INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INGENIERIA EN SISTEMAS COMPUTACIONALES

Propuesta para el desarrollo del proyecto del titulado:

Petri Nets

Presenta:

Rosas Tlatenchi Irving Daniel 181080145

Martínez Flores Mario Antonio 181080127

Sánchez Sánchez Diego 181080128

Profesor:

PARRA HERNANDEZ ABIEL TOMAS

CIUDAD DE MÉXICO **JUNIO / 2021**

Contenido

Introducción

La concurrencia y la distribución se han convertido en el paradigma y la preocupación dominantes en la informática. A pesar del hecho de que gran parte de la investigación inicial en programación orientada a objetos se centró en sistemas secuenciales, los objetos son una unidad natural de distribución y concurrencia, como se esclareció al principio de la investigación sobre el modelo Actor. Así, los modelos y teorías de concurrencia, siendo la más antigua las redes de Petri, y su relación con los objetos son un atractivo tema de estudio.

Las redes de Petri se han utilizado ampliamente como herramienta para modelar y analizar sistemas concurrentes y distribuidos durante muchos años; sin embargo, sus aplicaciones se limitan principalmente a las actividades anteriores de desarrollo de sistemas de software. Para hacer de las redes de Petri una metodología de desarrollo de software completamente desarrollada, se necesitan técnicas de generación de código sistemáticas

Presentamos un enfoque para derivar programas concurrentes a partir de redes de Petri que establece un vínculo entre las redes de Petri y la programación concurrente orientada a objetos y forma una base para una metodología de desarrollo de software transformacional basada en redes de Petri.

Objetivo General

Este documento se realiza con el fin de poder dar a conocer como una teoría de autómatas puede llegar a expresar un sistema de eventos concurrentes.

Objetivos Específicos

- Conocer los conceptos fundamentales de la teoría de Petri nets, así como dar a conocer el método de análisis y propiedades cualitativas.
- Analizar y conocer los lenguajes de modelado matemático para saber cómo funciona en un sistema distribuido.
- Implementar problemas que sean concurrentes considerando los lenguajes del modelado matemático.
- Conocer la teoría de autómatas para saber cómo se relaciona con el tipo de eventos concurrentes y así hacer implementaciones con el Petri nets.

Justificación

Este proyecto tiene la necesidad de realizarse para dar un ámbito muy general acerca del modelo computacional Petri Nets, ya son un formalismo apropiado para el modelado de sistemas dinámicos eventos discretos. Se han aplicado satisfactoriamente en campos tales como redes de comunicaciones, automatización lógica, sistemas de fabricación, etc. La naturaleza grafica de los modelos de redes Petri Nets facilita la conexión entre diseñadores y usuarios. Los fundamentos matemáticos del formalismo permiten además el análisis de corrección y de eficiencia. Así mismo, estos modelos pueden ser implementados de forma automática utilizando técnicas hardware o software, y pueden ser utilizados con fines de monitorización del sistema una vez que este se encuentra en funcionamiento. En otras palabras, se pueden usar durante todo el ciclo de vida del sistema. Las redes de Petri cuentan con una gran base matemática que permite la verificación de posesiones, tanto de eficacia, como de corrección (las cosas pasan de manera adecuada). El estudio de estas propiedades, teniendo en cuenta la asignación de prioridades de la etapa de planificación, permite la verificación de los requisitos funcionales y temporales del sistema modelado en etapas anteriores. De esta manera es posible la detección temprana de comportamiento-Repaso de los formalismos de redes de Petri Nets erróneos, facilitando su corrección antes de la puesta a punto del sistema. Una vez obtenido un modelo correcto del sistema que se comporta de acuerdo con las especificaciones, incluso teniendo en cuenta las restricciones de la plataforma de implementación, es preciso generar dicha implementación.

Resumen

Una red de Petri, también conocida como una red de lugar / transición (PT), es uno de los varios lenguajes de modelado matemático para la descripción de sistemas distribuidos. Es una clase de sistema dinámico de eventos discretos. Una red de Petri es un grafo bipartito dirigido que tiene dos tipos de elementos, lugares y transiciones, representados como círculos blancos y rectángulos, respectivamente.

Además de aclarar la definición y conceptos de las Redes Petri, así como representación matemática o gráfica de los eventos discreto que nos describe su topología de un sistema distribuido, paralelo o concurrente.

Palabras clave: Red, lenguaje, grafo bipartito, Token, Diagrama, Notación Grafica, Transiciones, Arcos, Multiset, Accesibilidad, K-Bounded.

Abstract

A Petri net, also known as a place / transition (PT) network, is one of several mathematical modeling languages for the description of distributed systems. It is a kind of dynamic system of discrete events. A Petri net is a directed bipartite graph that has two types of elements, places and transitions, represented as white circles and rectangles, respectively.

Keywords: Network, language, bipartite graph, Token, Diagram, Graphic notation, Transitions, Arcs, Multiset, Accessibility, K-Bounded.

Marco Teórico

Teoría de la computación

La teoría de la computación o teoría de la informática es un conjunto de conocimientos racionales y sistematizados que se centran en el estudio de la abstracción de los procesos, con el fin de reproducirlos con ayuda de sistemas formales; es decir, a través de símbolos y reglas lógicas. La teoría de la computación permite modelar procesos dentro de las limitaciones de dispositivos que procesan información y que efectúan cálculos; como, por ejemplo, el ordenador. Para ello, se apoya en la teoría de autómatas, a fin de simular y estandarizar dichos procesos, así como para formalizar los problemas y darles solución.

La teoría de la computación se ocupa de determinar qué problemas pueden ser resueltos computacionalmente y con qué eficiencia. La teoría considera distintos modelos de cómputo, como los autómatas finitos (que son los más sencillos), las máquinas de Turing (que son las computadoras usuales de hoy en día) y las computadoras cuánticas (cuyo funcionamiento no es digital). Las lógicas y los lenguajes formales juegan un rol central en la teoría de la computación porque permiten expresar propiedades de los programas y razonar sobre su comportamiento. La teoría de la computación también se encarga de entender el límite entre los problemas computables y los no-computables y, dentro del mundo de lo computable, clasificarlos de acuerdo a su grado de simpleza o dificultad.

A un nivel más abstracto, investigamos las propiedades teóricas de los sistemas de reescritura y los modelos de cómputo fuertes, como el cálculo lambda. A la inversa, estudiamos modelos de cómputo débiles, como los autómatas finitos y sus numerosas variantes. Nos ocupamos también de la noción de aleatoriedad en relación a los distintos modelos de cómputo y a los grados de dificultad de los problemas. Por último, introducimos nociones provenientes de la teoría de funciones computables y de la teoría de la aleatoriedad en el procesamiento cuántico de la información y en algunos modelos computacionales que intentan explicar ciertas características de la cognición humana.

Teoría de la computabilidad

La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que se pueden resolver con un algoritmo o equivalentemente con una máquina de Turing. Esta teoría explora los límites de la posibilidad de solucionar problemas mediante algoritmos. Gran parte de las ciencias computacionales están dedicadas a resolver problemas de forma algorítmica, de manera que el descubrimiento de problemas imposibles es una gran sorpresa. La teoría de la computabilidad es útil para no tratar de resolver algorítmicamente estos problemas, ahorrando así tiempo y esfuerzo.

Los problemas se clasifican en esta teoría de acuerdo a su grado de imposibilidad:

- Los computables: Son aquellos para los cuales sí existe un algoritmo que siempre los resuelve cuando hay una solución y además es capaz de distinguir los casos que no la tienen. También se les conoce como decidibles, resolubles o recursivos.
- Los semicomputables: Son aquellos para los cuales hay un algoritmo que es capaz de encontrar una solución si es que existe, pero ningún algoritmo que determine cuando la solución no existe (en cuyo caso el algoritmo para encontrar la solución entraría a un bucle infinito). El ejemplo clásico por excelencia es el problema de la parada. A estos problemas también se les conoce como listables, recursivamente enumerables o reconocibles, porque si se enlistan todos los casos posibles del problema, es posible reconocer a aquellos que sí tienen solución.
- Los incomputables: Son aquellos para los cuales no hay ningún algoritmo que los pueda resolver, no importando que tengan o no solución. El ejemplo clásico por excelencia es el problema de la implicación lógica, que consiste en determinar cuándo una proposición lógica es un teorema; para este problema no hay ningún algoritmo que en todos los casos pueda distinguir si una proposición o su negación es un teorema.

