Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

(7415) - PROGRAMACIÓN CONCURRENTE

Informe de Trabajo Final

ESTUDIANTE:

VALENZUELA, Gabriel Emanuel Matrícula:39.934.209

gabrielvalenzuela12@gmail.com

Profesores:

Dr. Ing. Orlando Micolini

Ing. Orlando Ventre

19 de febrero de 2019

Índice

1. Enunciado

1.1. Introducción

Las **redes de petri** son una herramienta para el estudio de sistemas. La teoría de las Petri Net permite al sistema ser modelado por una Petri Net (PN a partir de ahora), una represetnación matemática del sistema. El análisis de la PN puede entonces, humildemente, revelar información importante acerca de la entructura y el comportamiento dinámico del sistema modelado. Esta, puede ser entonces usada para evaluar el modelado del sistema y sugerir cambios o inconvenientes.

Por lo tanto, el desarrollo de una teoría de PN está basado en la aplicación de la PN en el modelado y diseño de sistemas.

Las PN están diseñadas específicamente para modelar aspectos importantes de un sistema, e.g. los componentes del mismo (subsistemas), el estado de los subsistemas, la concurrencia o paralelismo de los mismos, la sincronización, etc. Fueron desarrolladas por el trabajo de **Carl Adam Petri (1962)** en su disertación doctoral "Kommunikation mit Automaten"(Comunicación con autómata); Petri formuló las bases para una teoría de comunicación entre componentes asíncronos de un sistema de computación, haciendo énfasis en la descripción de relaciones causales entre eventos.

El trabajo de Petri llamó la atención de **A. W. Holt** entre otros miembros de la Information System Theory Project of Applied Data Research Inc. (ADR) además de la Project MAC del MIT, entre otros. El uso y estudio de la PN se ha extendido rápidamente en los últimos años.

1.2. Modelado

La apliación de las PN es a través del *modelado*. En muchos campos de estudio, un fenómenos no es estudiado directamente sino indirectamente a través de un modelo del mismo. Un modelo, es una representación, generalmente en términos matemáticos, de cuáles son los aspectos importantes del objeto o sistema bajo estudio. Por la manipulación de la representación, es de esperarse que el nuevo conocimiento acerca del fenómeno modelado puede ser obtenida sin daño, costo o inconveniente de manipular el fenómeno real en sí.

Gran parte de los modelos usan la matemática, los aspectos importantes de muchos fenómenos físicos puede ser descriptos numéricamente y la relaciones entre estos por ecuaciones o inecuaciones. Particularmente, en las ciencias naturales y la ingeniería, propiedades como la masa, posición, momento, aceleración y fuerzas pueden ser descriptas por ecuaciones matemáticas. Para obtener una aproximación exitosa del modelo, resulta necesario un conocimiento tanto del modelado del fenómenos como las propieades de la técnica de modelado.

1.3. Apliación de la Teoría de PN

El estudio de las PN se ha desarrollado en dos direcciones:

- Teoría de la PN aplicada - Centrada con la aplicación de las PN al modelado de sistemas, el análisis de tales sistemas, y las ideas resultantes del modelado de sistemas.
- Teoría pura de PN - Relacionada con el estudio de PN para el desarrollo de las herramientas bases, técnicas y conceptos necesarios para la aplicación.

1.4. Definición de Red de Petri

Basaremos nuestro formalismo a través de la teoría de bolsa o multiconjunto (bag theory) que es lo mismo que una estructura de datos de conjunto, pero que admite valores repetidos; las definiciones dadas son de estilo similar a la definiciones de la teoría autónoma (Hopcroft y Uliman 1969). De hecho, ellos definen una nueva clase de maquinas, las PN autómata que como veremos más adelante, esta idea, puede derivar en algunos resultados interesantes en la teoría del lenguaje formal y la teoría de autómata.

Definición 1.1 *Una red de Petri está compuesto por cuatro partes: Un conjunto de plazas* $\boldsymbol{P}$*, un conjunto de transiciones* $\boldsymbol{T}$*, un conjunto de funciones de entrada* $\boldsymbol{I}$ *o* I^+ *y un conjunto de funciones de salida* $\boldsymbol{O}$ *o* I^- *ambas de incidencia respecto a las plazas y relacionan tanto a las plazas como a las transiciones. De manera matemática, una PN* $\boldsymbol{C}$ *es una 4-upla:*

$$C = (P, T, I, O) \tag{1.1}$$

Con:

$$P = \{p_1, p_2, \ldots, p_n\}\ n \geq 0 \tag{1.2}$$

$$T = \{t_1, t_2, \ldots, t_m\}\ m \geq 0 \tag{1.3}$$

$$P \cap T = \emptyset \tag{1.4}$$

$$I : T \to P^\infty \tag{1.5}$$

$$O : T \to P^\infty \tag{1.6}$$

Que $P \cap T = \emptyset$ quiere decir que **P** y **T** son disjuntos. La cardinalidad de P es **n** y la de T es **m**. Denotaremos de manera arbitraria de ahora es mas como p_i y t_j.
Una plaza p_i es una plaza de **entrada** (input) de una transición t_j si $p_i \in I(t_j)$; y es una plaza de **salida** (output) cuando $p_i \in O(t_j)$. La entradas y salidas de una transición son multiset o un bag de plazas, i.e. las entradas y salidas de una transición permite a una plaza tener múltiples entradas o múltiples salidas. La multiplicidad de una plaza de entrada p_i para una transición t_j es el número de incidencias de una plaza en la bag de entrada de la transición denotada como $\#(p_i, I(t_j))$. De manera similar, la multiplicidad de una plaza de salida p_i para una transición

t_j es el número de incidencias de la plaza en la bag de salida de la transición $\#(p_i, O(t_j))$.
Si las funciones de entrada y salida son conjuntos (En vez de bags), entonces la multiplicidad de cada plaza es cero o uno. Estas funciones pueden ser útiles para extender el mapeo de plazas en las bolsas (bags) de transiciones además del mapeo de transiciones en las bolsas (bags) de plazas, i.e. definimos una transición t_j como una entrada de p_i si p_i es una salida de t_j de manera similar, una transición t_j es salida de una plaza p_i si p_i es una entrada de t_j.

Definición 1.2 *Entendemos la función de entrada* ***I*** *y de salida* ***O*** *como sigue:*

$$I : P \to T^{\infty}$$
$$O : P \to T^{\infty}$$

Tal que:

$$\#(t_j, I(p_i)) = \#(p_i, O(t_j))$$
$$\#(t_j, O(p_i)) = \#(p_i, I(t_j))$$

1.5. Grafos de Redes de Petri

Gran parte del trabajo teórico en las PN está basado en la efinició formal de la estructura de PN que daremos, sin embargo, una representación **gráfica** de una estructura de PN es muchas veces más útil para ilustrar la teórica. Un grafo de PN es una representación de una estrucutra de PN como un **multigrafo directo bipartito**. Una estrucutura de PN consiste de plazas y transiciones, de acuerdo a esto el grafo de una PN posee dos tipos de nodos. Un círculo que repreesenta una plaza y una bara que representa una transición. Arcos directos (flechas) conectan las plazas y transiciones, algunos desde las plazas a las transiciones y otros en sentido opuesto; un arco directo desde una plaza p_i a una transición t_j define la plaza como la entrada de la transición.
Múltiples entradas a una transición son indicadas por múltiples arcos, una plaza de salida está indicada por un arco desde la transición a una plaza, y nuevamente la representación múltiples en análoga a la anterior. Esto es la idea de **multigrafo**, una PN permite multiples arcos desde un nodo del grafo a otro, y dado que los arcos son directos se dice que es una **multigrafo directo**. Puesto además que los nodos se pueden agrupar en dos conjuntos (plazas y transiciones), de tal forma que cada arco es directo desde algún elemento de un conjunto a otro (ya sea plaza o transición) se tiene un **multigrafo directo bipartito**. Ahora podemos dar una definición más formal:

