HERRAMIENTA BASADA EN REDES DE PETRI PARA DISEÑO DE SUPERVISORES DE SISTEMAS DE EVENTOS DISCRETOS

MONOGRAFÍA

Cristhian David Buchely
Fausto Ruiz Coque

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELECOMUNICACIONES
DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL
POPAYÁN
2012

HERRAMIENTA BASADA EN REDES DE PETRI PARA DISEÑO DE SUPERVISORES DE SISTEMAS DE EVENTOS DISCRETOS

Monografía

Cristhian David Buchely
Fausto Ruiz Coque

Director
PhD. Carlos Alberto Gaviria López

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL
POPAYÁN
2012

Nota de Aceptación

_____

_____

_____

Director _____
PhD. Carlos Alberto Gaviria López

Jurado _____

Jurado _____

Fecha de sustentación: Popayán, 2012

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

El presente trabajo adopta el formalismo de redes de Petri (PN's), por su representación gráfica y soporte matemático, para el modelado de sistemas dinámicos de eventos discretos (DEDS's) y el diseño de controladores de eventos discretos (DEC's) o supervisores. Para el diseño del DEC se elegirá un método formal, basado en el formalismo de PN's, que permite sintetizar el DEC, teniendo en cuenta la naturaleza de los DEDS's (no deterministas, concurrencia, sincronización, conflictos, eventos controlables y observables, no controlables y observables, no controlables y no observables). Seguidamente se presenta una metodología para obtener el código en lenguaje *ladder* para el PLC, la cual fue aplicada para las familias SLC y Micrologix de Rockwell, probándose principalmente en un PLC Micrologix 1500 LRP.

Para el modelado del DEDS se cuenta con la herramienta PIPE la cual es de código libre y cuenta con opciones que permiten realizar el análisis y simulación de la PN obtenida, lo que la hace idónea para el propósito del presente proyecto.

Para el diseño del supervisor se desarrolla una herramienta software denominada CRP, teniendo en cuenta los criterios de la metodología seleccionada para el diseño del DEC. Una vez diseñado y verificado el supervisor, conforme al objetivo de control, la herramienta genera automáticamente el código *ladder* para el software RsLogix 500 de acuerdo a la metodología expuesta.

En la sección de casos de estudio se presentan algunos ejemplos en donde se puede apreciar el potencial de la metodología adoptada para el diseño del DEC. También se verifica en un caso real la metodología y la herramienta desarrollada.

INTRODUCCIÓN

En la actualidad los procesos en la industria deben ser muy eficientes y capaces de adaptarse de acuerdo a las necesidades de consumo de las personas, es por ello que éstos deben tener la capacidad de poderse reconfigurar y adaptar lo más rápido posible para poder satisfacer dichas necesidades.

Los controladores lógicos programables (PLC's) son computadoras de tipo industrial cuya finalidad es el control de los procesos productivos en las empresas. Desde su aparición en la década de los 60’s, el PLC ha facilitado la vida de ingenieros y técnicos de planta, debido a que son sistemas digitales programables que redujeron en gran medida el trabajo y tiempo que tomaba reconfigurar las líneas de manufactura [8].

Gran parte de los procesos productivos en la industria corresponden a DEDS, los cuales se caracterizan por su naturaleza no determinista, compartir recursos (conflicto), ejecutar tareas simultaneas y/o paralelas (concurrencia y sincronización), llevar a cabo acciones por un determinado tiempo, etc. Dichas características no pueden ser modeladas por medio de la teoría de control tradicional la cual basa el comportamiento de los modelos en ecuaciones diferenciales. Por ello la necesidad del estudio de los DEDS bajo una nueva óptica [3].

Los DEDS pueden ser “supervisados” por uno o varios controladores de eventos discretos (supervisor). En la actualidad dichos sistemas son cada vez más complejos, por lo que la necesidad de una herramienta más efectiva para el diseño del DEC, es aún mayor. El supervisor se implementa usualmente con PLC's, no obstante, la obtención del algoritmo de control para su posterior implementación mediante software en PLC, se hace con métodos heurísticos, donde la experiencia del programador es clave para la obtención de la solución [3].

