Facultad de Ingeniería
Escuela de Ingeniería Informática

CERTAMEN I

Por

Marcelo Leiton Valdés
Mauricio Moraga Michaud
Ignacio Villagran

Profesor: Alonso Inostrosa Psijas
Noviembre 2021

Resumen

En el siguiente informe se presenta cómo fue abordada la problemática presentada en el Certamen Nº1 realizado por estudiantes de la Universidad de Valparaíso en la asignatura ”Lenguajes de programación”, impartido por el profesor Alonso Inostrosa Psijas.

El trabajo consiste en implementar un lenguaje de programación que permita la creación y manipulación de redes de Petri. Para conceptualizar brevemente sobre el tema, las redes de Petri (PN, Petri Nets) fueron definidas por Carl Petri en 1962[1], quien fue un matemático y científico de la computación de nacionalidad Alemán. Estas corresponden a un tipo particular de grafo dirigido que, gracias a su representación matemática y gráfica, permite modelar sistemas concurrentes, paralelos y/o distribuidos([2]; [3]). En cuanto a lo solicitado en el Certamen I, el lenguaje debe contener instrucciones básicas que permitan la creación de elementos básicos para poder crear esta red. Las redes son creadas mediante places, arcos, transiciones y tokens los cuales se detallarán con mayor profundidad más adelante.

El lenguaje permite la creación de estos elementos básicos de forma independiente, además por medio de otras instrucciones definidas en el lenguaje debe ser posible la creación de redes de Petri de mayor complejidad, estas instrucciones serán especificadas en el apartado del diseño de la solución. Con esto para la implementación de estas redes se debe crear un lenguaje que permita crear redes mediante una cierta gramática definida y todo esto será realizado a través del uso de tecnologías ofrecidas por Flex, Bison y C. Flex permitirá la creación de analizadores léxicos, Bison es un programa generador de analizadores sintácticos y C es un lenguaje de programación de propósito general. Mediante el uso de estas tecnologías se desarrolló el Certamen I del curso de Lenguajes de Programación.

Índice general

Capítulo 1

Introducción

1.1. Resumen

El objetivo de este trabajo es presentar un lenguaje de programación que mediante expresiones regulares y gramáticas libres de contexto permitan la creación y también la manipulación de redes de Petri. Las redes de Petri fueron introducidas por A.C. Petri en 1962 para sincronizar la comunicación de autómatas, luego se ampliaron para definir un gran conjunto de modelos debido al aumento de complejidad y capacidades[4].
En esa época no fue tomado muy en cuenta el trabajo de Carl Petri, sino más bien fue una curiosidad, pero después comenzó a tomar más relevancia. Estas redes de petri corresponden a un tipo particular de grafos dirigidos que tiene una representación gráfica y matemática, permite modelar sistemas concurrentes, paralelos y/o distribuidos.

Una red de petri se compone principalmente de places, los cuales también se les llama lugares o nodos, transiciones que se activan o se disparan frente a la ocurrencia de algún evento y arcos que conectan un nodo o un place con una transición y una transición luego con un place o nodo y también tokens. Los tokens son elementos que pueden ir transitando de un place a otro a través de estas transiciones.

Una red de Petri se define como una 5-Tupla, es decir una tupla de 5 componente donde está definida como N = (P,T,F,W,Mo) en donde P es el conjunto finito de places o de nodos, en caso de tener noción de autómatas serían estados, T es un conjunto finito de transiciones, F es un conjunto finitos de arcos que conecta un place con una transición o una transición con un place, W es una función de peso, es decir los arcos pueden tener su peso el cual es un número entero positivo, si un arco no tiene definido su peso tiene un valor 1 y Mo es un marcado inicial es decir la cantidad de tokens que existen en los places al comienzo.

Estas redes tienen un potencial para poder representar diferentes tipos de sistemas, en particular sistemas concurrentes paralelos o distribuidos. Con las redes de petri se pueden llegar a simular unas simples redes tales como una acción en que dos entidades deben coordinarse, o una

acción que desencadena dos actividades diferentes hasta una acción que permita realizar redes complejas como lo puede ser el problema de exclusión mutua con una red de Petri. Cabe destacar que, en las redes de petri, no existen los estados iniciales o finales como se puede conocer en la teoría de autómatas, aquí se conoce como marcado inicial el cual es como se asigna tokens a los diferentes places.