Hay una versión más general de esta clasificación, donde los problemas incomputables se subdividen a su vez en problemas más difíciles que otros.

Modelo de computación

En la teoría de la computabilidad y en la teoría de la complejidad computacional, un modelo de computación es la definición un conjunto de operaciones que permiten ser usadas en el cómputo y sus respectivos costos. Solo asumiendo un cierto modelo de computación es posible analizar los recursos de cómputo requeridos, como el tiempo de ejecución o el espacio de memoria, o discutir las limitaciones de algoritmos o computadores.

Secuenciales

Son las técnicas utilizadas cuando el orden y la secuencia de los datos aportan mucho valor predictivo. Una secuencia es una serie de elementos que se suceden unos a otros y guardan relación entre sí.

Esta tipología está presente en casi cualquier problema de la vida real. Los ejemplos que se me vienen ahora a la cabeza son:

- Reconocimiento de actividad en vídeo.
- Análisis de la secuencia de ADN.
- Generación de música, de diálogo etc.
- Reconocimiento de voz.
- Análisis de sentimiento.
- Traducción de texto.
- Funcionales

El modelo de objetos describe las propiedades estructurales del sistema. El modelo dinámico y funcional describen su comportamiento.

- El modelo funcional describe los comportamientos y operaciones de los objetos.
- El modelo funcional muestra la dependencia de datos en el sistema.
- El modelo funcional describe la computación dentro del sistema, cómo los valores de salida se derivan de los valores de entrada, sin importar el orden en que son computados.

- El modelo funcional consiste de múltiples diagramas de flujo de datos.
- El modelo de objetos inicialmente incluye solo algunas operaciones de alto nivel definidas en ciertas clases.
- Las clases que tienen operaciones y comportamientos complejos son modelados funcionalmente.
- Construir un modelo funcional para estas clases ayuda a identificar las otras clases con las cuales se interacciona y con las cual existe alguna dependencia.
- Creando un modelo funcional también ayuda a identificar nuevos objetos, atributos, y operaciones.
- Los objetos, atributos, y operaciones identificadas deben ser luego añadidas al modelo de objetos
- El modelo funcional provee un medio para identificar las restricciones operacionales del mundo real.
- No todos los sistemas tienen un modelo funcional importante, al igual que no todos los sistemas tienen un modelo dinámico importante. En cualquier tipo de sistema el modelo de objetos es por lo general el más importante.

Concurrentes

La computación concurrente es una forma de computación en la que se ejecutan varios cálculos al mismo tiempo, durante períodos de tiempo superpuestos, en lugar de secuencialmente, y uno se completa antes de que comience el siguiente.

Ésta es una propiedad de un sistema, ya sea un programa, una computadora o una red, donde hay un punto de ejecución separado o "hilo de control" para cada proceso. Un sistema concurrente es aquel en el que un cálculo puede avanzar sin esperar a que se completen todos los demás cálculos. El concepto de computación concurrente se confunde frecuentemente con el concepto relacionado pero distinto de computación paralela, aunque ambos pueden describirse como "múltiples procesos que se ejecutan durante el mismo período de tiempo ".

Las ventajas de la computación concurrente incluyen:

- Mayor rendimiento del programa: la ejecución paralela de un programa concurrente permite que el número de tareas completadas en un tiempo determinado aumente proporcionalmente al número de procesadores de acuerdo con la ley de Gustafson.
- Alta capacidad de respuesta para entrada / salida: los programas intensivos en entrada / salida esperan principalmente a que se completen las operaciones de entrada o salida. La programación concurrente permite el tiempo que se gastaría esperando para ser utilizado para otra tarea.
- Estructura de programa más apropiada: algunos problemas y dominios de problemas se adaptan bien a la representación como tareas o procesos concurrentes.

Petri Nets

Es una clase de sistema dinámico de eventos discretos. Una red de Petri es un grafo bipartito dirigido que tiene dos tipos de elementos, lugares y transiciones, representados como círculos blancos y rectángulos, respectivamente. Un lugar puede contener cualquier número de fichas, representadas como círculos negros. Una transición está habilitada si todos los lugares conectados a ella como entradas contienen al menos un token. Al igual que los estándares de la industria, como los diagramas de actividades UML, el modelo y la notación de procesos de proceso de negocio y las cadenas de procesos impulsadas por eventos, las redes de Petri ofrecen una notación gráfica para procesos escalonados que incluyen elección, iteración y ejecución simultánea. A diferencia de estos estándares, las redes de Petri tienen una definición matemática exacta de su semántica de ejecución, con una teoría matemática bien desarrollada para el análisis de procesos.

Fundamentos de la red de Petri

Una red de Petri consiste en lugares, transiciones y arcos. Los arcos van de un lugar a una transición o viceversa, nunca entre lugares o entre transiciones. Los lugares desde los que se ejecuta un arco a una transición se denominan los lugares de entrada

de la transición; los lugares a los que se ejecutan los arcos desde una transición se denominan lugares de salida de la transición. Gráficamente, los lugares en una red de Petri pueden contener un número discreto de marcas llamadas tokens.

Cualquier distribución de tokens sobre los lugares representará una configuración de la red llamada marca. En un sentido abstracto relacionado con un diagrama de red de Petri, una transición de una red de Petri puede dispararse si está habilitada, es decir, hay suficientes tokens en todos sus lugares de entrada.

Definición formal y terminología básica

Las redes de Petri son sistemas de transición de estado que extienden una clase de redes llamadas redes elementales.

Si una red de Petri es equivalente a una red elemental, entonces Z puede ser el conjunto contable {0,1} y aquellos elementos en P que se asignan a 1 bajo M forman una configuración. Del mismo modo, si una red de Petri no es una red elemental, entonces el multiset M puede interpretarse como la representación de un conjunto de configuraciones no single. En este sentido, M extiende el concepto de configuración de redes elementales a las redes Petri.

En el diagrama de una red de Petri los lugares se representan convencionalmente con círculos, transiciones con rectángulos largos y estrechos y arcos como flechas unidireccionales que muestran conexiones de lugares a transiciones o transiciones a lugares.

Propiedades matemáticas de las redes de Petri

Una cosa que hace que las redes de Petri sean interesantes es que proporcionan un equilibrio entre la potencia de modelado el análisis: muchas cosas que a uno le gustaría saber sobre los sistemas concurrentes se pueden determinar automáticamente para las redes de Petri, aunque algunas de esas cosas son muy costosas de determinar en el caso general. Una visión general de tales problemas de decisión, con resultados de decidibilidad y complejidad para redes de Petri y algunas subclases, se puede encontrar en Esparza y Nielsen (1995).

Accesibilidad

El problema del realizacion para las redes de Petri es decidir, dado una red de Petri N y una marca M, si M∈R (N). Claramente, se trata de recorrer el gráfico de accesibilidad definido anteriormente, hasta que lleguemos a la marca solicitada o sepamos que ya no se puede encontrar. Esto es más difícil de lo que puede parecer al principio: el gráfico de accesibilidad es generalmente infinito, y no es fácil determinar cuándo es seguro detenerlo.

Si bien la accesibilidad parece ser una buena herramienta para encontrar estados erróneos, para problemas prácticos el gráfico construido generalmente tiene demasiados estados para calcular. Para aliviar este problema, la lógica temporal lineal se utiliza generalmente junto con el método tableau para demostrar que no se puede alcanzar tales estados.

Delimitación

Un lugar en una red de Petri se llama k-bounded si no contiene más de k fichas en todas las marcas alcanzables, incluyendo la marca inicial; se dice que es seguro si es 1-acotado; es acotado si es k-acotado para algunos k.

Una red de Petri (marcada) se llama k-acotada, segura, o acotada cuando todos sus lugares están. Una red de Petri (gráfico) se llama (estructuralmente) acotada si está acotada para cada posible marca inicial. Nótese que una red de Petri está acotada si y solo si su gráfica de accesibilidad es finita. La delimitación es decidida mirando la cubierta, mediante la construcción del árbol de Karp-Miller.