Las PN’s, por su representación gráfica y respaldo matemático, son una opción factible, entre varios métodos formales, para el diseño sistemático de DEC’s. Pero esto no es suficiente para el diseño formal y sistemático de supervisores, por ello es necesario un enfoque que tenga en cuenta las propiedades de las PN’s y esté soportado en un método formal para la síntesis del controlador. Además, este método debe satisfacer los requerimientos que exige la naturaleza de los DEDS.

Entre los métodos de diseño de supervisores en el formalismo de PN, se adopta en este trabajo el enfoque basado en restricciones sobre el vector de

marcado, vector de disparo y el vector de Parikh, por lo que en esta monografía se describe detalladamente este método, que permite modelar el DEC mediante una descripción matricial, lo cual facilita la construcción de algoritmos generales.

Después de obtener el DEC, se requiere de un procedimiento para el mapeado de la PN a código *ladder*, para lo cual se deben incluir en la herramienta diseñada las facilidades para la implementación.

1 SISTEMAS DINÁMICOS DE EVENTOS DISCRETOS Y REDES DE PETRI

Con el objeto de contextualizar al lector sobre la temática que se trabajará en el presente proyecto, en este capítulo se presenta, de manera breve, el concepto de DEDS, el cual aparece en varios trabajos de diferentes autores. Además se habla de las redes de Petri partiendo de una breve introducción que justifica porque se ha adoptado este formalismo para el desarrollo del trabajo, luego se da una definición formal acerca de estas, posteriormente se habla de las propiedades que estas poseen y los métodos de análisis, seguidamente se habla de los invariantes, una propiedad que estas poseen y finalmente se aborda el problema de vivacidad de las PN's, donde se presenta como prevenir el bloqueo de éstas mediante el concepto de sifones y trampas.

1.1 Sistemas Dinámicos de Eventos Discretos

Un DEDS (*Discrete Event Dynamic System*) es un sistema que evoluciona de acuerdo a la ocurrencia de eventos. Éstos pueden suceder de manera espontánea y no necesariamente en un intervalo regular de tiempo [1]. En la actualidad es fácil encontrar este tipo de sistemas en campos como las telecomunicaciones, computación y la industria; en aplicaciones tales como redes de comunicación, protocolos de comunicación. En los sistemas de manufactura se aplican, por ejemplo, en la coordinación de vehículos (AGV's), robots, celdas de manufactura etc. Es posible encontrar en estos sistemas, actividades que se den de manera concurrente (al mismo tiempo), sincronizadas, o que entren en conflicto con otras para su ejecución. Dichas actividades pueden describirse como los estados del sistema, a los cuales se asocia un valor lógico o simbólico, en lugar de numérico, y los cuales evolucionan de acuerdo a la ocurrencia de los eventos [1] - [3].

La Figura 1.1 ilustra el concepto de estado en un DEDS, donde a cada estado se ha asignado una etiqueta y el conjunto x de estados es finito y discreto. La evolución de los estados es dinámica por cuanto dependen del tiempo y estados pasados, pero el cambio de un estado a otro depende de la ocurrencia de un evento dentro de un conjunto de eventos también finito y discreto. A cada evento se asocia una etiqueta y su ocurrencia usualmente indica un fenómeno físico que causa el cambio de estado. Por ejemplo, en un sistema de manufactura típico, un evento puede ser "Motobomba encendida", "Maquina dañada", etc. Dependiendo de cómo se considere que puede

ocurrir un evento en el tiempo, los DEDS pueden clasificarse como determinísticos, estocásticos, temporizados o no temporizados.