En el trabajo a realizar se solicita la manipulación de redes de Petri el cual debe contener instrucciones básicas que permita la creación de los elementos descritos anteriormente, los que son posibles de crear de forma independiente, es decir, solo puede crear places, transiciones o arcos y en cuanto al arco, este debe indicar el inicio y el final, además del peso. Así también, se debe crear dos instrucciones que permitan el desarrollo de dos redes complejas que utilizen algún tipo de parámetro, al momento de crear los elementos estos también deben estar conectados. Además, se debe realizar una red de ejemplo la cual será detallada en los puntos a continuación. La forma en la cual se debe implementar la red debe ser mediante el uso de Flex, el cual se utiliza como un creador del analizador léxico, Bison que es un programa generador de analizadores sintácticos y para esto se hará uso del lenguaje c el cual es un lenguaje de programación de propósito general.

Capítulo 2

Marco Conceptual

2.1. Marco Conceptual

Primero debemos poner en contexto ciertos términos que son relevantes a la hora de comprender esta temática. Lo más importante es entender el concepto relacionado con el planteamiento de una problemática, pero, ¿qué es un problema?. En base a una de las definiciones de la Real Academia Española (RAE) y que resulta más apropiada para este contexto es que un problema es un planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos. En este trabajo abordaremos la problemática la cual es la implementación de un lenguaje que reconozca ciertas gramáticas permitiendo la creación de una red de Petri.

Lenguajes y su historia:

Se define un lenguaje de programación como un lenguaje formal, es decir, que contiene ciertas reglas gramaticales bien definidas que le proporcionan a las personas. Para este caso, en particular programadores, la capacidad para escribir una serie de instrucciones de órdenes con la finalidad de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o la ejecución de diversas tareas. [5][6][7][8]

El primer lenguaje de programación nació en 1801, con Joseph Marie Jacquard quien fue el inventor del telar programable. Él creó el primer sistema de instrucciones para un ”computador” dando inicios a los primeros pasos de los lenguajes de programación. Este fue el creador de lo que se conoce como el primer telar programable con tarjetas perforadas conocido como el telar de Jacquard presentado por primera vez en una exhibición industrial de Lyon en 1805. Consiste en que estas tarjetas se introducían en un telar que leía el código, las instrucciones de dichas tarjetas y automatizaba los procesos. Se estima que este fue el primer lenguaje de programación, dado que estas tarjetas perforadas se utilizaron a futuro en las primeras máquinas computacionales creadas por Charles Babbage, y posteriormente, por los primeros ordenadores.

Históricamente con el avance generado por estos grandes inventores, comenzaron a surgir

más lenguajes de programación. Con ello podemos hablar de uno de los padres de la computación Alan Turing, quien revolucionó el mundo del cómputo con su Máquina de Turing. Esta máquina demostraba que por medio de algoritmos se podría resolver cualquier problema matemático. Desde entonces surgió una revolución en el mundo de la computación e informática y gracias a todos estos grandes inventores podemos llegar a lo que hoy conocemos, obviamente el proceso histórico fue muy resumido dado que fueron muchos años de avance, muchos nombres a los cuales agradecer, pero en breve resumen el lenguaje de programación permitió agilizar estos procesos y permitir que más gente pueda entender sobre como darle instrucciones a un ordenador.

Carl Petri y sus redes:

Se hablaba anteriormente como surgieron los lenguajes de programación, dado que lo solicitado para esta entrega es la implementación de un lenguaje que permita la manipulación de redes de Petri. Ahora bien, ¿Quién fue Carl Petri?. Fue un matemático y científico de la computación alemán. Se considera que fue uno de los descubridores en los años 1960 de las denominadas Redes de Petri. Las Redes de Petri surgen en 1962 con el trabajo doctoral de Carl Adam Petri ”Kommunikation mit Automaten” (Comunicación con autómatas), en Alemania. En su disertación doctoral Petri formuló la base para una teoría de comunicación entre componentes asíncronos de un sistema de cómputo.