Definición 1.3 *Un grafo de Red de Petri* ***G*** *es un multigrafo directo bipartito,* $G = (V, A)$ *donde* $V = \{v_1 \dots v_n\}$ *es el conjunto de vértices y* $A = \{a_1 \dots a_n\}$ *el multiconjunto o bolsa de*

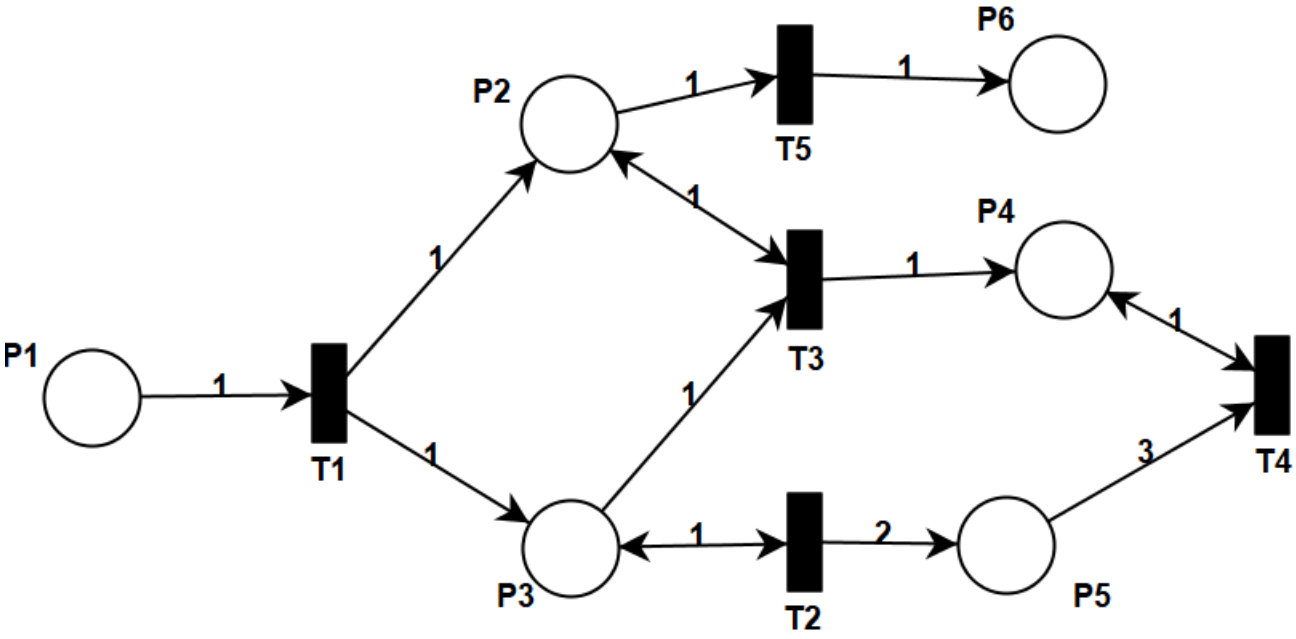


Figura 1.1: Grafo de la red de Petri C

arcos, $a_i = (v_j, v_k)$ *con* $v_j v_k \in V$.
El conjunto V *puede ser particionado en dos conjuntos disjuntos* P *y* T *tal que* $V = T \cup P$ *y para cada arco directo* $a_i \in A$, *si* $a_i = (v_j, v_k)$ *puede pasar que* $v_j \in P$ *y* $v_k \in T$ *o viceversa.*

Nota: Una representación alternativa de los arcos es el uso de **arcos agrupados**, los cuales difieren de los comunes ya que son más gruesos y rotulados con su multiplicidad.
Ejemplo: Llevar la PN C definida continuación de un grafo G.

$$C = (P, T, I, O)$$
$$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$$
$$T = \{t_1, t_2, t_3, t_4, t_5\}$$

$$\begin{aligned} I(t_1) &= \{p_1\} & O(t_1) &= \{p_2, p_3\} \\ I(t_2) &= \{p_3\} & O(t_2) &= \{p_3, p_5, p_5\} \\ I(t_3) &= \{p_2, p_3\} & O(t_3) &= \{p_2, p_4\} \\ I(t_4) &= \{p_4, p_5, p_5, p_5\} & O(t_4) &= \{p_4\} \\ I(t_5) &= \{p_2\} & O(t_1) &= \{p_6\} \end{aligned}$$

1.6. Redes de Petri marcadas

Una marca μ es un asignamiento de **tokens** a las plazas de una PN. Un token es un cocepto primitivo para una PN (Como las plazas y transiciones), estos son asignados a, y pueden residir

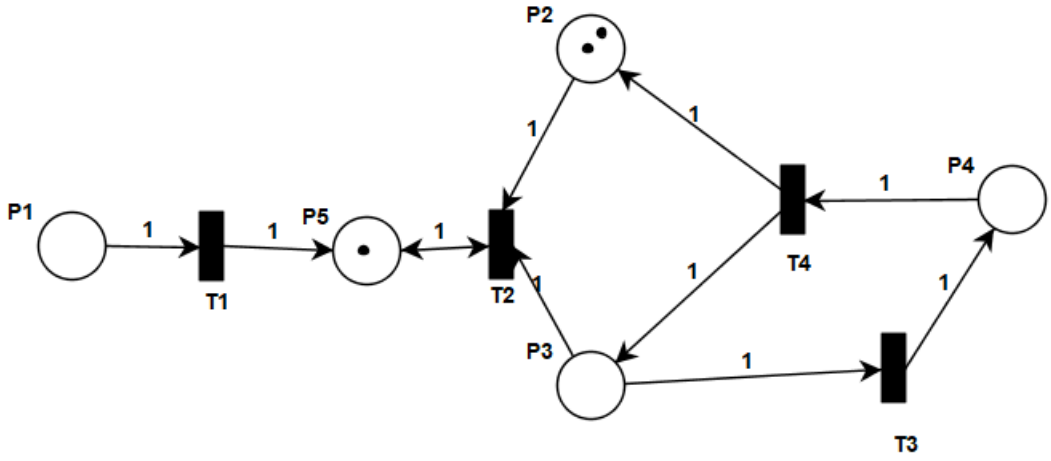


Figura 1.2: Red de Petri marcada $\mu = (0, 2, 0, 0, 1)$

en, las plazas de una PN. El número y posición de los tokens puede variar durante la **ejecuación** de una PN. Los tokes son usados para definir la ejecución de una PN.

Definición 1.4 *Una marca μ de una red de Petri $C = (P, T, I, O)$ es una función de un conjunto de plazas P a un entero no negativo, i.e.*

$$\mu : P \to N$$

La marca μ puede también definida como un n-vector $\mu = (\mu_1, \ldots, \mu_n)$, donde $n = |P|$ y cada $\mu_i \in N$, $i = 1, \ldots, n$. El vector μ nos da para cada plaza p_i en una Pn el número de tokens en esa plaza.
El número de tokens en una plaza p_i es μ_i. La definiciones de una marca como función y como vector puede reducirse a $\mu(p_i) = \mu_i$. La notación funcional es de alguna forma más general y por tanto más comoda de usar. Una PN marcada $M = (C, \mu)$ es una estructura de PN $C = (P, T, I, O)$ y una marcado μ. Esto suele expesarse como:

$$M = (P, T, I, O, \mu) \tag{1.7}$$

En un grafo de PN, los tokens son representados por puntos pequeños $\bullet$ en los círculos los cuales representan las plazas de una PN. Dado que el número de tokens los cuáles pueden ser asignados a una plaza de una PN es no acotado, existen una infinita cantidad de marcas para una PN. El conjunto de todas las marcas de una PN con n plazas es simplemente el conjunto de n-vectores, N^n. Este conjunto, aunque es infinito, es por su puesto numerable.