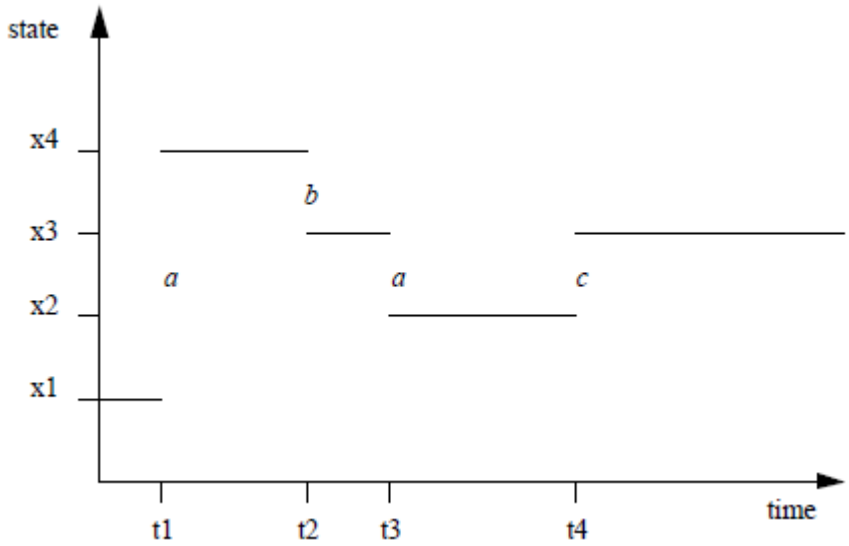


Figura 1.1 Trayectoria de estados de un DEDS [2].

Aunque la representación en la Figura 1.1 permite visualizar la evolución de los estados, un modelo de un DEDS debe permitir no solo una interpretación gráfica de la lógica del sistema a modelar, sino que también debe estar formalmente definido de manera matemática de modo que sus propiedades puedan ser evaluadas computacionalmente y que la evolución del sistema, ante la ocurrencia de eventos, pueda simularse y representarse en un computador.

Entre los formalismos matemáticos para representar un DEDS se encuentran los autómatas o máquinas de estados finitos, las cartas de flujo, las redes de Petri, entre otras.

1.2 Redes de Petri

Las PN's han sido estudiadas en un sinnúmero de textos y disciplinas como las ciencias de la computación, sistemas de comunicación, sistemas de manufactura, etc., donde se consideran una importante herramienta gráfica y matemática. Su uso ha ido incrementando en el modelado de DEDS y en el diseño y análisis de DEC's [4].

Las PN's permiten el modelado de cualquier sistema de eventos discretos [3], siendo su principal ventaja, frente a otros formalismos para representar DEDS, el que éstas permiten modelar y visualizar sistemas que demandan

requerimientos tales como concurrencia, sincronización o la compartición de recursos. Las redes de Petri han demostrado ser muy útiles en el modelado, análisis, simulación y control de sistemas de manufactura. Estas proporcionan modelos muy útiles por las siguientes razones [5]:

- Una PN tiene en cuenta las relaciones precedentes e interacciones existentes entre los diferentes eventos estocásticos, concurrentes y asincrónicos. Además, su representación gráfica ayuda a tener una visión general del sistema.
- El tamaño de buffers y los conflictos pueden modelarse fácil y eficientemente.
- Se pueden detectar los bloqueos del sistema.
- Son una herramienta práctica con un buen desarrollo matemático que las respalda.
- Varias extensiones de las PN's, tales como las PN's temporizadas, PN's estocásticas, PN's coloreadas, permiten el análisis cualitativo y cuantitativo de la utilización de recursos, efecto de fallas (en equipos), evaluación de tasas de rendimiento, etc.
- Los modelos con PN's pueden usarse para el análisis sistemático de sistemas complejos y la construcción sistemática de supervisores.
- Finalmente, las PN's pueden usarse para implementar, en tiempo real, el control de sistemas de manufactura flexibles.

Por lo mencionado anteriormente, las PN's pueden aplicarse a problemas de control de manufactura de alto nivel, donde el supervisor debe coordinar todas las máquinas de la fábrica o célula de manufactura, y bajo nivel, donde el DEC debe ordenar las interacciones entre dispositivos tales como motores, actuadores, válvulas, etc., de tal forma que se cumpla con una tarea específica [3].