Redes Petri discretas, continuas e híbridas

Así como para eventos discretos, existen redes de Petri para procesos continuos e híbridos discreto-continuos que son útiles en la teoría de control discreto, continuo e híbrido, y relacionados con autómatas discretos, continuos e híbridos.

Extensiones

Hay muchas extensiones a las redes de Petri. Algunos de ellos son completamente retro compatibles con la red de Petri original, algunos agregan propiedades que no se pueden modelar en el formalismo original de la red de Petri. Aunque los modelos compatibles con versiones anteriores no amplían la potencia computacional de las redes de Petri, pueden tener representaciones más sucintas y pueden ser más convenientes para el modelado.

El término red de Petri de alto nivel se utiliza para muchos formalismos de la red de Petri que extienden el formalismo básico de la red de P/T; esto incluye redes de Petri de colores, redes de Petri jerárquicas como redes dentro de redes, y todas las demás extensiones esbozadas en esta sección. El término también se utiliza específicamente para el tipo de redes de colores compatibles con CPN Tools.

Una breve lista de posibles extensiones:

- Arco de restablecimiento: No impone una condición previa al disparo y vacía el lugar cuando se activa la transición; esto hace que la accesibilidad sea indecidible, mientras que algunas otras propiedades, como la terminación, siguen siendo decidibles.
- Arco inhibidor: Impone la condición previa de que la transición sólo puede dispararse cuando el lugar está vacío; esto permite que se expresen cálculos arbitrarios sobre el número de tokens, lo que hace que el formalismo Turing sea completo e implica la existencia de una red universal.
- Red de Petri estándar: Los tokens son indistinguibles. En una red de Petri de color, cada token tiene un valor. En herramientas populares para redes Petri de colores como CPN Tools, los valores de los tokens se escriben, y se pueden probar (usando expresiones de guardia) y manipular con un lenguaje de programación funcional.
- Jerarquía: Esto en forma de diferentes puntos de vista que apoyan los niveles de refinamiento y abstracción fue estudiado por Fehling. Otra forma de jerarquía se encuentra en las llamadas redes de Petri de objetos o sistemas de objetos

donde una red de Petri puede contener redes de Petri como sus tokens que inducen una jerarquía de redes de Petri anidadas que se comunican mediante la sincronización de transiciones en diferentes niveles.

- Redes de Petri priorizadas: Agregan prioridades a las transiciones, por lo que una transición no puede dispararse, si se habilita una transición de mayor prioridad (es decir, puede disparar).
- Dualistic Petri Nets: Es una extensión de Petri Net desarrollada por E. Dawis, para representar mejor el proceso del mundo real. dP-Nets equilibran la dualidad de cambio/no-cambio, acción/pasividad, (transformación) tiempo/espacio, etc. Entre las construcciones bipartidas de Petri Net de transformación y lugar dando como resultado la característica única de marcar las transformaciones, es decir, cuando la transformación está "funcionando" se marca.

Metodología del Trabajo

Decidimos usar la metodología ágil por las siguientes razones son: menores costes y tiempos de ejecución, rapidez ante los cambios, mayor satisfacción del cliente y motivación del equipo y un resultado de mayor calidad. En cuanto a las desventajas, si las hubiera, la flexibilidad de este tipo de metodología las minimiza. Y aunque, en determinados casos, aparecieran inconvenientes las ventajas los superan. Podemos abordar problemas complejos adaptativos, a la vez que se entregan productos de forma eficiente y creativa con el máximo valor.

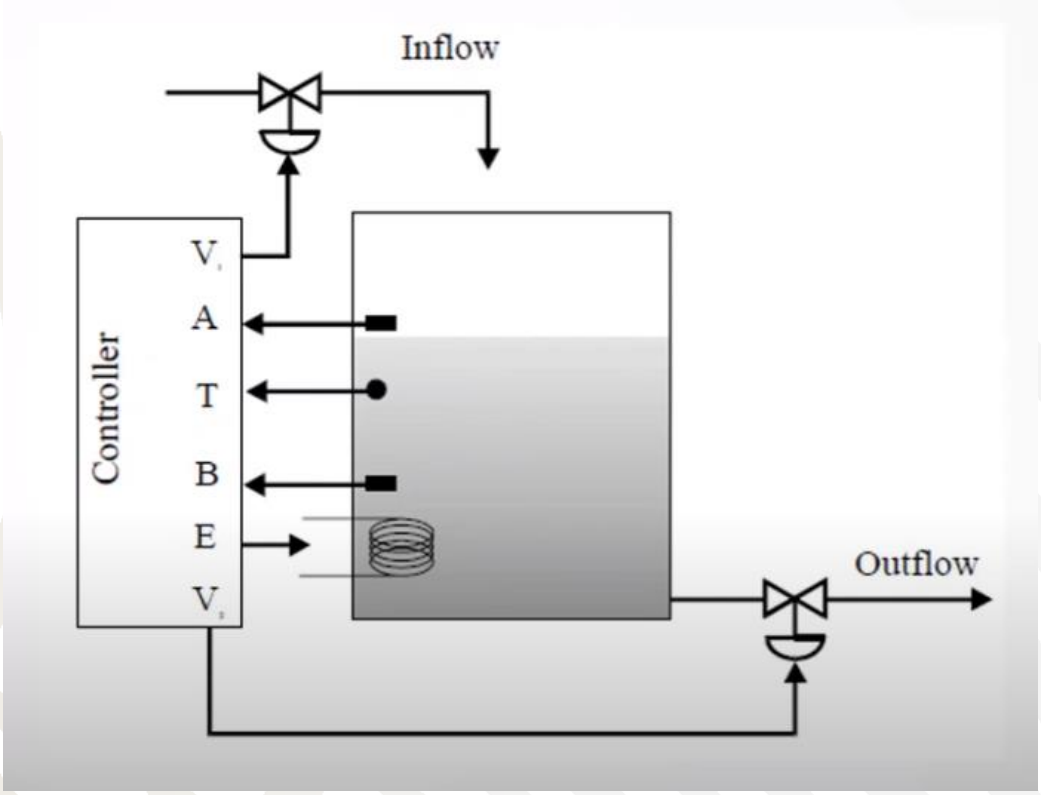
Vamos a poder optar por los siguientes usos:

1. Investigar e identificar mercados viables, tecnologías, y capacidades;
2. Desarrollo de productos y mejoras;
3. Lanzamientos de productos y mejoras, diariamente tantas veces como sea posible;
4. Desarrollo y mantenimiento en la Nube (online, seguridad, por-demanda) y otros entornos operacionales de desarrollo para el uso de producto; y,
5. Mantenimiento y renovación de productos.

Desarrollo e Implementación de las Redes Petri

Proceso de dos tanques

Tenemos un proceso consistente de un tanque con liquido y una resistencia calentadora, una entrada de flujo y una salida de flujo reguladas por válvulas; también se tienen sensores para detectar si el tanque se encuentra en nivel bajo o alto de liquido



El tanque puede ser llenado a partir de la apertura de la válvula de entrada (V_1) y puede ser drenado por la válvula de salida (V_2). Abrir ambas válvulas puede resultar en un comportamiento no definido.

Abrir solo la válvula V_1 significa un incremento del líquido. Este incremento es observado por el sensor de nivel bajo (B) y eventualmente por el sensor de nivel alto (A).

De manera similar, abrir únicamente la válvula (V_2) significa el drenado del tanque, en este caso, el drenado es observado primeramente por el sensor de nivel alto (A) y posteriormente por el sensor de nivel bajo (B).

La resistencia que trabaja como calentador (E) pierde encenderse o apagarse. Cuando la resistencia esta encendida la temperatura del tanque aumenta hasta que eventualmente el sensor de temperatura (T) emite una señal que significa que se ha alcanzado una cierta temperatura preestablecida. Si la resistencia se enciende y el tanque este vacío es un caso no considerado en este.

¿Cuáles procesos y eventos se pueden deducir de la descripción del proceso?

Tenemos 3 sensores: A, B y T, que representan el nivel alto, nivel bajo y temperatura en el tanque, respectivamente.

Tenemos 3 salidas: V_1 , V_2 , E, las válvulas de entrada y salida, y la activación del a resistencia, respectivamente.