1.7. Ejecución de una red de Petri

La **ejecución** de una PN es contralada por el número y distribución de los tokens en la PN. Los tokens residen en la plazas y controlan la ejecución de la transición de la red. Una PN ejecuta por **disparo de las transiciones**.

Una transición al **dispararse** remueve los tokens desde sus plazas de entradas y creando nuevos tokes los cuales son distribuidos a sus plazas de salidas. Cabe aclarar que una transición puede ser disparada si está **habilitada**, i.e. si cada una de sus entradas poseen al menos tantos tokes como los indicados por el arco desde la plaza a la transición. Los tokens en las plazas de entrada los cuales habilitan una transición son los **tokens habilitantes**. Por ejemplo, si las entradas de una transición t_j son las plazas p_i, p_k y p_k la transición estará habilitada si al menos hay un token en p_i y al menos dos tokens en p_k.

Definición 1.5 *Una transición $t_j \in T$ una una red de Petri marcada $C = (P, T, I, O)$ con una marca μ está habilidad si para todo $p_i \in P$ se cumple*

$$\mu(p_i) \geq \#(p_i, I(t_j)) \tag{1.8}$$

Como dijimos, al dispararse una transición se remueven todos los tokens habilitantes de las plazas de entrada y se depositan en cada una de las plazas de salida, tantos tokens como los indicados por los arcos. Disparar una transición en general cambiará la marca μ de la PN a una nueva μ'. Notemos que puesto que sólo las transiciones habilitadas pueden ser disparadas, el número de tokens en cada plaza siempre permanece no negativo cuando se dispara la transición, i.e. si **no** hay tokes suficientes en alguna plaza de entrada de la transición, entonces la transición no está habilitada y no puede ser disparada.

Definición 1.6 *Una transición t_j es una red marcada de Petri con una marca μ puede ser dispara únicamente cuando está habilitada. Dispara una transición habilitada t_j resulta en una nueva marca μ' definida como:*

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)) \tag{1.9}$$

Por ejemplo, consideremos la siguiente PN: Luego de ejecutar los disparos correspondientes, se

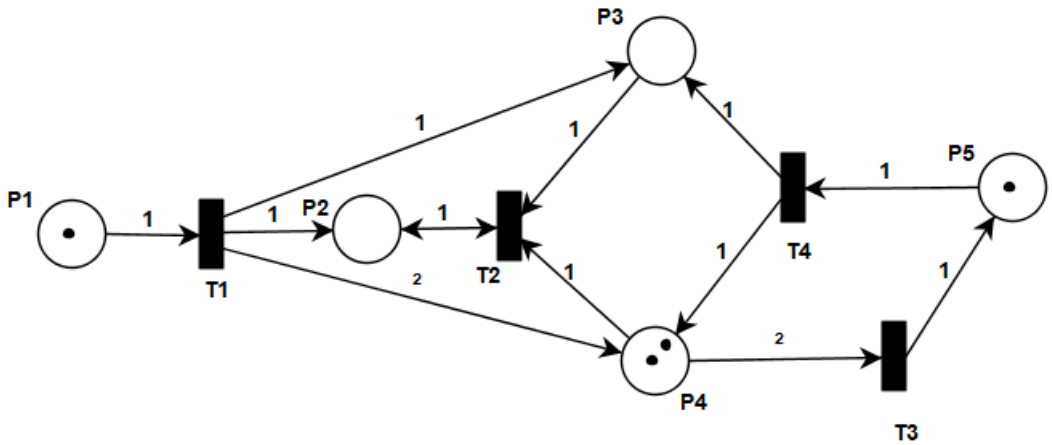


Figura 1.3: Red de Petri marcada

llaga a:

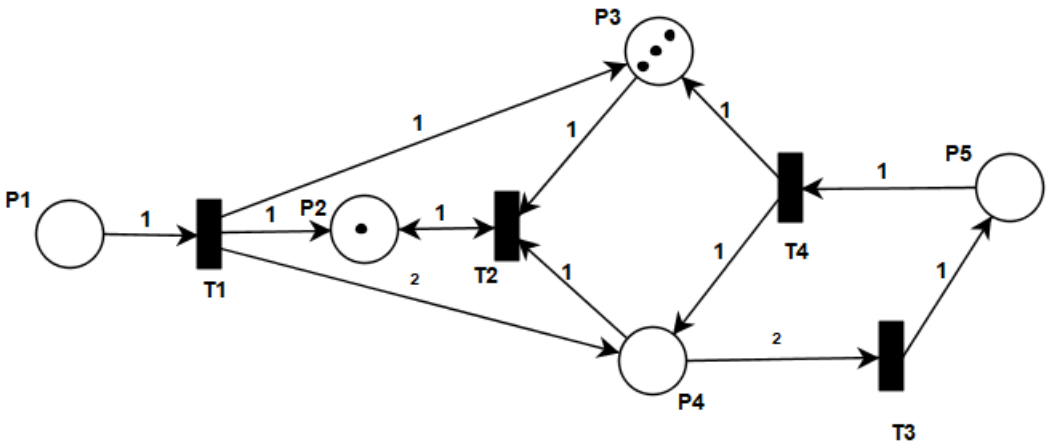


Figura 1.4: Red de Petri final

1.8. Estado de las plazas de una red de Petri

El **estado** o **condición** de una PN está definida por su marca. El disparo de una transición representa un cambio en el estado de una PN por el cambio de su marca. El estado de las plazas o estadio espacial de una Pn con n plazas es el conjunto de todas las marcas, esto es, N^n. El cambio en el estado causado por el disparo de una transición está definido por un cambio de la función δ denominada **función de estado próximo**. Cuando se aplica a una marca (estado) μ y una transición t_j produce una nueva marca (estado) el cual resulta del disparo de la transición t_j en la marca μ. Puesto que t_j puede ser disparada si está habilitada, $\delta(\mu, t_j)$ está **no definida** si t_j no está habilitada en la marca μ. Si t_j está habilitada se tiene:

$$\delta(\mu, t_j) = \mu'$$

Donde μ' es la marca la cual resulta de remover los tokens desde las entradas de t_j y añadiendo los tokens a la salida de t_j.

Definición 1.7 *La función de estado próximo* $\delta : N^n \times T \to N^n$ *para una red de Petri* $C = (P, T, I, O)$ *con marca* μ *y transición* $t_j \in T$ *está definida si, y solo si*

$$\mu(p_i) \geq \#(p_i, I(t_j)) \tag{1.10}$$

para todo $p_i \in P$. *Si* δ *está definida, entonces:*

$$\delta(\mu, t_j) = \mu' = \mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)) \tag{1.11}$$

oara todo $p_i \in P$

Dada una PN $C = (P, T, I, O)$ y una transición inicial μ_0, podemos ejecutar la PN por sucesivos disparos de las transiciones. Al disparar todas las plazas habilitadas, si llegamos a una transición en la cual no está habilitada, entonces no se pueden disparar las transiciones, la función de estado

próximo está indefinida para todas las transiciones y la ejecución debe detener. La **secuencia de marcas** $(\mu_0, \mu_1, \dots)$ y la **secuencia de transiciones** las cuales fueron disparadas $(t_{j_0}, t_{j_1}, \dots)$ las cuales están relacionadas por:

$$\delta(\mu_k, t_{j_k}) = \mu_{k+1} \tag{1.12}$$

con $k = 0, 1, 2, \dots$. Dada una secuencia de transición y μ_0, podemos fácilmente derivar la secuencia de marcado para la ejecución de la PN, y, excepto para unos pocos casos aislados dada una secuencia de marcas, podemos derivar la secuencia de transición. Ambas secuencia por lo tanto, proveen un registro de la ejecución de la PN.
En una marca μ, un conjunto de transiciones será habilitada y posiblemente disparada, dando como resultado una nueva marca μ' donde diremos que es **factible inmediatamente** de μ, i.e. podemos inmediatamente conseguir el estado μ' del estado μ.