1.2.1 Definición

Una red de Petri (PN) marcada es un grafo dirigido, cuya estructura está representada por una quíntupla (***P, T, Pre, Post,*** $\boldsymbol{\mu}$) donde ***P*** y ***T*** son conjuntos discretos que representan los vértices del grafo, los cuales se conocen como *lugares* y *transiciones* respectivamente, $\boldsymbol{\mu}$ es un vector que describe el número de marcas en cada lugar de la red, ***Pre*** es una función que describe el "flujo" que puede darse de ***P*** a ***T,*** también conocido como función de entrada, ***Post*** es una función que describe el "flujo" de ***T*** a ***P,*** conocido como función de salida [3]***.*** Formalmente se tiene:

- **P** = **{**p_1, p_2, p_3,…, p_n**}** es un conjunto finito de lugares,
- **T** = **{**t_1, t_2, t_3,…, t_m**}** es un conjunto finito de transiciones,
- $\boldsymbol{\mu} = \{\mu_1, \mu_2, \mu_3, \dots, \mu_n\}$ es un conjunto de marcas iniciales,

- **Pre(P**x**T)** es un conjunto de arcos dirigidos de **P** a **T** y
- **Post (T**x**P)** es un conjunto de arcos dirigidos de **T** a **P.**

P y **T** son conjuntos disjuntos, por lo que cualquier elemento de **(P** U **T)** se conoce como nodo. Una PN se asume que está conectada, lo que significa que existe al menos un arco dirigido entre nodos.

Los lugares dentro de la PN sirven para representar actividades del sistema, estos pueden almacenar *marcas* las cuales definen el estado actual del sistema. Así, el cambio de un estado a otro en el DEDS es representado dinámicamente en una PN mediante un cambio en la marcación $\boldsymbol{\mu}$. Los lugares en una PN se representan por medio de círculos. Por otro lado, las transiciones permiten que la PN evolucione de un estado a otro mediante el *disparo* de las mismas. Estas se representan en la PN mediante barras. Un arco dirigido desde un lugar a una transición define a éste como *lugar de entrada*. Un arco dirigido desde una transición a un lugar, define a éste como *lugar de salida* [7].

Las marcas en la PN se representan mediante puntos. El marcado $\boldsymbol{\mu}$ de una PN es el mapeo de cada lugar en la PN, el cual representa, mediante un número entero no negativo, la cantidad de marcas que contiene cada lugar [3]. Un arco con peso D_{ij}^{+} desde la transición *j* al lugar *i* implica que al dispararse la transición *j* el lugar *i* recibirá D_{ij}^{+} marcas. Un arco con peso D_{kj}^{-} desde el lugar *k* a la transición *j* implica que el lugar *k* debe contener al menos D_{kj}^{-} marcas para que la transición *j* se dispare, una vez ésta se dispare, el lugar *k* perderá D_{kj}^{-} marcas. De este modo, para que una transición pueda dispararse, todos los lugares de entrada deben contener un número mínimo de marcas [7]. Las transiciones que cumplen esta condición están habilitadas y son libres de dispararse (el disparo depende de igual manera de la ocurrencia del evento el cual es de carácter no determinista) al hacerlo, todos sus lugares de entrada pierden un número determinado de marcas, el cual depende del peso de los arcos definido en ***Pre,*** y todos sus lugares de salida ganan un número de marcas, el cual depende del peso de los arcos definido en ***Post*** [8]***.***

El marcado inicial de una PN se denota como μ_0**.** En la Figura 1.2 se muestra una red de Petri marcada sencilla con dos lugares, dos transiciones y cuatro arcos.