Definición de red de Petri:

Una Red de Petri es un modelo gráfico, formal y abstracto para describir y analizar el flujo de información. El análisis de las Redes de Petri ayuda a mostrar información importante sobre la estructura y el comportamiento dinámico de los sistemas modelados. Las Redes de Petri pueden considerarse como autómatas formales o como generadores de lenguajes formales y tienen asociación con la teoría de grafos. Son excelentes para representar procesos concurrentes, así como, procesos donde pueden existir restricciones sobre la concurrencia, precedencia, o frecuencia de esas ocurrencias.[9]
Una red de petri se compone de nodos (también llamados Places), transiciones (que reflejan la ocurrencia de acciones o eventos), arcos (o aristas) y tokens.

Capítulo 3

Definición del Problema

3.1. Formulación del Problema

El problema nace en base la propuesta definida en el certamen 1 del curso de lenguajes de programación por el profesor Alonso Inostrosa, la cual consistía en realizar un lenguaje que permita la creación y manipulación de redes de Petri, también conocidas como PN (Petri nets). Estas redes son una representación matemática, gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente. El lenguaje a implementar debe permitir la creación de estas redes por medio de instrucciones, estas instrucciones permitirán crear simples redes de Petri hasta redes mucho más complejas. Estas instrucciones llamadas complejas, se deben definir para que realicen una acción mediante la formalidad de sintaxis, esta sintaxis debe permitir realizar un sistema el cual puede parecer complejo de definir con unas instrucciones específicas para ello simplificando el problema o haciéndolo más sencillo para definir. Esto se debe trabajar con el uso de tecnologías que permiten la creación del lenguaje como lo son Flex, Bison y en este caso utilizamos el lenguaje C.

3.2. Solución Propuesta

Como se comentaba en el anterior enunciado, se hace uso de tecnologías tales como Flex, Bison y C. Se implementará un lenguaje en base un programa que actúa como un escáner o que genera un analizador léxico[1][2]. Sus siglas describen que es un generador/analizador de léxicos rápido(fast lexical analyzer generator), este programa es de libre uso que es una alternativa de otro programa llamado lex[4]. Se hará uso de Bison, el cual es un programa generador de analizadores sintácticos de propósito general perteneciente al proyecto GNU disponible para prácticamente todos los sistemas operativos, se usa normalmente acompañado de flex, esta información es obtenida del sitio oficial del fabricante, la función de un analizador sintáctico es analizar una cadena de símbolos según las reglas de una gramática formal, usualmente hace uso de un compilador, en cuyo caso, transforma una entrada en un árbol sintáctico de derivación[4][5]. Como se comentaba estos dos programas de complementan uno al otro, por eso se hará uso de estos dos combinados, por una parte se genera la gramática el analizador léxico y el analizador sintáctico se encarga de obtener

la gramática libre de contexto en un programa en C para describir la gramática. Se hará uso del lenguaje de programación C para este caso particular, dado que Bison lo permite y como grupo se definió trabajar en él por afinidad. Adentrándonos en el tema principal el cual es permitir la implementación de la red de Petri, se crearán instrucciones las cuales serán definidas de tal forma que sea simple de entender y siempre pensando de la forma que sea lo más parecido a como es definir una red de Petri. Para definir la red se debe ingresar por consola o terminal estas entradas que acepta el lenguaje, en caso de no ser correctas el programa debe reconocer esto y no realizar la red si no se define bien. Para definir la red se detalla una serie de instrucciones la cual desencadenará una acción, esto será definido en el apartado del diseño de la solución.

3.3. Objetivos

3.3.1. Objetivo General

Implementar un lenguaje que permita la implementación de unas redes en particular llamadas redes de Petri. El lenguaje debe ser capaz de recibir una o unas entradas que serán instrucciones para la creación de la red. Debe contener instrucciones que permitan la creación desde una simple red hasta una red con una complejidad mayor. Se reconoce para una red simple la relación de elementos básicos de la red, tales como places, arcos, transiciones y tokens de forma independiente. En cuanto a las redes complejas, el lenguaje debe permitir la creación de al menos dos problemas complejos de redes de Petri mediante el uso de instrucciones aceptadas por el lenguaje. Finalmente se debe establecer una acción que permita saber si establecidos dos places cualquiera, exista un camino el cual los une.

3.3.2. Objetivos Específicos

- Dejar como parametro el uso de un token al requerir tanto una transaccion como un arco.
- Crear dos instrucciones que permita crear algún tipo de redes complejas.
- Al utilizar el lenguaje creado, se debera contruir y explicar una red de ejemplo.