También podemos ver que existen dos procesos:

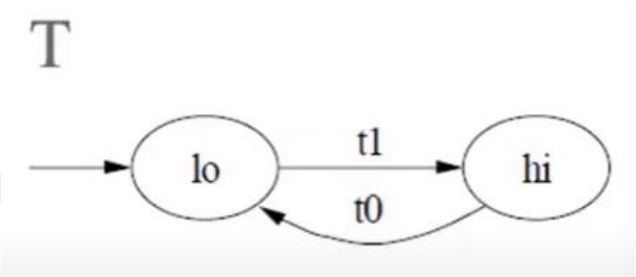
- Módulo de temperatura, donde intervienen T y E.
- Modulo de flujo de líquido, donde intervienen V_1, V_2, A y B.

En esta ocasión solo trabajaremos con el modulo del sensor de temperatura del tanque y su respectiva resistencia.

Resultados

Módulo de temperatura: Sensor de temperatura

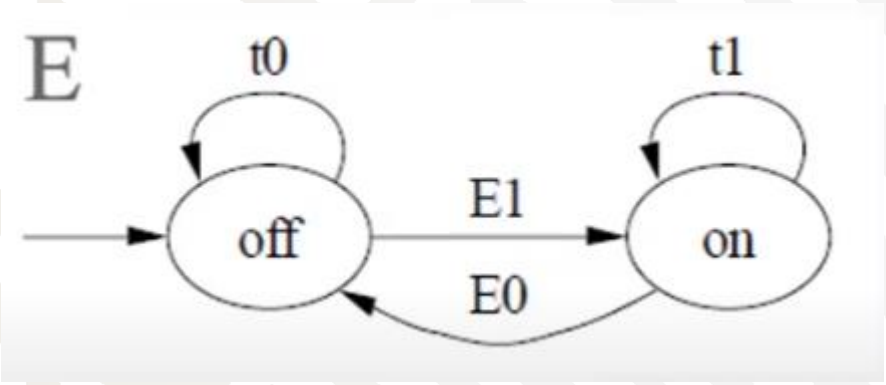
El sensor de temperatura puede estar en uno de estos dos estados; *high* cuando la temperatura esta arriba del valor preestablecido y *low* cuando la temperatura esta abajo. El cambio entre estos estados será asociado a los eventos (t_1) y (t_0). Donde, (t_1) representa el aumento de temperatura sobre el valor preestablecido y (t_0) representa la disminución de la temperatura por debajo del valor preestablecido.



Módulo de temperatura: Resistencia

La resistencia puede encenderse o apagarse, este cambio es controlado por los eventos E_1 y E_0, que representan el encendido de la resistencia y su apagado, respectivamente.

Sin embargo, existe una interacción entre la resistencia y sensor de temperatura; la resistencia y sensor de temperatura; la temperatura solo incrementará si la resistencia está en estado *on* (encendido) y solo disminuirá si la resistencia se encuentra en estado *off* (apagada). La interacción se modela mediante la sincronización o composición en paralelo de los modelos de componentes T y E, esto es $T \,||\, E$.



Obteniendo la red de Petri de T

$G_T = (X_T, E_T, f_{G_T}, \tau_T, x_{0_T}, X_{m_r})$

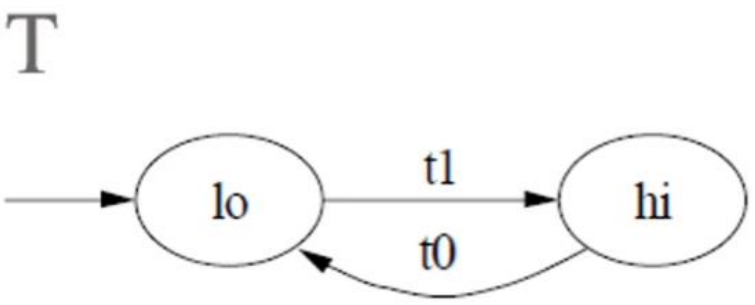
$X_T = \{lo, hi\}$

$E_T = \{t_1, t_0\}$

f_{G_T}:

$f(lo, t_1) = hi$

$f(hi, t_0) = low$



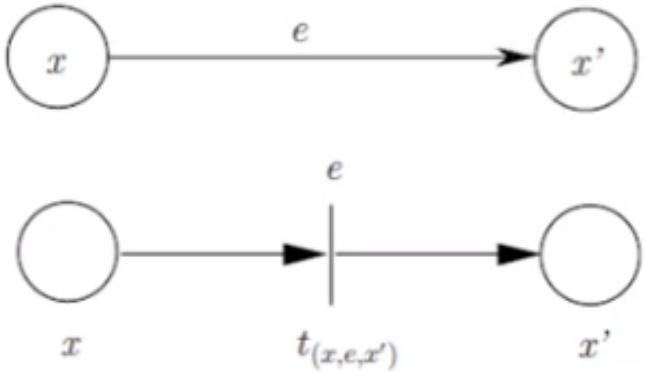
$\tau_T(lo) = t_1$

$\tau_T(hi) = t_0$

$X_{0_T} = lo$

$X_{m_T} = \emptyset$

Podemos crear la red de Petri a partir de la construcción básica que se muestra en la figura de la derecha. Entonces:



$N_T = (P_T, T_T, A_T, W_T, E_T, l_T, x_{0_T}, X_{m_T})$

$P_T = \{p_1, p_2\}$

$T_T = \{t_1, t_0\}$

$A_T = \{(p_1, t_1), (t_1, p_2), (p_2, t_0), (t_0, p_1)\}$

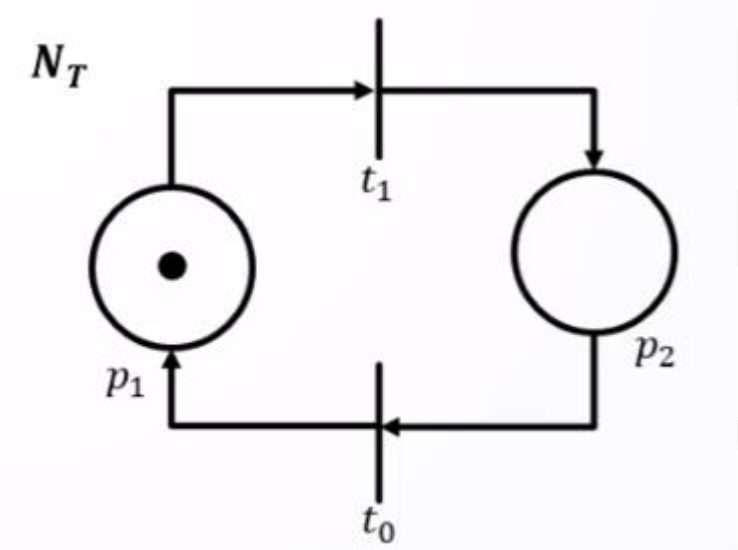
$W_T = 1\ para\ todos\ los\ arcos$

$E_T = \{t_1, t_0\}$

$l_T: l(t_1) = t_1, l(t_0) = t_0$

$X_{0_T} = [1,0]$

$X_{m_T} = \emptyset$



Ecuación de estado de N_T

La ecuación de estado de la red Petri etiquetada N_T es:

$x' = x + uA$

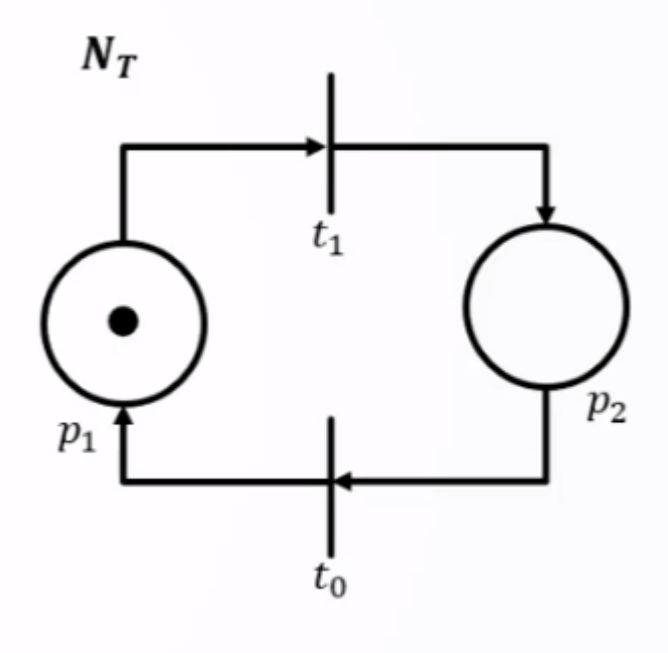
Donde $x' = [x'(p_1), x'(p_2), \dots, x'(p_n)]$ es el estado que se genera a partir del estado actual $x = [x(p_1), x(p_2), \dots, x(p_n)]$, mientras que el vector u representa la transición de interés que está habilitada.