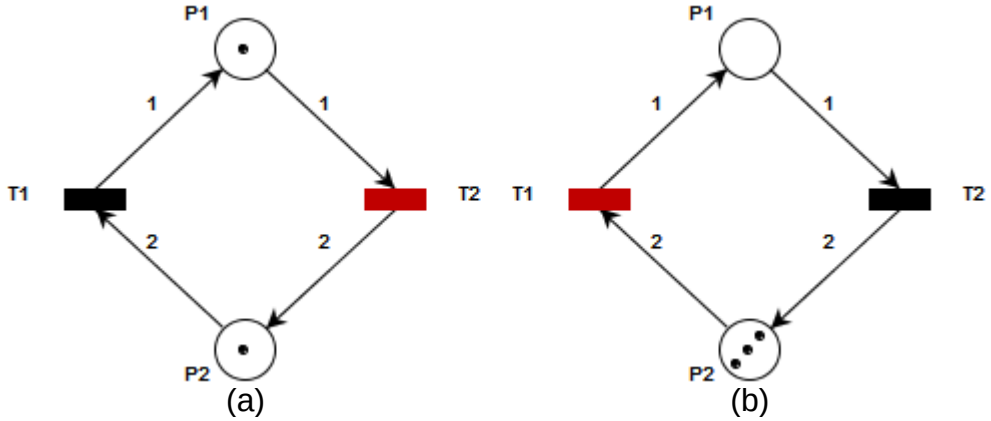


Figura 1.2 Red de Petri sencilla: a) Marcado inicial. b) Marcado posterior al disparo de T_2.

p_1 y p_2 son los lugares, t_1 y t_2 son las transiciones y las flechas representan los arcos. En la Figura 1.2 a) los dos lugares contienen una marca, solo la transición t_2 está habilitada ya que el lugar de entrada, p_1, contiene una marca, igual al peso asociado al arco de entrada. Por el contrario t_1 está inhabilitada porque el número de marcas del lugar de entrada, p_2, es menor al peso del arco de entrada. En la literatura se puede encontrar que los arcos no tengan asociado un número, lo cual significa que el peso del arco es uno. Cuando t_2 se dispare el marcado de la PN será como se muestra en la Figura 1.2 b). Para el nuevo marcado de la PN la transición habilitada es t_1, lo que la hace libre de dispararse, cuando suceda el disparo, el nuevo marcado de la PN será como el marcado inicial.

1.2.2 Propiedades de las redes de Petri

Las PN's como herramienta gráfica y matemática poseen numerosas propiedades, estas se dividen según el marcado inicial, conocidas como dinámicas, y según la estructura, conocidas como estáticas. A continuación se mencionan las más importantes para una PN con *n* lugares y *m* transiciones [9].

1.2.2.1 Propiedades estáticas

Las propiedades mencionadas a continuación solo conciernen a PN's puras y ordinarias. Una PN se dice pura si ningún lugar es a su vez, entrada y salida de una misma transición. Una PN se dice ordinaria, si el peso de todos los arcos es unitario.

- **Vivacidad estructural:** Se dice que una PN es viva estructuralmente si esta cuenta con un marcado inicial $\boldsymbol{\mu_0}$.

- **Controlabilidad:** Una PN es controlable si desde un marcado inicial $\boldsymbol{\mu_0}$ se puede alcanzar cualquier marcado $\boldsymbol{\mu_k}$. Si una PN con *m* lugares es completamente controlable, se tiene que: Rango (D[1]) = *m.*
- **Limitado estructural:** Se dice que una PN es limitada estructuralmente si es limitada (el número de marcas en todo lugar siempre es menor o igual a un límite) para cualquier marcado inicial finito $\boldsymbol{\mu_0}$**.**
- **Conservabilidad:** Se dice que una PN es conservativa (o parcialmente conservativa) si existe un entero positivo *x(p)* para cada (o algún) lugar p tal que la sumatoria de las marcas, $\boldsymbol{\mu_0^T x = \mu^T x,}$ sea constante para cada marcado posterior a $\boldsymbol{\mu_0}$**.** La ecuación anterior define un invariante de lugar.
- **Repetitividad:** Una PN se dice que es repetible si a partir de un marcado $\boldsymbol{\mu_0}$ y una secuencia de disparos **σ** desde $\boldsymbol{\mu_0}$ hasta $\boldsymbol{\mu_k}$ las transiciones definidas en **σ** se disparen infinitamente.
- **Consistencia:** Se dice que una PN es consistente si existe un $\boldsymbol{\mu_0}$ y una secuencia de disparos **σ** desde $\boldsymbol{\mu_0}$ hasta $\boldsymbol{\mu_k}$ tal que cada transición definida en **σ** se dispare al menos una vez.
- **S – y T – Invariantes:** Un *n*-vector *x* (o *m*-vector *y*) de enteros se llama S-invariante (o T-Invariante) si $D^T x = 0$ (o $Dy = 0$).