Capítulo 4

Diseño de la solución

4.1. Solución del problema

Para realizar la solución tuvimos que abordar el uso de estructuras, el deber de las estructuras dentro del código c es para que puedan almacenar varias variables y además la dirección siguiente para conectar con varias estructuras, ejemplo un place que a través de un arco pasa a una transición.

Para poder crear las estructuras en esta instancia son necesarios los lenguajes léxicos lo cuales en este caso fueron flex y bison para realizar la actividad, primero bison genera los analizadores sintácticos y después lex genera los analizadores léxicos, el lenguaje de programación que se decidió ocupar fue C como lenguaje base de programación.

Lo que hace C dentro del lenguaje de programación funciona como la definición de las estructuras para poder generar estas estructuras.

Para poder crear el lenguaje de programación que permita la creación y manipulación de PN o mejor llamadas Petri nets para la representación gráfica de un sistema a eventos discretos se debe contener instrucciones básicas que permitan la creación de elementos básicos, volviendo a las estructuras estas son las responsables de estas instrucciones simples, primero el place representa gráficamente un círculo, la transición se representa con un rectángulo y el arco como la flecha que apunta desde el place a una transición, dentro un place pueden haber tokens el cual es un valor dependiendo de como funciona el modelo que se va a representar.

Capítulo 5

Experimentación

5.1. Ejemplos de ejecución

Para los ejemplos de ejecución se definirán los siguientes:
Ejemplos de redes:

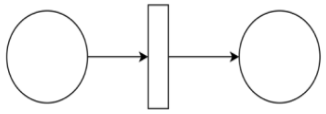
Para la creación de un place, se debe iniciar con la letra P, luego se debe asignar un número identificador del place, mediante corchetes se puede ingresar un nombre de place, y seguido de paréntesis la cantidad de token que pueda tener, su estructura es la siguiente P#[NOMBRE](#) .

Para la creación de una Transacción se debe iniciar con la letra T, luego se debe asignar un número identificador y luego un nombre el cual puede o no estar definido, su estructura sería la siguiente T#[NOMBRE].

Para la creación de un arco debe haber una conexión entre un Place a una Transición, o al reves de una Transición a un Place como por ejemplo ->[#](P#,T#), la cual indica un arco, que tiene entre corchetes un peso, y dentro del paréntesis un place con su identificador y una transición con su identificador. otro ejemplo es el siguiente ->[#](T#,P#).

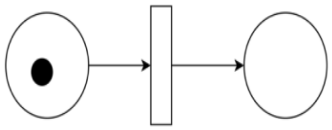
Creación	**Gramática**	**Ejemplo**	**Observación**
Place	P#	P1	Place independiente sin nombre
Place	P#(#)	P1(1)	Place independiente con token
Place	P#[Nombre](#)	P1[NOMBRE](1)	Place con nombre y token
Transición	T#[NOMBRE]	T1[NOMBRE]	Transición independiente
Arco entre Place y Transición	->(P#,T#)	->(P1,T1)	Arco sin peso inicio P1 y destino T1
Arco entre Transición y Place	->(T#,P#)	->(T1,P1)	Arco sin peso inicio T1 y destino P1
Arco entre Place y Transición	->[#](P#,T#)	->[1](P1,T1)	Arco con peso inicio P1 y destino T1
Arco entre Transición y Place	->[#](T#,P#)	->[1](T1,P1)	Arco con peso inicio T1 y destino P1

Para la creación de una red sencilla debe ejecutar el siguiente comando: P1->T1->P2



(a)

Si desea insertar tokens a algún place debe ingresar el siguiente comando para ingresar una red del tipo como se muestra a continuación:P1(1)->T1->P2



(b)

Se recomienda para la creación de estas redes cerrar el programa y volver a ejecutarlo para no interferir con la nueva red que se desea crear.

Ejemplo de Representación del problema de exclusión mutua:
Para realizar esta red debe ingresar la siguiente instrucción en la consola:
MOSTRAR RED E/M
Con esto se creará la siguiente red.