Definición 1.8 *El conjunto de factabilidad $R(C, \mu)$ de una red de Petri $C = (P, T, I, O)$ con marca μ es el conjunto más pequeño de marcas definido por:*

1. $\mu \in R(C, \mu)$
2. *SI* $\mu' \in R(C, \mu)$ *y* $\mu'' = \delta(\mu', t_j)$ *para alguna* $t_j \in T$ *entonces,* $\mu'' \in R(C, \mu)$

Hablaremos más de la factibilidad de una PN más adelante. Por el momento es conveniente extender la función de estado siguiente para mapear una marca y una secuencia de transiciones en una nueva marca.

Definición 1.9 *La función de estado siguiente extendida está definida para una marca μ y una secuencia de transiciones $\sigma \in T$ donde:*

$$\delta(\mu, t_j\sigma) = \delta(\delta(\mu, t_j)\sigma) \tag{1.13}$$

$$\delta(\mu, \lambda) = \mu \tag{1.14}$$

Nota: Algunas literaturas hacen referencia al conjunto de factibilidad de una PN marcada como su clase marcada. Más específicamente, la clase marcada futura es la que definimos como factibilidad. La clase pasa es el conjunto de factibilidad de la PN inversa (Plazas por transiciones, enroque).
La teoría de la PN como hemos dicho, no fue desarrollada por una sola persona, e.g la teoría original de **Petri** de 1962 no permitía múltiples entradas y salidas, se limitaba a una relación binaria. El trabajo de **Holt y Commoner** presentado en 1970 (Wood Hole conference) generalizaba la clase de marcado y la regla de disparo permitiendo marcas arbitrarias, esta definió la base del modelo PN actualmete (con la excepción de múltiples arcos). Una de las primeras definiciones formales fue dada por **Patil** en 1870 durante su disertación de Ph.D, donde definió la PN como una cuatroupla (T,P,A,B) donde T es el conjunto de transiciones, P el de plaas, A el de arcos y B el de las marcas iniciales. **Hack** en 1975 eventualmente asentó la definición

de PN como una cuatroupla (P,T,F,B) donde nuevamente P es el conjunto de plazas y T el de transiciones, pero F y B son funciones que mapean las plazas y transiciones en el número de tokens necesarios para la entrada (F) producidas por la salida (B). Esto es, una transición t_j puede ser disparada sólamente si al menos hay $F(t_j, p_i)$ tokens en la plaza $p_i \in P$. Un disparo de la transición remueve $F(t_j, p_i)$ tokens de cada entrada y deposita $B(t_j, p_i)$ tokens en cada plaza de salida. Las funciones F y B pueden ser representadas por matrices.
Las diferencias en la definiciones es estrictamente de notación, sin ambargo, en algunos casos las definiciones pueden restringir la clase de PN no permitiendo las múltiples entradas y salidas o restringiendo la forma de las transiciones, e.g. algunas transiciones son requeridas para tener un conjunto no nulo de plazas de salida y de entradas, o la entrada y salida de una transición deben ser disjuntas (libre de autobucles).

2. Modelado con redes de Petri

Como dijimos antes, las PN fueron diseñadas y son usadas principalmente para ***modelar***. Muchos sistemas, especialmente aquellos con componentes independientes, pueden ser modelados por una PN. Veremos varios ejemplos de diversos tipos de sistemas los cuales han sido modelado por PN.

2.1. Eventos y condiciones

La visión de la PN se simplifica en dos conceptos primitivos: eventos y condiciones. Los **eventos** son accione las cuales tomar lugar en el sistema, la ocurrencia de estos está controlada por el estado del sistema el cual puede ser descripto por un conjunto de condiciones. Una **condición** puede mantenerse (ser verdad) o no (ser falsa).
Puesto que los eventos son acciones, estos *pueden* ocurrir. Para que un evento ocurra, puede ser necesaria ciertas condiciones a mantenerse; estas se denominan **precondiciones** y la ocurrencia del evento puede causar que estas dejen de mantenerse y causar otras condiciones, las **postcondiciones** se hacen verdaderas.

2.1.1. Ejemplo: Taller de maquinaria

Consideremos cierto taller que posee tres máquinas distintas M_1, M_2 y M_3 y dos operarios O_1 y O_2. El operario O_1 puede operar las maquinas M_1 y M_2 mientras que O_2 pude operar las máquinas M_1 y M_3. Para operar, se requieren de dos estados de macanización. Primero debe ser mecanizado por la máquina M_1 y entonces por la máquina M_2 o bien por M_3. Se tienen por tanto, las siguientes condiciones:

a. Una orden llega y espera por el mecanizado de M_1.

b. Una orden ha sido procesada por M_1 y espera por ser procesada por M_2 o M_3.

c. La orden se completa.

d. M_1 está desocupada.

e. M_2 está desocupada.

f. M_3 está desocupada.

g. O_1 está desocupado.

h. O_2 está desocupado.

i. M_1 es operada por O_1.

j. M_1 es operada por O_2.

k. M_2 es operada por O_1.

l. M_3 es operada por O_2.

Además, los siguientes eventos pueden ocurrir:

1. Llega una orden.
2. El operador O_1 comienza la orden en la máquina M_1.
3. El operador O_1 finaliza la orden en la máquina M_1.
4. El operador O_2 comienza la orden en la máquina M_1.
5. El operador O_2 comienza la orden en la máquina M_1.
6. El operador O_1 comienza la orden en la máquina M_2.
7. El operador O_1 finaliza la orden en la máquina M_2.
8. El operador O_2 comienza la orden en la máquina M_3.
9. El operador O_1 finaliza la orden en la máquina M_3.
10. La orden es despachada.

Armando la tabla de precondiciones y postcondiciones, tenemos:

Evento	Precondiciones	Postcondiciones
1.	Ninguna	a
2.	a,g,d	i
3.	i	g,d,b
4.	a,h,d	j
5.	j	b,h,d
6.	b,g,e	k
7.	k	c,g,e
8.	b,f,h	l
9.	l	c,f,h
10.	c	Ninguna

La PN correspondiente es:

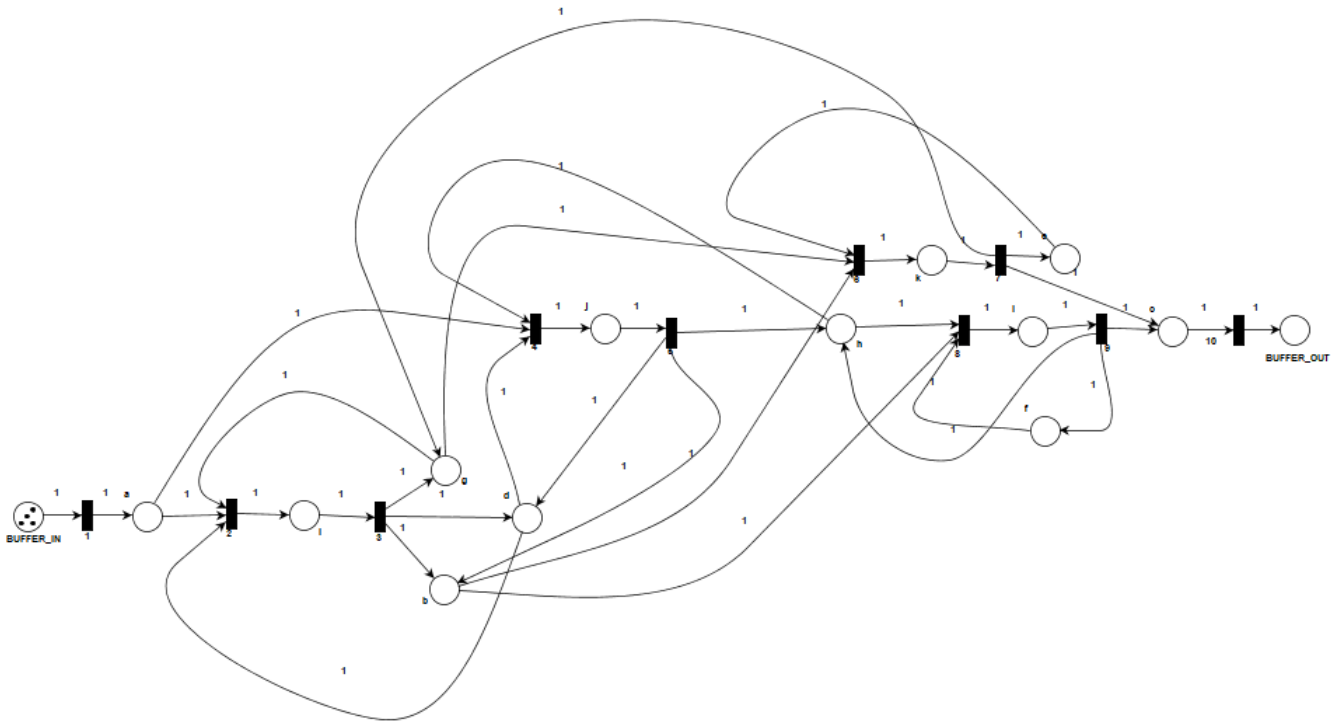


Figura 2.1: Red de Petri proceso de mecanizado

Podemos ver, en una mirada mas detalla el proceso de mecanizado:

De manera análoga, se puede dibujar un sistema de computación el cual proceso trabajos desde un dispositivo de entrada y produce las salidas correspondientes a un dispositivo de salida.

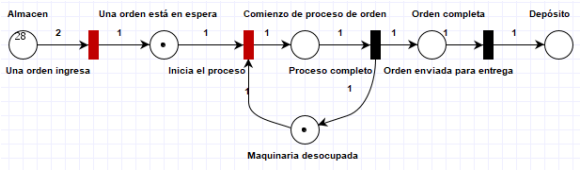


Figura 2.2: 1

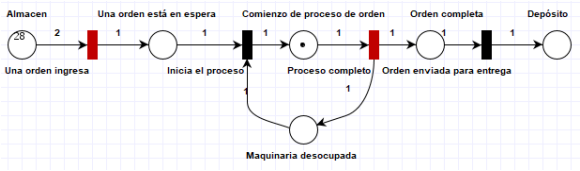


Figura 2.3: 2

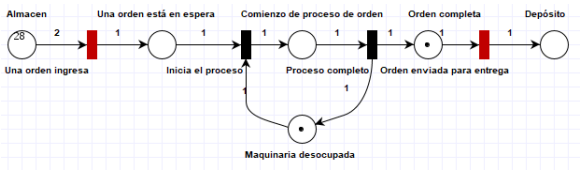


Figura 2.4: 3

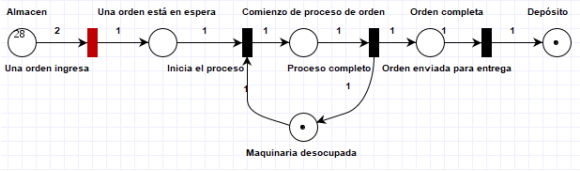


Figura 2.5: 4

2.2. Redes de Petri Especiales

Ciertos tipos de Redes de Petri tienen estructuras particulares, i.e. poseen características y propiedades que no poseen la mayoría de las redes. Todas las redes presentadas a continuación, poseen características estructurales, i.e. relacionadas con redes de Petri no marcadas (No existen tokens).

2.2.1. Grafo de estado

Una red de Petri no marcada es un grafo de estado si, y solo si, toda transición tiene exactamente una entrada y una salida. De esto podemos observar que, si tenemos un grafo de estado y lo marcados, su comportamiento será equivalente al no marcado (Representando un autómata el cual posee un solo estado a la vez) si y solo si contiene exactamente un solo token. En los grafos de estado los pesos de los arcos son 1.

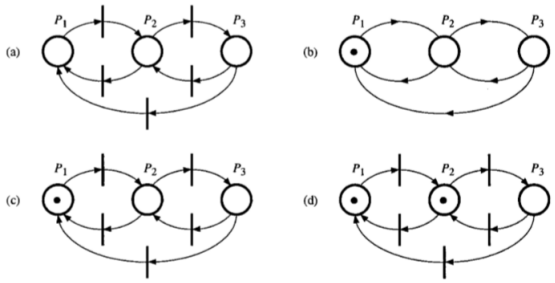


Figura 2.6: (a) No marcado. (b) Representación de un grafo de estado en el sentido clásico. (c) Marco con **un** token. (d) Marcado con varios tokens

2.2.2. Grafo de evento

Una red de Petri es un grafo de evento si, y solo si, toda plaza tiene exactamente una entrada y una salida, se lo suele denominar también grafo de transición o grafo marcado. Podemos ver que un grafo de evento es por tanto el clon de un grafo de estado.

2.2.3. Red libre de conflicto

Se denomina así a la Re de Petri en la cual toda plaza posee a lo sumo una transición de salida. Recordemos que un conflicto, mas precisamente un conflicto estructura, se corresponde a la existencia de una plaza la cual tiene como mínimo dos transiciones de salida o más. Un conflicto estructura se denota por: $< P, T_j >$ con $j = 1, ..., n$

2.2.4. Red de Petri de elección libre

Aquí encontramos dos definiciones. Una Rede de Petri de elección libra es una red en la cual para todo conflicto $< P, T_j >$ ninguna de las transiciones posee otra plaza de entrada que no sea P, una extendida, todas las transiciones tienen el mismo conjunto de plazas de entrada, provocando que si una transición involucrada en un conflicto esta habilitada, todas los están.

2.2.5. Red de Petri simple

Es aquella en la cual cada transición puede solamente ser concernida por un conflicto a lo sumo. En otras palabras, si una transición T y dos conflictos existen entonces la red no es simple. El conjunto de grafos de estado esta incluido en el conjunto de las redes de elección libre, estas son a su vez subconjuntos de las redes de Petri ordinarias i.e. redes de Petri marcadas autómatas funcionando acorde las reglas antes planteadas. Una red de Petri se dice que es pura, si no tiene auto-loops. Nota adicional, en un grafo de estado puede haber conflictos, pero no hay sincronización (i.e. una transición con al menos dos entradas). En un grafo de evento, hay muchas sincronizaciones, pero no hay conflicto.

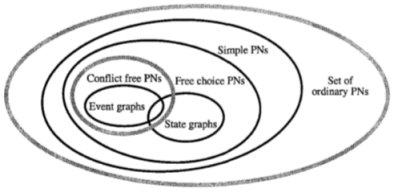


Figura 2.7: Relaciones a lo largo de los diferentes subconjuntos de Redes de Petri ordinarias

2.2.6. Red de Petri extendidas

Una red de Petri extendida contiene arcos especiales denominado como **inhibidor**. Un arco inhibidor es un arco directo el cual va desde una plaza P_i a una transición T_j y al final del

mismo es marcado por un pequeño circulo blanco. La transición T_j estará habiitada si la plaza P_i no contiene ningún token.