Las pruebas de las anteriores propiedades pueden apreciarse en [9].

1.2.2.2 Propiedades dinámicas

- **Alcanzabilidad:** El marcado de una PN cambia con el disparo de una transición habilitada, si la PN es pura, no contiene auto-bucles. Basado en lo anterior se puede decir que un marcado posterior al inicial de la PN se puede alcanzar mediante el disparo conveniente de las transiciones. Formalmente se tiene que un marcado $\boldsymbol{\mu_i}$ es alcanzable desde un marcado inicial $\boldsymbol{\mu_0}$ si existe una secuencia de disparos $\sigma = t_1, t_2, \ldots, t_n$ que cambie el marcado desde $\boldsymbol{\mu_0}$ hasta $\boldsymbol{\mu_i}$, lo cual se escribe así: $\boldsymbol{\mu_0}$ **[σ>**$\boldsymbol{\mu_i}$.
- **Limitada o Acotada:** Se dice que una PN es *k*-limitada o limitada si el número de marcas en cada lugar no excede un umbral *k* finito de marcas para cualquier marcado alcanzable desde $\boldsymbol{\mu_0}$**.**
- **Seguridad:** Una PN es segura si todos sus lugares, en cualquier marcado que se de en la red, no contienen más de una marca, es decir si es 1- limitada.

---

[1] *D* es la matriz de incidencia de la PN la cual está definida en la sección 1.2.3.3.

- **Conservabilidad:** Una PN se dice que es conservativa si el número de marcas en la PN es constante para todos los marcados alcanzables.
- **Vivacidad:** Una PN es viva si es posible una nueva secuencia de disparos σ, sin importar el marcado que haya sido alcanzado. Una red de Petri es viva si sus transiciones lo son. Si un sistema es modelado y controlado con redes de Petri y dicha PN es viva, esto indica que el sistema estará libre de *bloqueos* durante la operación.
- **Reversibilidad:** Una PN se dice que es reversible si $\boldsymbol{\mu_0}$ es alcanzable desde cualquier marcado posterior a $\boldsymbol{\mu_0}$.
- **Persistencia:** Se dice que una PN es persistente si en un momento dado existen varias transiciones habilitadas y el disparo de cualquiera de ellas no inhabilita a las demás.

1.2.3 Análisis de las redes de Petri

En general, existen 3 técnicas para el análisis de las PN, a saber: método basado en el grafo, reglas de reducción y método matemático [9].

1.2.3.1 Método basado en el grafo

Éste método de análisis se divide en dos partes para PN limitadas: el *árbol de alcanzabilidad* y el *grafo de cobertura.* Básicamente los dos métodos encuentran todos los marcados que puede alcanzar la red y pueden aplicarse a los diferentes tipos de PN. Sin embargo, el uso de éste método tiene sus limitaciones cuando se aplica a sistemas muy grandes, debido al aumento en la complejidad computacional y al problema conocido como *explosión de estados,* el cual menciona que el número de marcados que puede alcanzar la red de Petri aumenta exponencialmente con respecto al número de lugares de la misma. Para redes limitadas, el árbol de alcanzabilidad contiene todos los marcados posibles y todas las secuencias de disparo válidas, en cuanto el grafo de cobertura provee todos los posibles marcados y el respectivo disparo de las transiciones entre ellos. En la Figura 1.3 se puede observar una PN con su respectivo árbol de alcanzabilidad y cobertura.

