se desborden. En este ejemplo, se podría decir que la cantidad de clientes que pueden ingresar y ser atendidos es limitada, por cuestiones de espacio y trabajadores.

#### 4.2 Red Segura

Esta red no es segura ya que hay marcas que contienen más de un token, ya que pueden ingresar hasta 5 clientes a ser atendidos.

#### 4.3 Sin Deadlock

Un Deadlock ocurre cuando un sistema queda en un estado donde no puede proceder porque todos los procesos están esperando por algún evento que nunca ocurrirá. En una red de Petri, esto significa que ninguna transición puede dispararse porque no hay tokens suficientes en los lugares necesarios para habilitar alguna transición. En nuestro caso, al no tener Deadlock, siempre habrá al menos una transición que pueda dispararse.

### 5 Invariante de Transición

Es un vector conformado por números enteros asociados a una secuencia de disparos. Son útiles para determinar propiedades estructurales de una RdP en forma analítica. Un invariante de transición es el conjunto mínimo de transiciones que, cuando las dispare, vuelvo al estado inicial. Esto nos indica que algo se hizo. Si me fijo cuántos invariantes se completaron, voy a entender cuántos ciclos se completaron.

### Petri net invariant analysis results T-Invariants

T0	T1	T10	T11	T2	T3	T4	T5	T6	T7	T8	T9
1	1	0	1	0	1	1	0	0	1	1	0
1	1	1	1	0	1	1	0	1	0	0	1
1	1	0	1	1	0	0	1	0	1	1	0
1	1	1	1	1	0	0	1	1	0	0	1

*The net is covered by positive T-invariants, therefore it might be bounded and live*

## 6 Invariante de Plaza

Un invariante de plaza es el conjunto de plazas en donde la suma de sus tokens se mantiene constante a lo largo de todos los marcados de la red.

P-Invariants														
P0	P1	P10	P11	P12	P13	P14	P2	P3	P4	P5	P6	P7	P8	P9
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	0	1	0	0	1	1
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

*The net is covered by positive T-invariants, therefore it is bounded.*

### 6.1 Bounded and live

El programa quiere decir que, además de estar acotada (bounded), la red siempre tiene la posibilidad de disparar alguna transición (live), por lo que no entra en un estado de bloqueo definitivo.

### 6.2 Ecuaciones de Invariante de Plaza

A continuación, se listan las ecuaciones de invariante de plaza derivadas de la red:

$$M[P1] + M[P2] = 1$$

$$M[P10] + M[P11] + M[P12] + M[P13] = 1$$

$$M[P0] + M[P11] + M[P12] + M[P13] + M[P14] + M[P21] + M[P3] + M[P5] + M[P8] + M[P9] = 5$$

$$M[P2] + M[P3] + M[P4] = 5$$

$$M[P5] + M[P6] = 1$$

$$M[P7] + M[P8] = 1$$

## 7 Tabla de Estados en la Red de Petri

En la siguiente tabla se detallan las plazas de la red de Petri y el estado que representan:

Plaza	Estado
P0	Buffer de entrada de Clientes
P1	Recursos compartidos del sistema
P2	Cliente listo para entrar
P3	Cliente esperando a ser atendido
P4	Recursos compartidos del sistema
P5	Gestión de las Reservas
P6	Recursos compartidos del sistema
P7	Gestión de las Reservas
P8	Recursos compartidos del sistema
P9	Agente esperando a cancelar o confirmar reserva
P10	Recursos compartidos del sistema
P11	Agente por confirmar reserva
P12	Agente por cancelar reserva
P13	Agente pagando la reserva
P14	Cliente listo para retirarse

Table 1: Tabla de estados en la red de Petri.

## 8 Tabla de Eventos en la Red de Petri

A continuación se listan las transiciones y el evento asociado en la red de Petri:

Transiciones	Evento
T0	Input de Clientes
T1	Cliente entra al negocio
T2	Cliente es atendido
T3	Cliente es atendido
T4	Cliente pasa a la Confirmación
T5	Cliente pasa a la Confirmación
T6	Agente confirma la reserva
T7	Agente rechaza la reserva
T8	Cliente pasa a retirarse
T9	Se paga la reserva
T10	Cliente pasa a retirarse
T11	Cliente se retira de la agencia

Table 2: Tabla de eventos en la red de Petri.