2.3. Redes prioritarias

Este tipo de redes es usado cuando deseamos elegir entre un número de transiciones habilitadas: De acuerdo a la figura siguiente, presenta dos conflictos estructurales $< P1, T_1, T_2 >$, $< P2, T_2, T_3, T_4 >$. Un orden estricto para las correspondientes transiciones es asignado para cada conflicto. La relación $T_j < T_k$ significa que T_j posee mayor prioridad sobre T_k, si ambas están habilitadas.

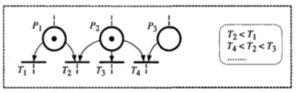


Figura 2.8: Parte de una red prioritaria. T_2 posee mayor jerarquía que el resto

Un ejemplo de aplicación es el siguiente:

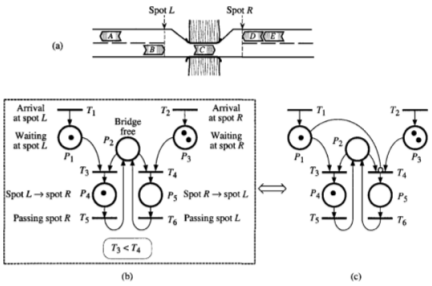


Figura 2.9: (a) Modelo de un cruce por un puente. (b) Red de Petri prioritaria. (c) Equivalencia con red extendida.

2.4. Redes de Petri no autónomas

Son aquellas en las cuales están sincronizadas y/o temporalizadas. Está claro que una Red de Petri no autónoma no puede convertirse en una Red de Petro ordinaria.

3. Propiedades principales de una Red de Petri

3.1. Red acotada

Una plaza P_i se dice que es acotada para un marcado inicial m_0 si existe algún entero $k > 0$ tal que,para todos los marcados alcanzables para m_0, el número de tokens en $P_i < k$ (P_i se dice que está k-acotada). Si todas las plazas son acotadas, la red es k-acotada.

3.2. Red segura

Una Red de Petri se dice que es segura para un marcado inicial m_0 si para todos los marcados alcanzables, cada plaza contiene un cero o un token. Una red segura es por lo tanto un caso particular de una red acotada,donde todas las plazas son 1-acotadas.
Nota: Las propiedades de seguridad y acotación dependen del marcado inicial m_0.

3.3. Liveness & Deadlock

El marcado de una Red de Petri evoluciona con el disparo de las transiciones. Cuando ciertas transiciones no son disparables y cuando todo o una parte de la red ya no "funciona"; es probable que haya un problema en el diseño del sistema.
Una transición T_j está **viva** par un marcado inicial m_0 si para todo marcado alcanzable $m_i \in \mathcal{M}(m_0)$ existe una secuencia de disparo S de m_i tal que contiene a la transición T_j. En otras palabras, cualquiera sea la evolución, existe una posibilidad de disparar T_j. Si todas las transiciones están vivas para la marca m_0, la red se dice que es **viva** (Ninguna transición se volverá no disparable). Ahora bien, una transición T_j se dice que es quasi viva para un marcado inicial m_0 si existe una secuencia de disparo desde m_0 que contiene a T_j; si todas las transiciones son quasi vivas, la red es **quasi viva**. En otras palabras, una transición es quasi viva si exista una posibilidad que esta transición puede ser disparada.
En cuando al **deadlock** o estado de sumidero, es un marcado para cual ninguna transición está habilitada. Una red se dice que está free deadlock para un marcado inicial m_0 si ningún marcado alcanzable es un deadlock.
Nota: Las propiedades de deadlock y quasi vida son independientes. La propiedades de vivacidad y deadlock no lo son, una red viva no puede tener un deadlock i.e. una red con un deadlock no puede ser una red viva.

4. Transiciones y plazas invariantes

Comenzando desde un marcado inicial, el marcado puede evolucionar para una red dada y si no existe deadlock puede volverse indefinido. Sin embargo, no cualquier marcado puede ser alcanzado, todo marcado alcanzable posee algunas propiedades en común; una propiedad la cual no varia cuando las transiciones se disparan se dice que es *invariante*. De manera similar, no cualquier secuencia puede ser disparada, algunas propiedades invariantes son comunes a las posibles secuencias de disparo.
Un invariante de plaza o de marcado, se obtiene si existe un subconjunto de plazas $P = P_j\ j = 1, ..., n \in \mathcal{P}$ y un vector de ponderación $(q_1, ..., q_n)$ para el cual todos los pesos q_i son enteros positivos tales que:

$$q_1 m(P_1) + q_2 m(P_2) + ... + q_n m(P_n) = k \forall m_i \tag{4.1}$$

El conjunto de lugares P es un componente conservador, la propieades de ser un componente conservador es independiente de la marca inicial (Propiedad estructural). Por otro lado, la constante de la plaza invariante depende de la marca inicial.
La definición de transiciones invariantes es similar, salvo que se trata de un componente repetitivo.

5. Redes sincronizadas

En una red autónoma, sabemos que una transición puede ser disparada si esta habilitada, pero no sabemos cuándo será disparada. Una red sincronizada, un evento es asociado con cada transición y cada dispara de este ocurriría: Si la transición esta habilitada y cuando el evento asociado ocurre. Los eventos externos corresponden a cambios en el estado del mundo exterior (incluido el tiempo); por posición, un cambio en el estado interno, un cambio en el marcado, podría llamarse un evento interno. Los eventos no tienen duración.

6. Redes temporales

Existen diversos caminos de abordar las redes temporales, de todos ellos abordaremos las redes que tienen una retención temporal en las plazas y la forma de analizarlas. Su primera aproximación fue ser una especificación formal para el diseño de sistemas complejos autómatas. Lo mas importante para esta aplicación son las posibilidades para el diagnóstico de sistemas, denominadas detección de errores, localización evaluación reconocimiento y reacción. Es importante minimizar el tiempo total necesario para el diagnóstico interno de procesos de un sistema; considerando solo las relaciones causales entre los eventos es por tanto no suficiente.
En orden de modelar sistemas dirigidos por eventos en una forma natural por las redes de Petri nos es deseable ser capaces de especificar **delays** y distancias seguras de tiempo entre los eventos. Esto es porque extendemos las redes de Petri clásica por retenciones temporales para los tokens en las plazas. El tiempo asignado a un aplaza puede ser interpretado en diferentes maneras, como un delay mínimo y máximo, una combinación de ambos, como una retención de tiempo exacta o periodos de validez, entre otros.
Aquí se considerada una extensión temporal donde para cada plaza un intervalo especifica el tiempo de retención mínimo y máximo para los tokens que residen en esta. Cuando un token llega a una plaza, “su tiempo” comienza a correr. Si el token permanece en la plaza hasta después de que el tiempo máximo de retención ha pasado, podemos definir que no puede ser usado para disparar la transición y almacenarlo simplemente en la plaza o podemos resetear su tiempo a cero. El tiempo mínimo de retención puede ser visto como una clase de tiempo de preparación requerido antes de que el token este listo para usar y el token solo espera lista por una limitada cantidad de tiempo, hasta su máximo de retención. Si este se pasa, entonces es necesario prepara de nuevo y poder usar para dispara cuando se alcance su tiempo mínimo. La

segunda versión es la que se considerara. El mínimo y máximo tiempo de retención para cada plaza puede ser visto como una ventana temporal para cada token en la plaza. La ventana esta cerrada hasta que token considerada hasta alcanzar el tiempo minino de retención. Entonces la venta se abre y el token puede dejar la plaza, pero solo hasta que el tiempo máximo de retención se alcance, después la ventana se cierra. Formalmente una red de Petri con retención temporal en la plaza o con ventanas temporales ($tw - PN$) es un par P = (N,I) tal que:

$$N = \text{Es una red de Petri, P es un conjunto de plazas} \tag{6.1}$$

$$I = P \to \mathbb{Q}_{\nvDash}{}^{+} \times (\mathbb{Q}_{\nvDash}{}^{+} \cup \infty) \text{con} I(p) = (l_p, u_p) \wedge l_p \leq u_p \forall p \in P \tag{6.2}$$

Los tokens pueden llegar obviamente a una plaza en diferentes momentos, por lo tanto, cada uno necesita su propio reloj.