$$u = [0, \dots, 0, 1, 0, \dots 0]$$

A representa la matriz de incidencia, que se compone de:

$a_{ji} = w(t_j, p_i) - w(p_i, t_j)$

Con $j = 1, \dots, m \;\; e \;\; i = 1, \dots, n$



Matriz de incidencia de N_T

Tenemos que:

$P_T = \{p_1, p_2\}$

$T_T = \{t_1, t_0\}$

$w_T: \{w(p_1, t_1) = 1, w(t_1, p_2) = 1, w(p_2, t_0) = 1, w(t_0, p_1)\}$

Entonces tenemos la siguiente matriz de incidencia:

$a_{ji} = w(t_j, p_i) - w(p_i, t_j)$

$$A_{N_T} = \begin{bmatrix} a_{t_1p_1} & a_{t_1p_2} \\ a_{t_0p_1} & a_{t_0p_2} \end{bmatrix} = \begin{bmatrix} 0-(p_1,t_1) & (t_1,p_2)-0 \\ (t_0,p_1)-0 & 0-(p_2,t_0) \end{bmatrix}$$

$$A_{N_T} = \begin{bmatrix} 0-1 & 1-0 \\ 1-0 & 0-1 \end{bmatrix} = A_{N_T} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

Evolución de la red de Petri N_T

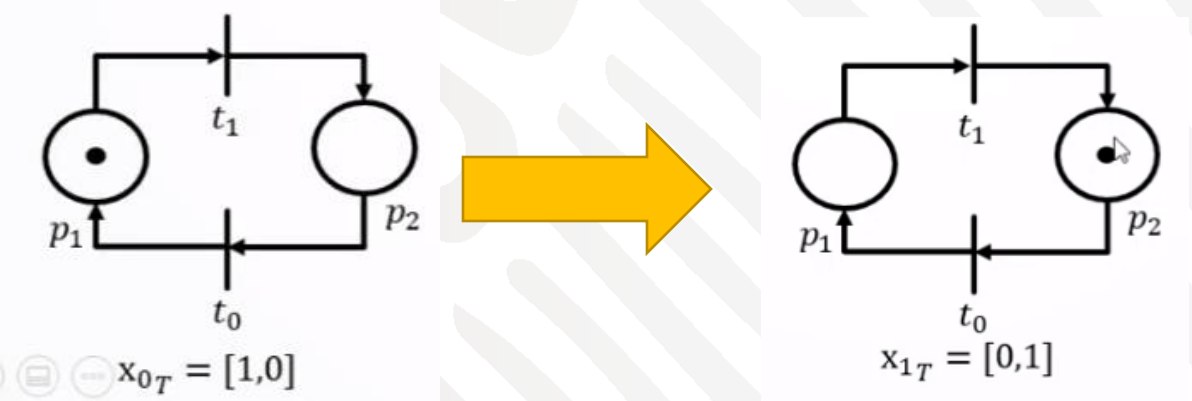
Partiendo de la ecuación de estado

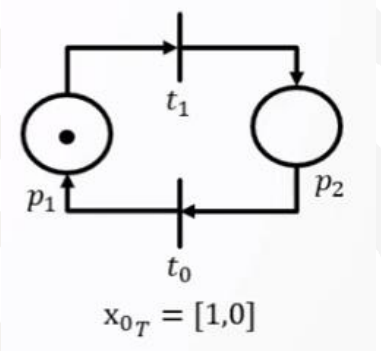
$$x' = x + uA$$

Con $x_{0_T} = [1,0]$ la única transición habilitada es t_1 entonces $u_T = [t_1, t_0] = [1,0]$

$$x_{1_T} = x_{0_T} + u_T A_{N_T} = [1\ 0] + [1\ 0]\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = [1\ 0] + [-1\ 1] = [0\ 1]$$

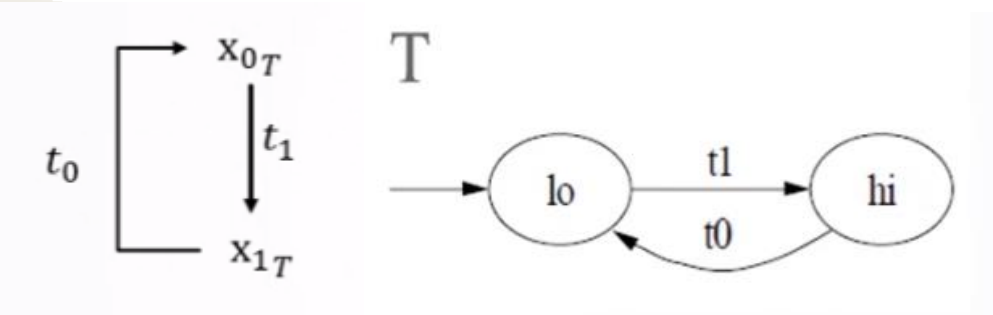
$$x_{2_T} = x_{1_T} + u_T A_{N_T} = [0\ 1] + [0\ 1]\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = [0\ 1] + [1\ -1] = [0\ 1] = x_{0_T}$$





Árbol de evolución de la red de Petri N_T

El árbol de evolución de $N_T$vendría dado por:



Podemos notar que:

- No hay bloqueos
- El árbol de evolución se corresponde con el diagrama del autómata.

Obteniendo la red de Petri de E

$G_E = (X_E, E_E, f_{G_E}, \tau_E, x_{0_E}, X_{m_E})$

$X_E = \{off, on\}$

$E_E = \{t_1, t_0, E_1, E_0\}$

f_{G_E}:

$f(off, t_0) = off$

$f(off, E_1) = on$

$f(on, t_1) = on$

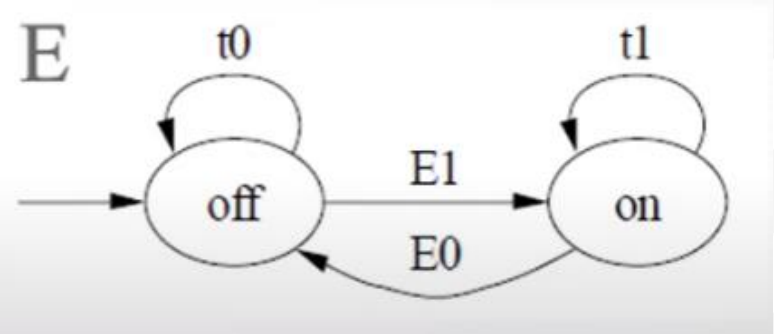
$f(on, E_0) = on$

$\tau_E(off) = \{t_0, E_1\}$

$\tau_E(on) = \{t_1, E_0\}$

$x_{0_E} = off$

$X_{m_T} = \emptyset$



Obteniendo la red de Petri de E

$N_E = (P_E, T_E, A_E, W_E, E_E, l_E, x_{0_E}, X_{mE})$

$P_E = \{p_1, p_2\}$

$T_E = \{t_1, t_0, E_1, E_0\}$

$A_E = \begin{Bmatrix} (p_1, t_0) & (p_1, E_1) & (E_1, p_2) & (t_0, p_1) \\ (p_2, E_0) & (p_2, t_1) & (E_0, p_1) & (t_1, p_2) \end{Bmatrix}$