## 9 Determinación de la cantidad de hilos

Para la obtención de la cantidad máxima de hilos activos simultáneamente, utilizamos un algoritmo basado en el procedimiento descrito en el paper del profesor. Dicho algoritmo se basa en el análisis de los invariantes de transición (IT) y el conjunto de plazas asociadas, eliminando aquellas que no corresponden a acciones activas en el sistema.

### 9.1 Obtener los IT de la RdP

Los invariantes de transición (IT) representan combinaciones de transiciones que pueden ejecutarse sin afectar el estado general del sistema. Son subconjuntos de transiciones que, al activarse en conjunto, llevan a un nuevo estado sin modificar la estructura global de la red de Petri.

Se han identificado los siguientes IT en nuestra red de Petri:

1.  $\{T0, T1, T3, T4, T7, T8, T11\}$
2.  $\{T0, T1, T3, T4, T6, T9, T10, T11\}$
3.  $\{T0, T1, T2, T5, T7, T8, T11\}$
4.  $\{T0, T1, T2, T5, T6, T9, T10, T11\}$

El análisis de estos ITs nos permite comprender cómo se estructura el sistema y qué partes de la red pueden ejecutarse en paralelo.

### 9.2 Obtener el conjunto de plazas asociadas al IT

Cada IT está relacionado con un conjunto específico de plazas dentro de la red de Petri. Estas plazas representan los estados alcanzables en la ejecución de las transiciones asociadas a cada IT. A continuación, se presentan las plazas detectadas para cada IT:

1.  $\{P0, P1, P2, P3, P4, P7, P8, P9, P10, P12, P14\}$
2.  $\{P0, P1, P2, P3, P4, P7, P8, P9, P10, P11, P13, P14\}$
3.  $\{P0, P1, P2, P3, P4, P5, P6, P9, P10, P12, P14\}$
4.  $\{P0, P1, P2, P3, P4, P5, P6, P9, P10, P11, P13, P14\}$

Este conjunto nos permite identificar todas las plazas que se activan en cada posible ejecución del sistema.

### 9.3 Determinar las plazas relacionadas a acciones de cada IT

Para determinar la cantidad máxima de hilos activos simultáneamente, es necesario filtrar las plazas que realmente representan **acciones activas** en el sistema. Según el procedimiento del profesor, eliminamos las plazas que corresponden a **restricciones**, **recursos** e **idle**, dejando únicamente aquellas plazas que intervienen en el flujo de ejecución.

El conjunto de plazas de acciones activas detectadas es:

1.  $\{P2, P8, P12\}$

2.  $\{P2, P8, P11, P13\}$
3.  $\{P2, P5, P12\}$
4.  $\{P2, P5, P11, P13\}$

Aquí se observa que las plazas de acción activas varían dependiendo del IT, lo que indica que la concurrencia puede cambiar según el flujo de ejecución que se tome.

## 9.4 Obtención del conjunto de estados del conjunto de plazas $PA$

Una vez identificadas las plazas de acción activas, el siguiente paso es agruparlas en un solo conjunto global  $PA$ . Este conjunto representa todas las plazas que están involucradas en la ejecución activa del sistema.

1.  $PA = \{P2, P5, P8, P11, P12, P13\}$

Para visualizar los posibles marcados en detalle, consulte la siguiente [Tabla con posibles marcados](#).

Este conjunto será utilizado en la siguiente etapa para calcular la cantidad máxima de hilos simultáneamente activos.

## 9.5 Suma de marcas y determinación de hilos simultáneos

El último paso consiste en analizar el **árbol de alcanzabilidad** de la red de Petri. Para ello:

1. Se obtiene el conjunto de todos los estados posibles de la red de Petri.
2. Para cada estado, se calcula la suma de marcas (tokens) en las plazas del conjunto  $PA$ .
3. Se selecciona el estado donde la suma de marcas es **máxima**.
4. Este valor máximo representa la **cantidad máxima de hilos simultáneos** en el sistema.

