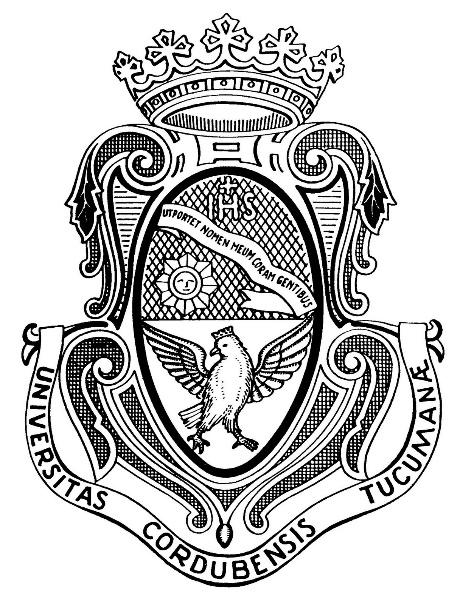
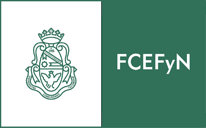
UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES





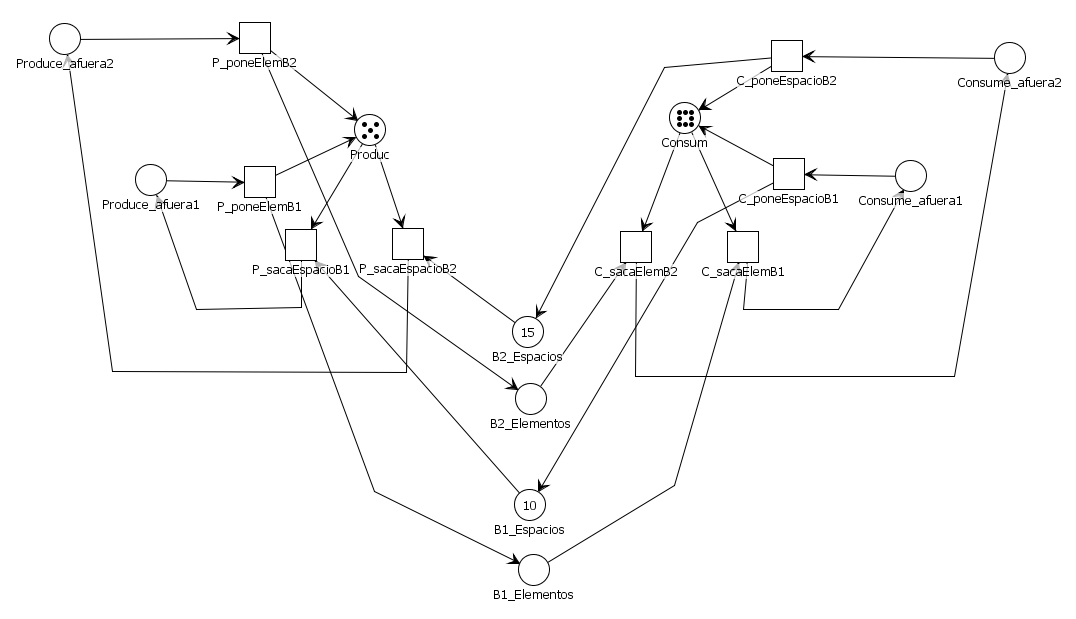
Trabajo Práctico Nº 2

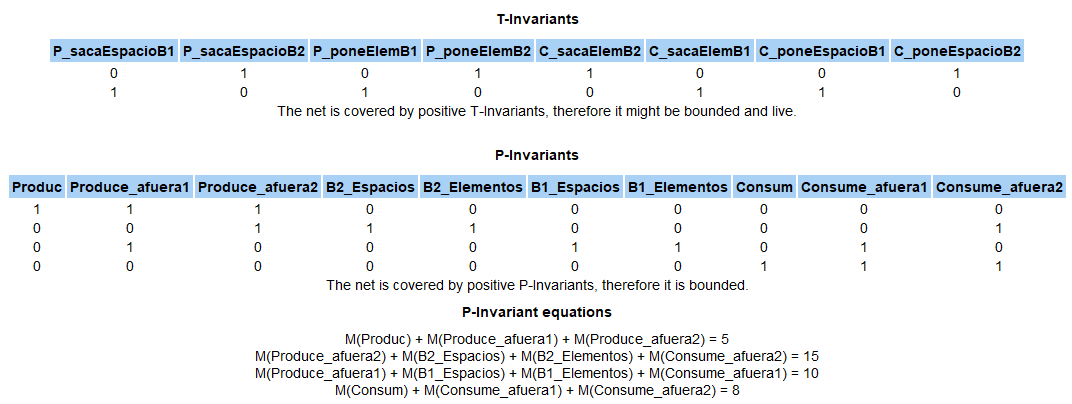
Materia: Programación Concurrente

Grupo: “ThreadRipper”

**Año 2019**

|  |  |
| --- | --- |
| **Alumnos** | **Matrícula** |
| **García,** Bruno Emilio | 39449179 |
| **Oroná,** Madelein Ayelen | 37852092 |
| **Losano Quintana,** Juan Cruz | 39621464 |
| **Vignolo**, Gabriel Enrique | 39080905 |

**Red de Petri:**

**Análisis de Invariantes:**

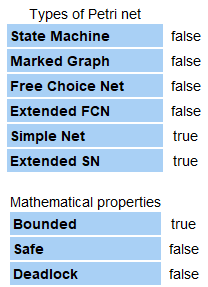
Las invariantes de transición y de plaza son útiles para probar propiedades dinámicas de la red, como vivacidad, inanición, alcanzabilidad, etc.