7. Marco teórico

La red de Petri que modela el sistema se muestra a continuación:

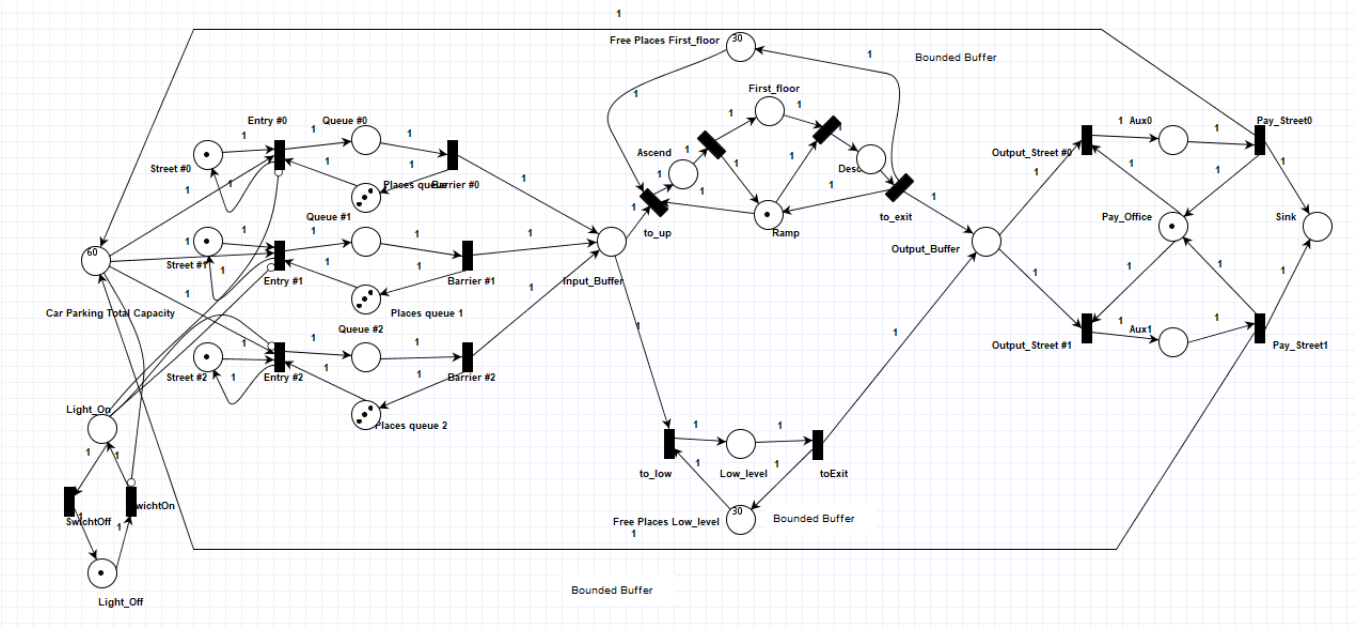


Figura 7.1: High level red de Petri

8. Tabla de eventos y estados

Tabla de Estados	
Switch ON	Cartel encendido
Switch OFF	Cartel Apagado
CarParking Total Capacity	Capacidad total del estacionamiento
Street #0	Ingreso por calle 1
Street #1	Ingreso por calle 2
Street #2	Ingreso por calle 3
Queue #0	Cola espera entrada 1
Queue #1	Cola espera entrada 2
Queue #2	Cola espera entrada 3
InputBuffer	Elección de piso
toUp	Al primer piso
toDown	A planta baja
toExit	Hacia la salida
OutputBuffer	Elección de calle de salida
OutputStreet #0	Salida por calle 1
OutputStreet #1	Salida por calle 2
PayStreet #0	Pago por calle 1
PayStreet #1	Pago por calle 2

Tabla de Eventos	
Switch ON	Cuando no hay más lugar en la playa
Switch OFF	Cartel hay lugar en la playa
Entry #0	Vehículo que quiere ingresar por calle 1
Entry #0	Vehículo que quiere ingresar por calle 1
Entry #0	Vehículo que quiere ingresar por calle 3
Barrier #0	Vehículo pasa barrera de entrada 1
Barrier #1	Vehículo pasa barrera de entrada 2
Barrier #2	Vehículo pasa barrera de entrada 3
Ascend	Sube al primer piso, usa recurso rampa
Descend	Baja del primer piso, usa recurso rampa
toDown	Entra a planta baja
toExit	Hacia la salida por planta baja ó primer piso
OutputStreet #0	Sale por calle 1
OutputStreet #1	Sale por calle 2

9. Cantidad de Threads

La creación de los hilos se es dinámica, si bien se procura usar la menor cantidad de hilos ya que en el archivo de configuración se cargan las transiciones que se pueden disparar en cadena, se opta por usar el framework **Executor**. Las tareas son unidades lógicas de trabajo, y los hilos son mecanimos por los cuales las tareas se pueden ejecutar de manera asíncrona. Una política de ejecución se conoce como *thread-per-task* donde a través de métodos como .start() y .join(),para el caso de Java, se controla la ejecución de los mismos.
Este método carece de recursos de administración, por lo que de acuerdo al libro de Brian Goetz, se opta por usar el framework Executor el cuál se encarga de la ejecución asíncrona de las tareas soportando una variada cantidad de políticas y además permitiendo que los hilos, si implementan la interfaz Callable puedan devolver mayor información que si implementan la clásica interfaz Runneable.
El funcionamiento del framework Executor se puede observar en la siguiente imagen:

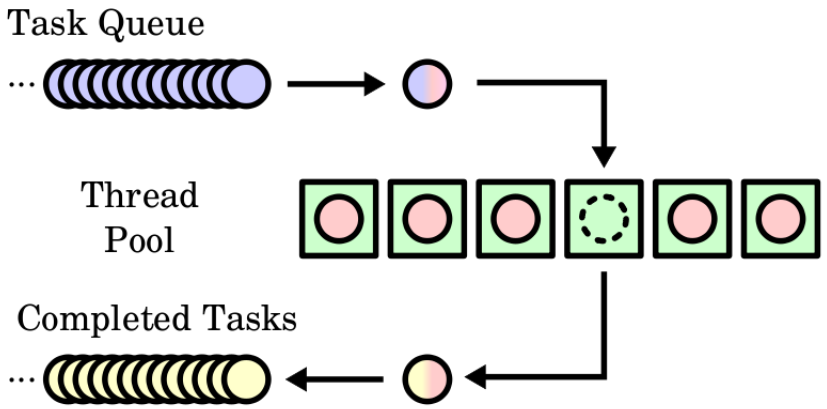
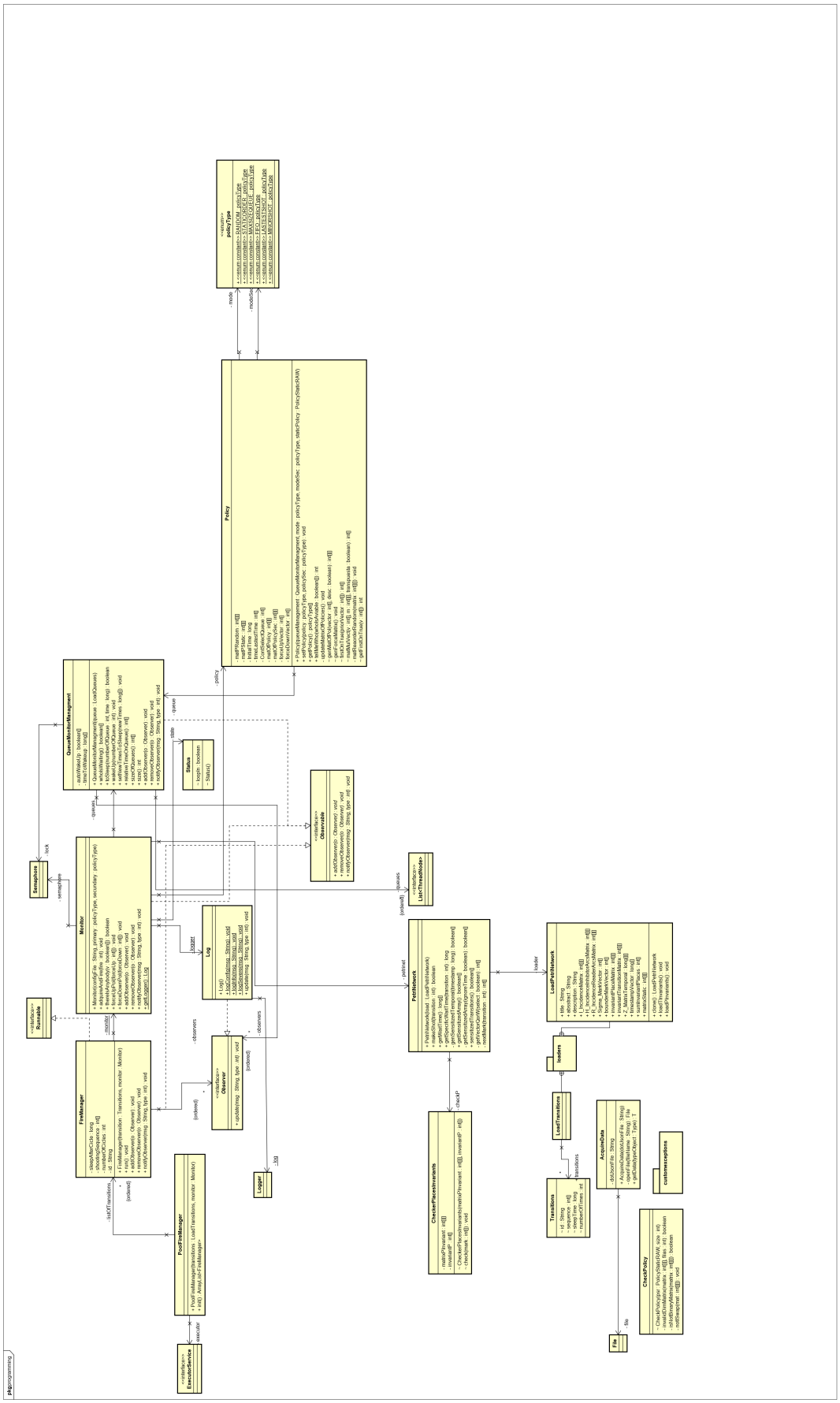
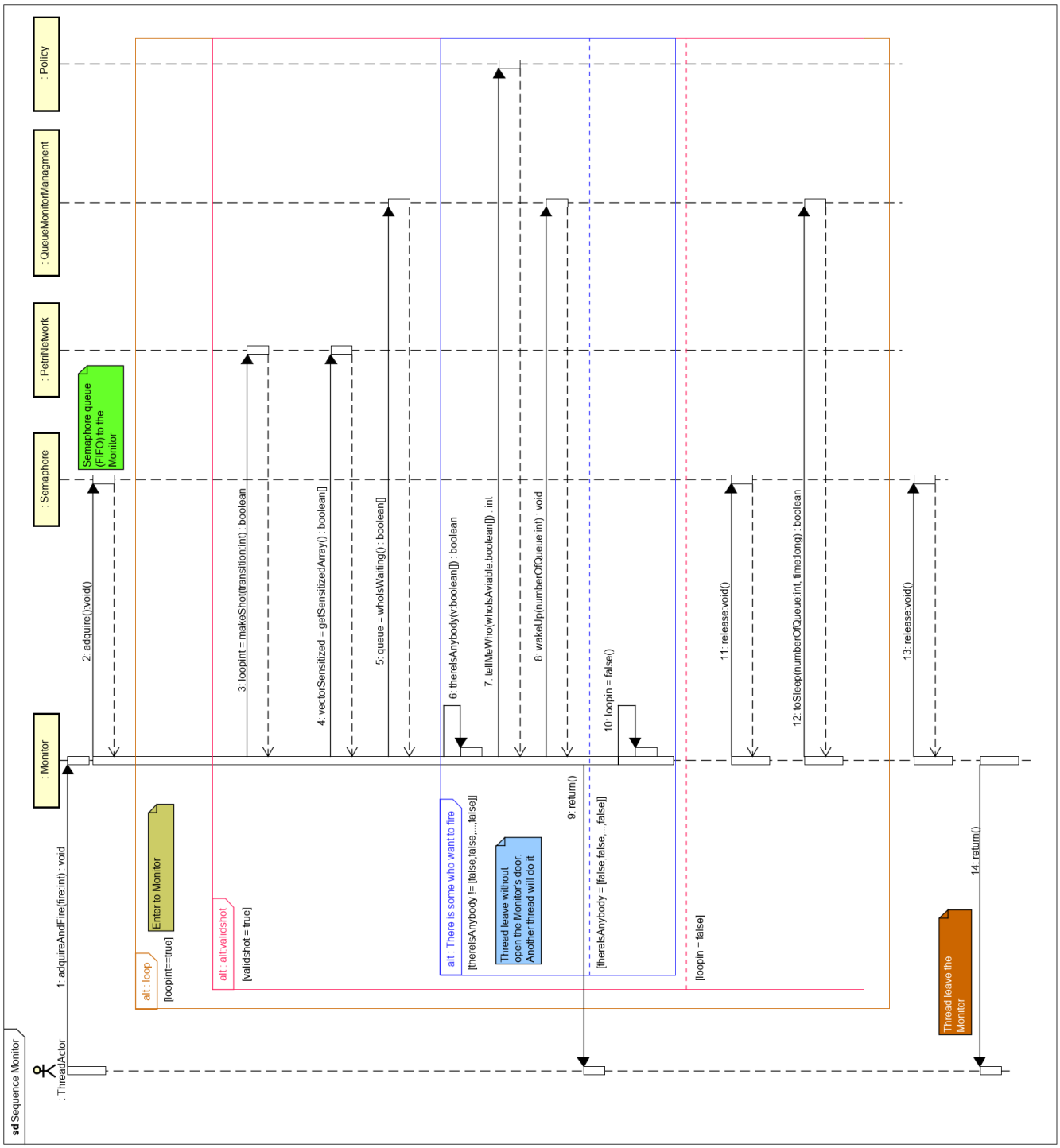


Figura 9.1: Representación del Thread Pool de Executor

10. Diagrama de clase



11. Diagrama de secuencia



Monitor.png

Figura 11.1: Diagrama de secuencia

12. Simulación,Código & Resultados

Toda la información inherente a simulación, código y resultados se encuentra alojada en el siguiente repositorio de GitHub:

Figura 12.1: Repositorio de GitHub

13. Conclusiones

Este trabajo supuso un desafío en múltiples sentidos. Resultó hasta cierto punto ser un dolor de cabeza, pero gracias a la ayuda recibida por parte de amigos como de los docentes a cargo se pudo resolver.

Las aplicaciones de las redes de Petri es impresionante, resulta muy interesante ver como con simples operaciones matemáticas como lo es el producto de matrices se puede llegar a describir por completo a un sistema. Además, la parte del código, si bien se ejecutó y se testeó a criterio personal quedan mejoras por realizarse, ya que una adapción de la interfaz Callable permitiría obtener un mayor control de la ejecución.