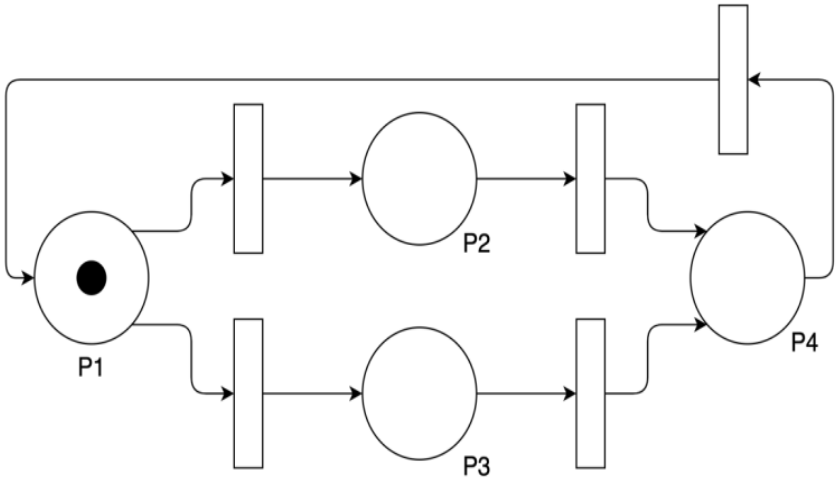


Figura 5 - Representación del problema de exclusión mutua con una PN.

Para crear la red Productor/Consumidor debe ejecutar el siguiente comando en la consola: MOSTRAR RED P/C

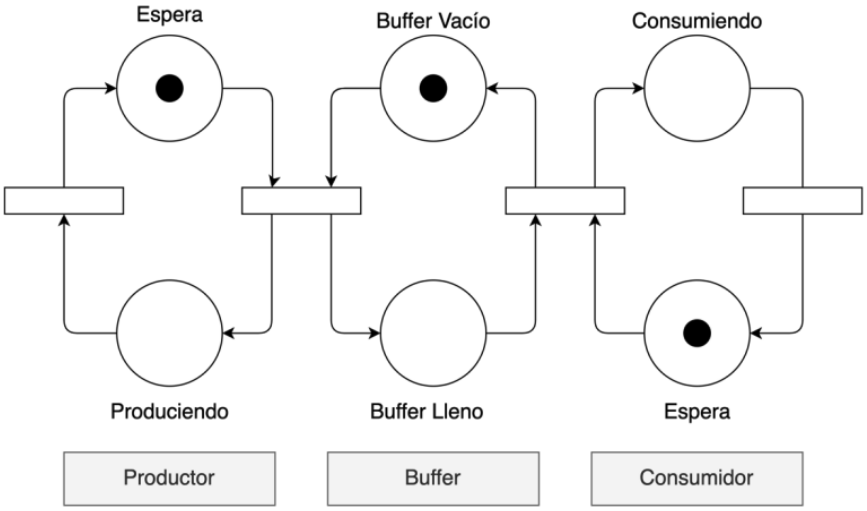


Figura 6 - Solución al problema del "Productor/Consumidor" con una PN.

en cuanto a la función para mostrar si, dados dos places cualquiera, existe un camino (independiente de la cantidad de tokens requeridos) que los una topológicamente, deberá ingresar por consola el siguiente comando: MOSTRAR CAMINOS (PLACE1,PLACE2)

Capítulo 6

Implementación

6.1. Software utilizado

Los softwares utilizados para este proyecto fueron lex y bison para la creación de este lenguaje de programación, en el caso de yacc es un programa para generar analizadores sintácticos y para lex es un programa para generar analizadores léxicos.

6.2. Hardware utilizado

En el caso del Hardware las herramientas que se llegaron a utilizar para realizar el proyecto fueron como principipal el acceso a un computador para poder correr el lenguaje, y como el lenguaje de programación no es demandante no es necesario de un computador de gran potencia.

6.3. Lenguajes de programación

6.3.1. Requerimientos

Requerimientos de Mínimos

Para los requerimientos mínimos un computador tiene que tener los siguientes componentes: Se recomienda como mínimo un Pentium 4, a 1 GHz para un sistema de escritorio y para este trabajo se utilizó un sistema operativo Linux con una distribución de Debian 11.