Por medio del análisis de los T-Invariantes, se pueden observar los circuitos/caminos que tendrá la RdP. Éstas indican posibles loops en la red. Y por medio de t-invariantes positivos, la RdP puede ser segura y viva.

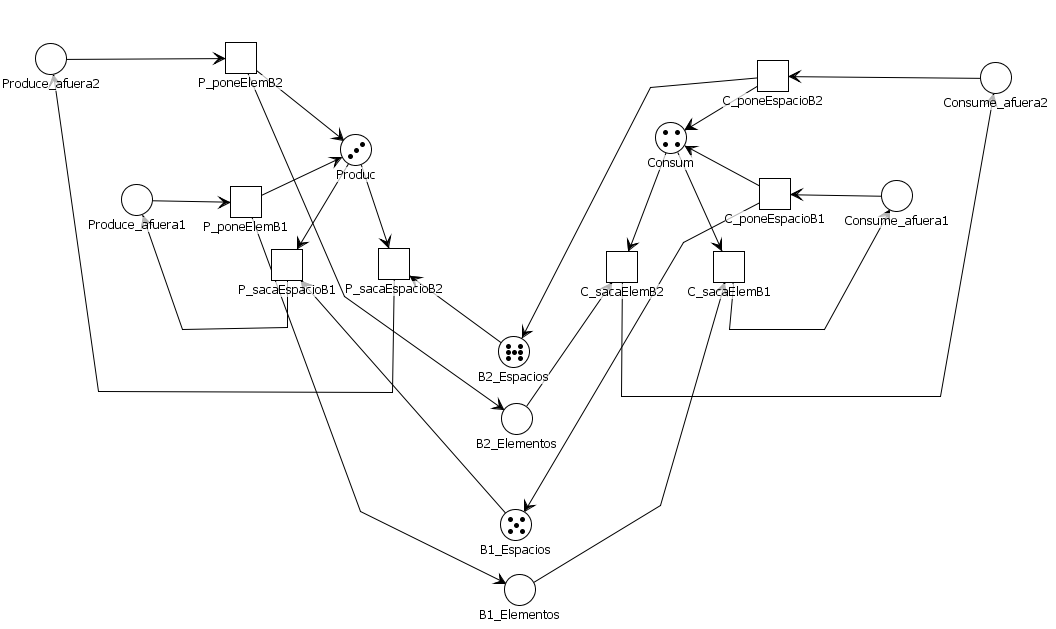
Por medio del análisis de los P-Invariantes, se puede advertir el número de tokens que se tendrá en todas las marcas alcanzables

**Propiedades de la red de Petri:**

Al analizar la red, empleando el software “Petrinator”, vemos que la misma tiene las siguientes propiedades de alcanzabilidad, seguridad e inanición:



Aclaración: Se modificó el marcado de la RdP para poder calcular estas propiedades (ya que la red era muy grande para el SW) Con un buffer1 de capacidad de 5, un buffer2 de capacidad 7, 3 productores y 4 consumidores (se trató de mantener aproximadamente la relación 1:2 de tokens que posee el anterior)



Red de Petri modificada para el cálculo de propiedades.

**Análisis de las clases:**

**Clase Main:** se inicializan los 2 buffers (uno de tamaño 10 y otro de tamaño 15), los productores y consumidores y los hilos para los productores y consumidores. Se hace start a dichos hilos para correr el programa. También se crea un objeto RdP y el monitor.

**Clase Producer:** es una clase que utiliza la interfaz Runnable y en su método run dispara la transición de la red de Petri a través del monitor, luego se duerme 50 ms simulando que tarda ese tiempo en producir el objeto y luego vuelve a disparar la transición de la red de Petri a través del monitor para colocar ese objeto producido. Si se ingresó al monitor con la transición 2 coloca en el buffer 1 y si se ingresó con la transición 3 en el buffer 2.

**Clase Consumer:** es una clase que también utiliza la interfaz Runnable y en su método run dispara la transición de la red de Petri a través del monitor, luego se duerme 50 ms simulando que tarda ese tiempo en consumir el objeto y luego vuelve a disparar la transición de la red de Petri a través del monitor para quitar ese objeto consumido. Si se ingresó al monitor con la transición 6 saca del buffer 1 y si se ingresó con la transición 7 saca del buffer 2.

**Clase Buffer:** en esta clase se utilizan los métodos set y get para poner y sacar cosas de un buffer.

**Clase RdP:** en esta clase se cargan en el programa la matriz de incidencia y la matriz de marcado de la red de Petri, hecha en el Petrinator. Se le asigna un marcado a cada plaza de la red en el método inicialización. Se realiza el método disparar el cual es utilizado por el método dispararTransción del monitor para ver si la transición se puede disparar o no, en caso de poderse disparar se devuelve true y si no se puede se devuelve false. También se realiza un método llamado sensibilizadas que se fija que transiciones están sensibilizadas en la red de Petri y crea el vector de sensibilizado utilizado por el monitor.

**Clase GestorDelMonitor:** en esta clase el productor y el consumidor van a intentar disparar la transición para realizar la tarea que corresponda. Esta clase va a tener una exclusión mutua ya que solo va a poder haber un hilo ejecutándose dentro del monitor.

**Clase Colas:** esta clase es utilizada por el GestorDelMonitor para ver quienes están en la cola en ese momento. También la utiliza para bloquear hilos en caso de que no se pueda disparar la transición y para despertarlos cuando sea necesario.