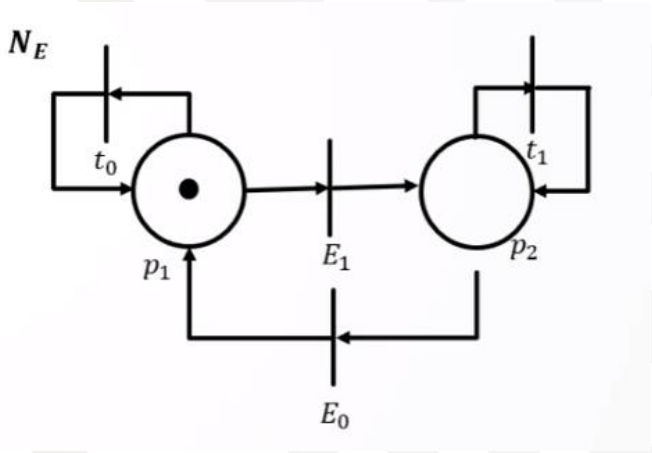
$w_E = 1\ para\ todos\ los\ arcos$

$E_E = \{t_1, t_0, E_1, E_0\}$

$l_E: l(t_1) = t_1, l(t_0) = t_0, l(E_1) = E_1, l(E_0) = E_0$

$x_{0E} = [1,0]$

$X_{m_E} = \emptyset$



Matriz de incidencia de N_E

Tenemos que:

$P_E = \{p_1, p_2\}$

$T_E = \{t_1, t_0, E_1, E_0\}$

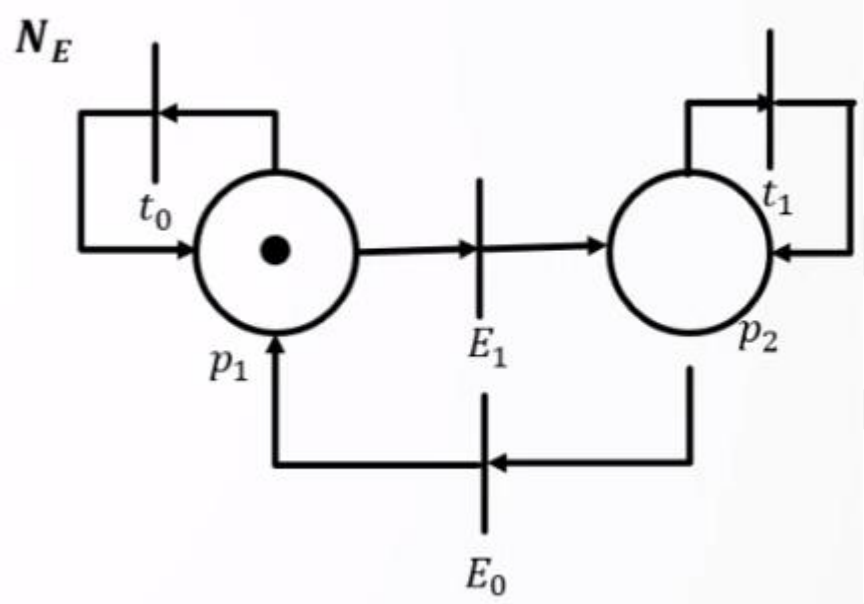
$$w_E = \begin{Bmatrix} w(p_1, t_0) = 1, w(p_1, E_1) = 1, w(E_1, p_2) = 1, w(t_0, p_1) = 1 \\ w(p_2, E_0) = 1, w(p_2, t_1) = 1, w(E_0, p_1) = 1, w(t_1, p_2) = 1 \end{Bmatrix}$$

Entonces la matriz de incidencia

$$a_{ji} = w(t_j, p_i) - w(p_i, t_j)$$

$$A_{N_E} = \begin{bmatrix} a_{t_1p_1} & a_{t_1p_2} \\ a_{t_0p_1} & a_{t_0p_2} \\ a_{E_1p_1} & a_{E_1p_2} \\ a_{E_0p_1} & a_{E_0p_2} \end{bmatrix} = \begin{bmatrix} 0-0 & w(t_1,p_2) - w(p_2,t_1) \\ w(t_0,p_1) - w(p_1,t_0) & 0-0 \\ 0 - w(p_1,E_1) & w(E_1,p_2) - 0 \\ w(E_0,p_1) - 0 & 0 - w(p_2,E_0) \end{bmatrix}$$

$$A_{N_E} = \begin{bmatrix} 0-0 & 1-1 \\ 1-1 & 0-0 \\ 0-1 & 1-0 \\ 1-0 & 0-1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$$



Evolución de la red de Petri N_E

Partiendo de la ecuación de estado

$x' = x + uA$

Con $X_{0_E} = [1\ 0]$ Tenemos dos transiciones habilitadas: $t_0\ y\ E_1$

Con t_0 tenemos $u_{E_{t_0}} = [0,1,0,0]$

$$x_{1_E} = x_{0_E} + u_{E_{t_0}} A_{E_T} = [1\ 0] + [0\ 1\ 0\ 0]\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} = [1\ 0] + [0\ 0] = [1\ 0] = x_{0_E}$$

Ahora disparando E_1, entonces $u_{E_{E_1}} = [0,0,1,0]$

$$x_{2_E} = x_{0_E} + u_{E_{t_0}} A_{E_T} = [1\ 0] + [0\ 0\ 1\ 0]\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} = [1\ 0] + [-1\ 1] = [0\ 1]$$

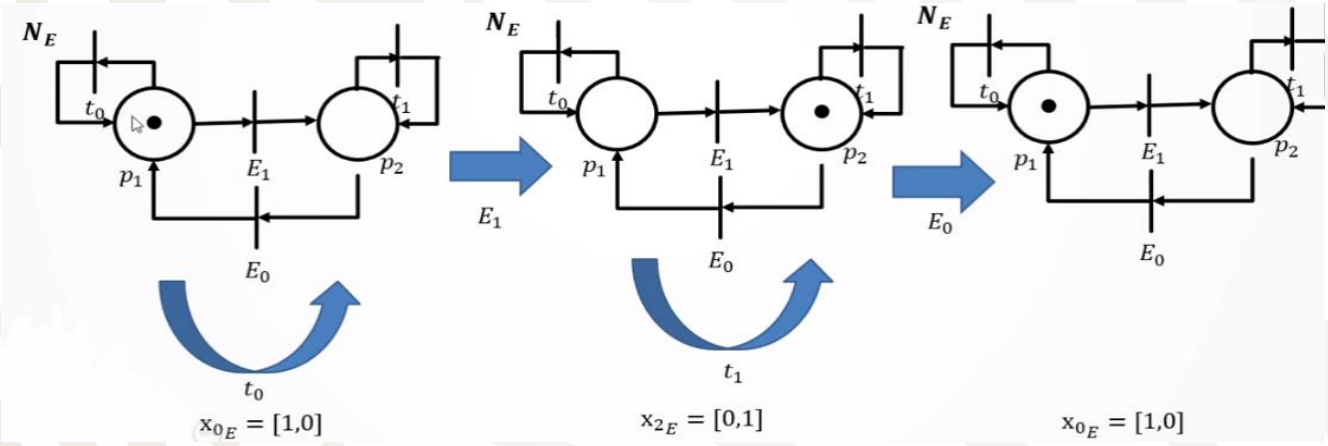
Estando en $x_{2_E} = [0\ 1]$ tenemos dos transiciones habilitadas: $t_1\ y\ E_0$

Con t_1 tenemos $u_{E_{t_1}} = [1,0,0,0]$

$$x_{3_E} = x_{2_E} + u_{E_{t_1}} A_{E_T} = [0\ 1] + [1\ 0\ 0\ 0]\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} = [0\ 1] + [-1\ 1] = [1\ 0] = x_{2_E}$$

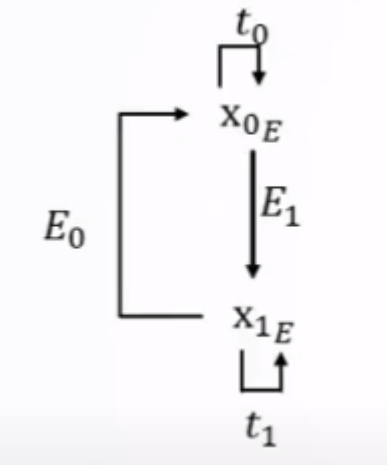
Ahora disparando E_0, entonces $u_{E_{E_0}} = [0,0,0,1]$

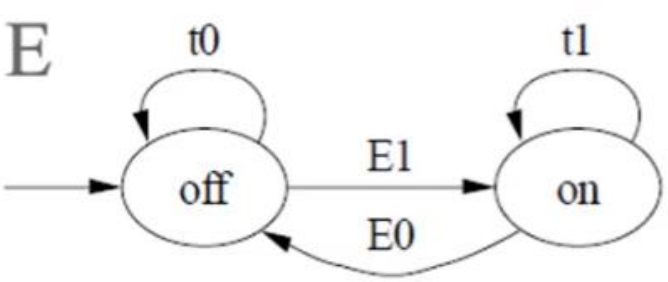
$$x_{4_E} = x_{2_E} + u_{E_{t_1}} A_{E_T} = [0\ 1] + [0\ 0\ 0\ 1]\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} = [0\ 1] + [1\ -\ 1] = [1\ 0] = x_{0_E}$$



Árbol de evolución de la red de Petri N_E

El árbol de evolución de N_E vendría dado por:





Podemos notar que:

- No hay bloqueos
- El árbol de evolución se corresponde con el diagrama autómata.