Tipo de instalación	RAM (mínimo)	Disco duro
Con escritorio	256 Megabytes	10 Gigabyte

Requerimientos de Recomendados

Para los requerimientos recomendados de un computador tiene que tener los siguientes componentes:

Se recomienda como recomendado un Intel i3, a 3 GHz para un sistema de escritorio y para este trabajo se utilizó un sistema operativo Linux con una distribución de Debian 11.

Tipo de instalación	**RAM (Recomendado)**	**Disco duro**
Con escritorio	4 Gigabyte	50 Gigabyte

6.3.2. Preparación de Ambiente

Se debe instalar Flex y Bison dentro del sistema operativo para poder compilar y ejecutar el lenguaje de programación, para facilitar la ejecución es recomendado utilizar un sistema operativo unix, una vez instalados estos programas es necesario un editor de texto si le combiene en este caso se usará visual studio code y una terminal o cosola.

6.3.3. Compilación

La forma en que se compilara el programa será a través de la terminal Se debe proceder a la ejecución del programa un archivo Makefile será creado con el único propósito de crear y compilar los archivos correspondientes de igual manera, cada vez que se ejecute el programa se elimina el cache del anterior ejecutado. En la terminal una vez ubicado en la carpeta deberá ejecutar: make.

6.3.4. Ejecución

Para la ejecución se debe hacer uso del archivo Makefile, el cual contiene todos los comandos necesarios para la compilación del programa.
En la consola o terminal, solo deberá inicializar el comando make y el programa será compilado, posteriormente debe ingresar el comando ./certamen.exe y podrá ingresar la gramática.

6.3.5. Manual de Usuario

Pasos a seguir para compilar y ejecutar el programa:

Como el archivo Makefile ya debería estar creado dentro del archivo el único paso a seguir es ejecutar un ”certamen.exe”, lo que hará este será compilar y ejecutar el programa, si fue anteriormente ejecutado se limpiarán los archivos que se crearon con el comando clean.

En caso de no tener make instalado deberá instalarlo en base al sistema operativo que esté usando, para Linux es sudo apt-get install make.

Capítulo 7

Conclusiones

7.1. Conclusiones

En base al trabajo realizado, es importante desarrollar lenguajes de programación que nos permitan realizar tareas en base a la gramática que éste acepte, en nuestro caso se propone implementar un lenguaje de programación que por medio de una gramática definida acepte la creación de redes de Petri. En el desarrollo de procesos de automatización, los lenguajes utilizados tienen como fundamento o base las redes de Petri, tales como lo son las máquinas de estados finitos.
Las redes de petri pueden ser aplicadas para la modelación de sistemas de eventos discretos, estas ofrecen una forma de representación matemática y gráfica de modelos de sistemas, nuestro modelo y gramática permite la creación de redes de Petri simples hasta redes de Petri complejas tal como lo es el problema de exclusión mutua, el problema del productor/consumidor etc...Con esto destacar la importancia de las herramientas que nos permiten realizar el certamen propuesto como lo es Flex, Bison y C, los cuales permiten la implementación de un lenguaje que reconoce ciertas instrucciones definidas y la creación de la red.

Bibliografía

[1] C. A. Petri, *Comunicación con máquinas.*, 1962.

[2] Murata, *Petri nets: Properties, analysis and applications.*, 1989, vol. 77.

[3] J. Lobos, *Aplicación de Redes dePetri al Dimensionamiento de Servicios en Motores de Búsqueda Vertical para la Web.*, 2012.

[4] M. Diaz, *Petri Nets: Fundamental Models, Verification and Applications (1.a ed.). Wiley-Iste.*, 2008.

[5] *Tecnicos de Soporte Informatico de la Comunidad de Castilla Y Leon. Temario Volumen i Ebook. MAD-Eduforma.* Editorial MAD, 20006.

[6] M. Juganaru Mathieu, *Introducción a la programación.*, 2014.

[7] L. H. Yáñez, *Fundamentos de la programación*, 2013-2014.

[8] L. Joyanes Aguilar, *FUNDAMENTOS DE PROGRAMACIÓN Algoritmos, estructura de datos y objetos. McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.*, 2008.

[9] P. s. f. Mayorga, "Fundamentos de las redes de petri." [Online]. Available: http://personal.cimat.mx:8181/~mayorga/cursos/docs/Simulacion_Fundamentos_Redes_Petri.pdf