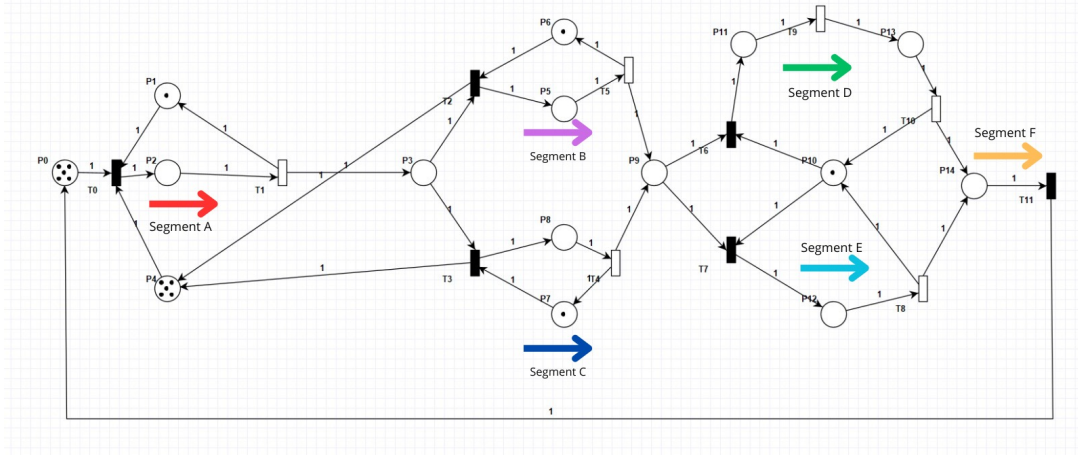
A partir de este procedimiento, se determinó que el valor máximo de marcas en la red es:

$$\text{Valor máximo de HILOS SIMULTÁNEOS} = 4$$

Este resultado nos indica que, en el punto de mayor concurrencia del sistema, pueden llegar a ejecutarse **hasta 4 hilos en simultáneo** de manera óptima.

## 10 Red de Petri final

La red de Petri final utilizada en el programa consta de 6 segmentos.



## 11 Justificación de la elección de los segmentos

La elección de los segmentos en la red de Petri se basó en la identificación de estructuras lineales, forks y joins en la red. Cada segmento se definió con el objetivo de maximizar la concurrencia y optimizar el uso de los hilos disponibles:

1. **Segmento A:** Es una secuencia de transiciones sin bifurcaciones, por lo que se ejecuta de manera secuencial sin interferencias.
2. **Segmentos B, C, D, E:** Estos segmentos contienen transiciones que generan caminos en paralelo (forks). Al dividir la ejecución en hilos independientes, se evita que un solo hilo tenga que decidir qué camino seguir, delegando la decisión a la política de ejecución del sistema.
3. **Segmento D:** Aunque es parte de un fork, en su parte final no comparte transiciones con otros segmentos, por lo que actúa también como un segmento lineal.
4. **Segmento F:** Representa el punto de convergencia de caminos en la red (join). Para evitar bloqueos, se permite la ejecución paralela de los segmentos anteriores, asegurando que todas las dependencias se resuelvan antes de continuar con la ejecución final.

## 12 Determinación de hilos máximos por segmento

A partir de la segmentación de la red de Petri y la identificación de los subconjuntos de plazas asociadas a cada segmento, se determina la cantidad máxima de hilos que pueden ejecutarse simultáneamente. La segmentación permite dividir la ejecución en partes independientes, asegurando que no haya interferencias en la ejecución concurrente.

Cada segmento tiene un conjunto de plazas asociadas, denominadas  $PS_i$ , que representan los estados alcanzables en dicho segmento. A continuación, se presentan los segmentos con sus respectivas plazas y la cantidad máxima de hilos necesarios para su ejecución:



- $PS_a = \{P2\} \Rightarrow \text{MAX} = 1$
- $PS_b = \{P5\} \Rightarrow \text{MAX} = 1$
- $PS_c = \{P8\} \Rightarrow \text{MAX} = 1$
- $PS_d = \{P11, P13\} \Rightarrow \text{MAX} = 1$
- $PS_e = \{P12\} \Rightarrow \text{MAX} = 1$
- $PS_f = \{P14\} \Rightarrow \text{MAX} = 1$

Dado que cada segmento requiere un hilo para su ejecución, la suma de los hilos necesarios en todos los segmentos nos da la cantidad total de hilos que el sistema puede requerir a lo largo de su ejecución:

$$\text{Hilos totales del sistema} = 1 + 1 + 1 + 1 + 1 + 1 = 6$$

Sin embargo, esto no implica que en todo momento haya 6 hilos ejecutándose simultáneamente. El análisis realizado mediante el primer algoritmo determinó que la **máxima cantidad de hilos activos simultáneamente** en la red de Petri es **4**. Esto significa que, en el punto de mayor concurrencia del sistema, pueden ejecutarse hasta 4 hilos en paralelo.

La segmentación aplicada permite distribuir la carga de trabajo de manera eficiente, asegurando que cada hilo ejecute su parte sin interferencias, optimizando el uso de los recursos y maximizando el rendimiento del sistema.