Composición en paralelo de T y E

Consideremos los autómatas T y E que corresponden al módulo de temperatura

$X_T = \{lo, hi\}$

$E_T = \{t_1, t_0\}$

f_{G_T}:

$f(lo, t_1) = hi$

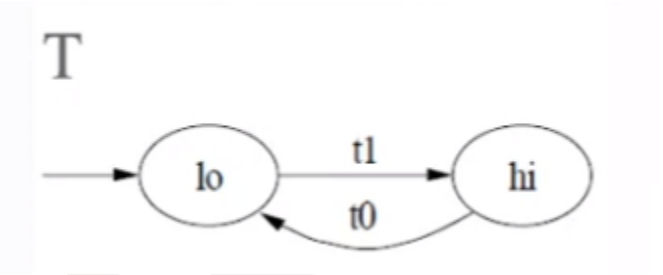
$f(hi, t_0) = low$

$\tau_T(lo) = t_1$

$\tau_T(hi) = t_0$

$X_{0_T} = lo$

$X_{m_T} = \emptyset$



$X_E = \{off, on\}$

$E_E = \{t_1, t_0, E_1, E_0\}$

f_{G_E}:

$f(off, t_0) = off$

$f(off, E_1) = on$

$f(on, t_1) = on$

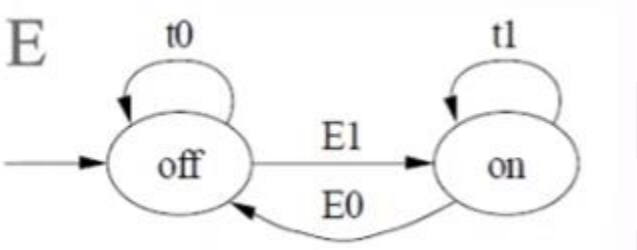
$f(on, E_0) = on$

$\tau_E(off) = \{t_0, E_1\}$

$\tau_E(on) = \{t_1, E_0\}$

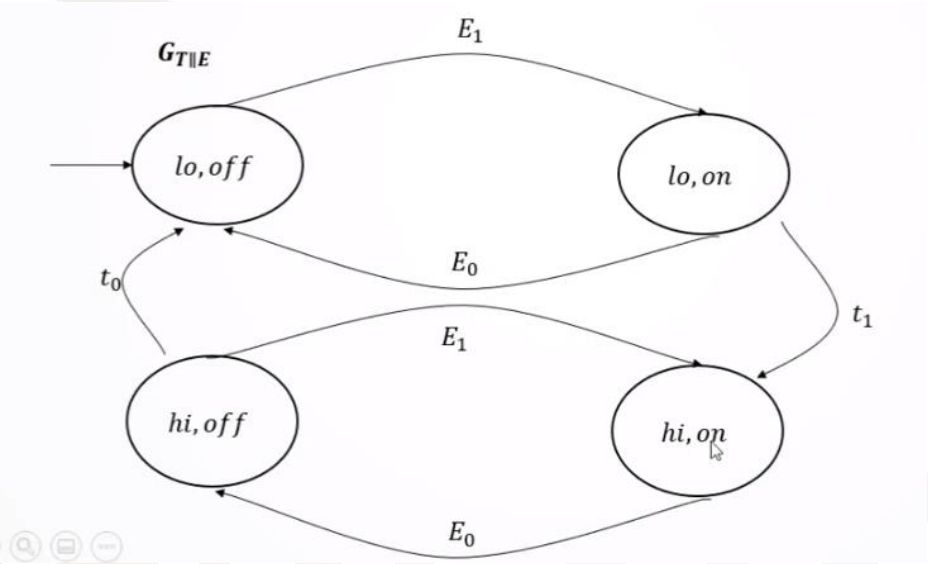
$x_{0_E} = off$

$X_{m_T} = \emptyset$



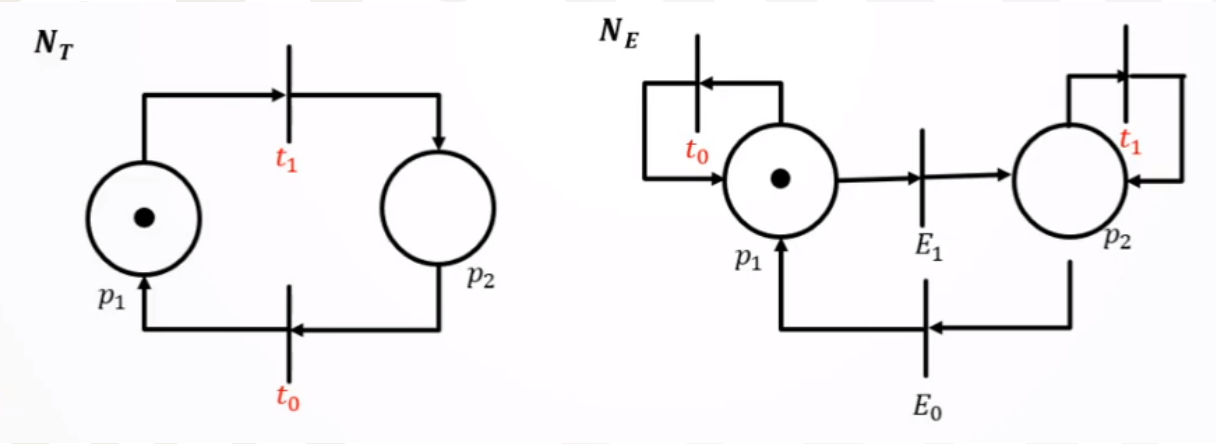
Para utilizar la composición en paralelo necesitamos identificar los eventos en común en los estados de cada elemento en común en este caso es la temperatura.

En base a este análisis obtenemos el siguiente diagrama autómata. Los eventos comunes son $E_{C_{(T||E)}} = \{t_1, t_0\}$, los demás eventos son independientes. El autómata $G_{T||E}$ es:



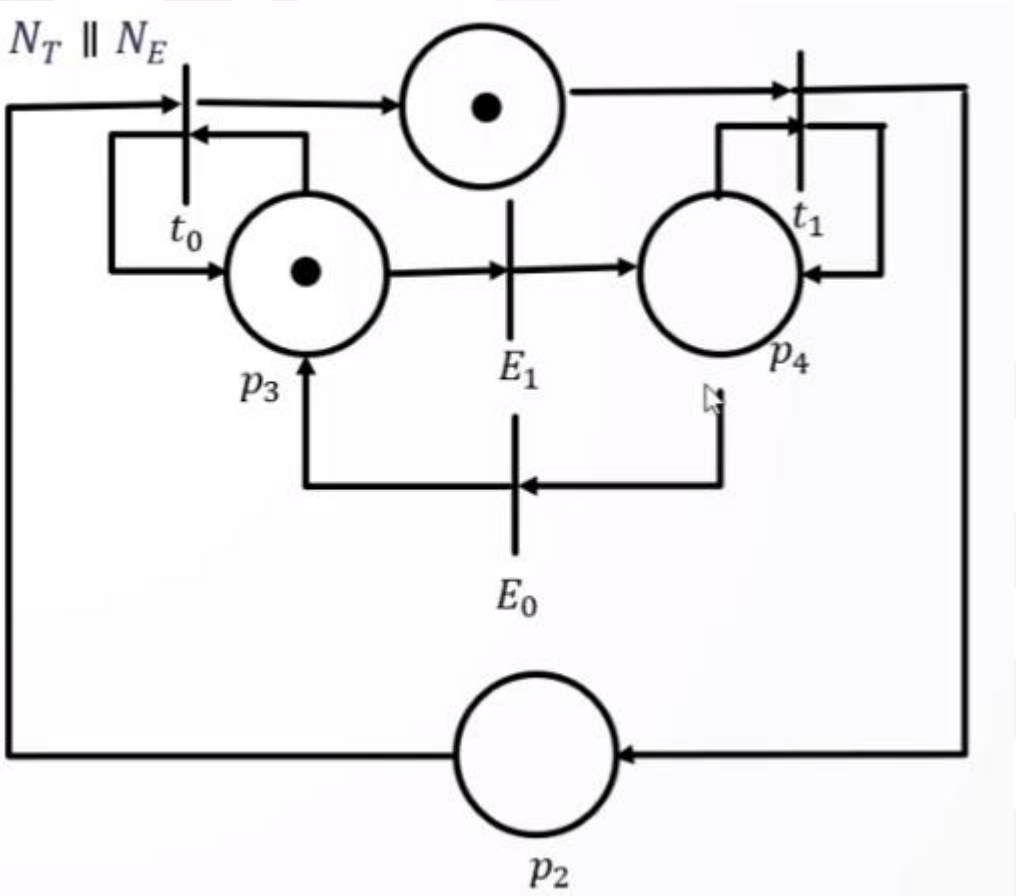
Composición en paralelo de T y E mediante redes de Petri

Tenemos las dos redes de Petri etiquetadas N_T y N_E, lo primero que debemos identificar para obtener la red modular $N_T \,||\, N_E$ son las transiciones en común de las redes, esto es, $T_{(N_T||N_E)} = \{t_1, t_0\}$. Después creamos el grafico donde las redes compartirán las transiciones en común.



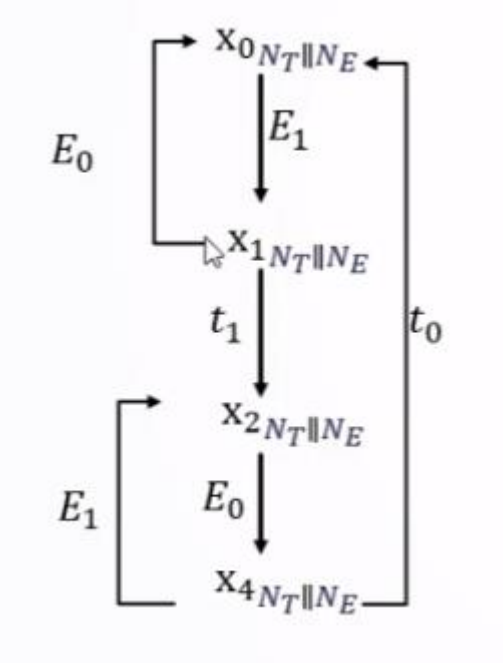
Tenemos las dos redes de Petri etiquetadas N_T y N_E, lo primero que debemos identificar para obtener la red modular $N_T||N_E$ son las transiciones en común de las redes, esto es, $T(N_T||N_E) = \{t_1, t_0\}$. Después creamos el grafico donde las redes compartirán las transiciones en común.

En el estado inicial $x_{0_{N_T||N_E}} = [1,0,1,0]$. Podemos corroborar que esta red de Petri es equivalente a su composición paralela mediante autómatas evolucionando la red.

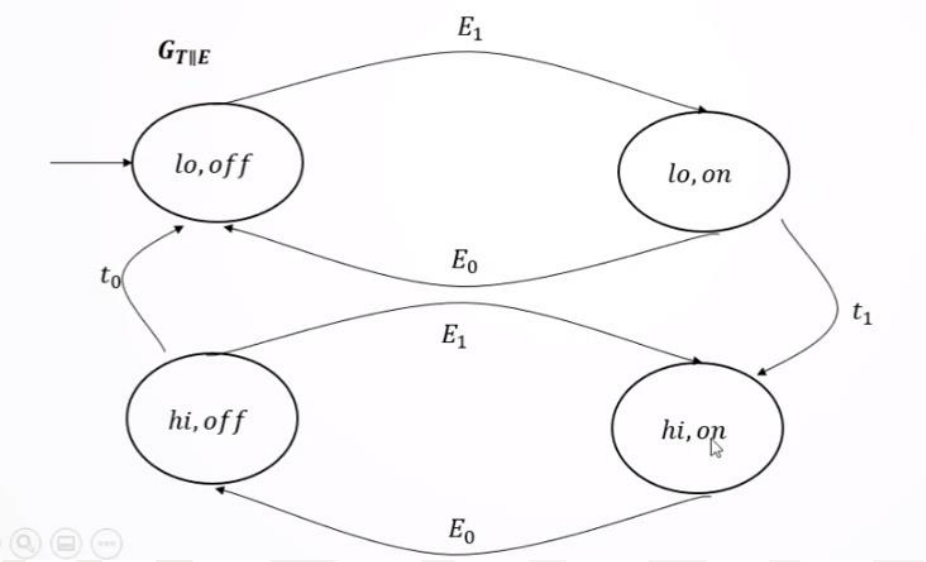


Árbol de evolución de la red de Petri $N_T||N_E$

El árbol de evolución de $N_T||N_E$ vendría dado por:



Como se aprecia el árbol de evolución recorre el mismo camino que el diagrama de autómatas. Por lo tanto, la red de Petri $(N_T||N_E)$ es equivalente al autómata $G_{T||E}$.



Conclusión

Como hemos visto durante el desarrollo del sensor observamos que dentro de la teoría de autómatas estos son autómatas finitos, los cuales son un poco mas sencillos de desarrollar, recordemos que para la realización del sensor y la resistencia es muy importante trabajar de forma separada y ordenada, como nos centramos en la metodología ágil fue la forma mas simple para realizar el ejemplo, con ayuda de esto y la segmentación, fue de gran utilidad ya que esto fue repartido entre los miembros del equipo.

Podemos concluir que el sensor de temperatura y su resistencia, son totalmente compatibles con el modelo computacional Redes de Petri, como equipo concluimos que es un modelo muy entendible y bastante confiable a la hora de trazar un problema computacional, ya que cuenta con una representación matemática que es bastante amable a la hora de plantear el problema.

También podemos decir que este problema nos ofreció un modelo secuencial el cual por el desarrollo nos dimos cuenta que este sensor solo puede contar con 2 estados en especifico los cuales siempre serán uno de tras de otro y no podrán cambiar su orden.

Como estudiantes pudimos observar y practicar de manera más especifica la teoría de autómatas además del modelo computacional que nos fue asignado, los cual se realizo de manera satisfactoria.

Referencias

Davis, M. (12 de 06 de 2021). *Teoría de la computabilidad*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computabilidad

Dean, K. (25 de 05 de 2021). *Teoría de la computación*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computaci%C3%B3n

García, A. A. (15 de 03 de 2017). *MODELOS COMPUTACIONALES*. Obtenido de utem: http://www.informatica.utem.cl/~mcast/CCOMPUTACION/Introduccion/ModelosComputacionales.pdf

N, A. (15 de 04 de 2019). *Modelo Funcional*. Obtenido de Itam: http://ftp.itam.mx/pub/alfredo/OBJETOS/OMT/Modfun.pdf

Patterson, D. A. (04 de 06 de 2021). *Computación concurrente*. Obtenido de xcv Wiki: https://es.xcv.wiki/wiki/Concurrent_computing

R., J. (15 de 07 de 2019). *Modelo de computación*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Modelo_de_computaci%C3%B3n

Sanz, F. (26 de 12 de 2020). *Modelos de Secuencia*. Obtenido de The Machine Learners: https://themachinelearners.com/modelos-secuencia/#:~:text=Los%20modelos%20de%20secuencias%20%28en%20ingl%C3%A9s%20sequence%20models%29,no%20lo%20confund%C3%A1is%20con%20las%20series%20temporales%20%29

Sergio, A. (12 de 05 de 2019). *TEORÍA DE LA COMPUTACIÓN*. Obtenido de ICC: https://icc.fcen.uba.ar/teoria-de-la-computacion/