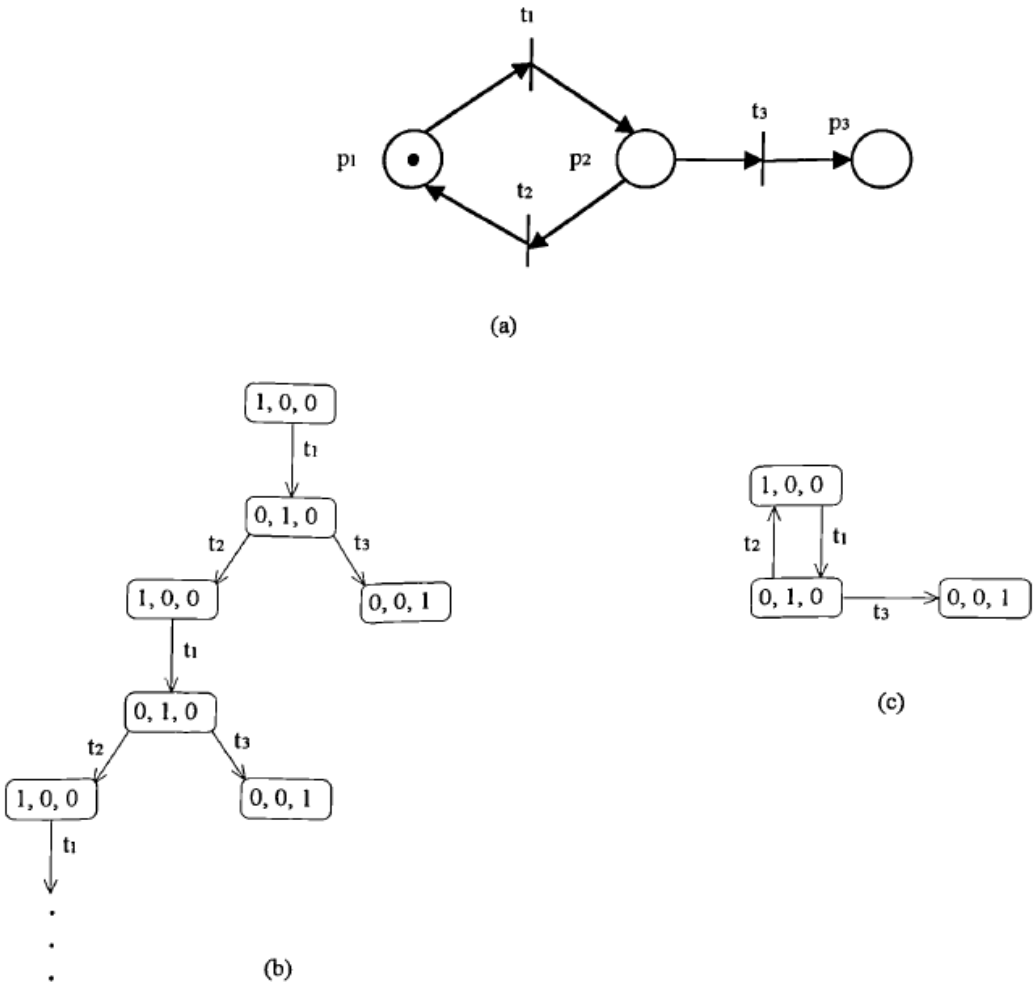


Figura 1.3 (a) Red de Petri. (b) Árbol de alcanzabilidad. (c) Grafo de cobertura [3].

1.2.3.2 Reglas de reducción

Cuando se desea modelar sistemas complejos, que cuentan con un número significativo de estados, la PN obtenida es extensa, lo que dificulta su análisis. Para dar solución a este inconveniente surgen las reglas de reducción las cuales se usan para convertir sistemas complejos en sistemas más simples logrando disminuir el número de nodos de la PN (lugares y transiciones); este método permite a la nueva PN conservar sus propiedades originales. Dichas reglas se ilustran en la Figura 1.4.

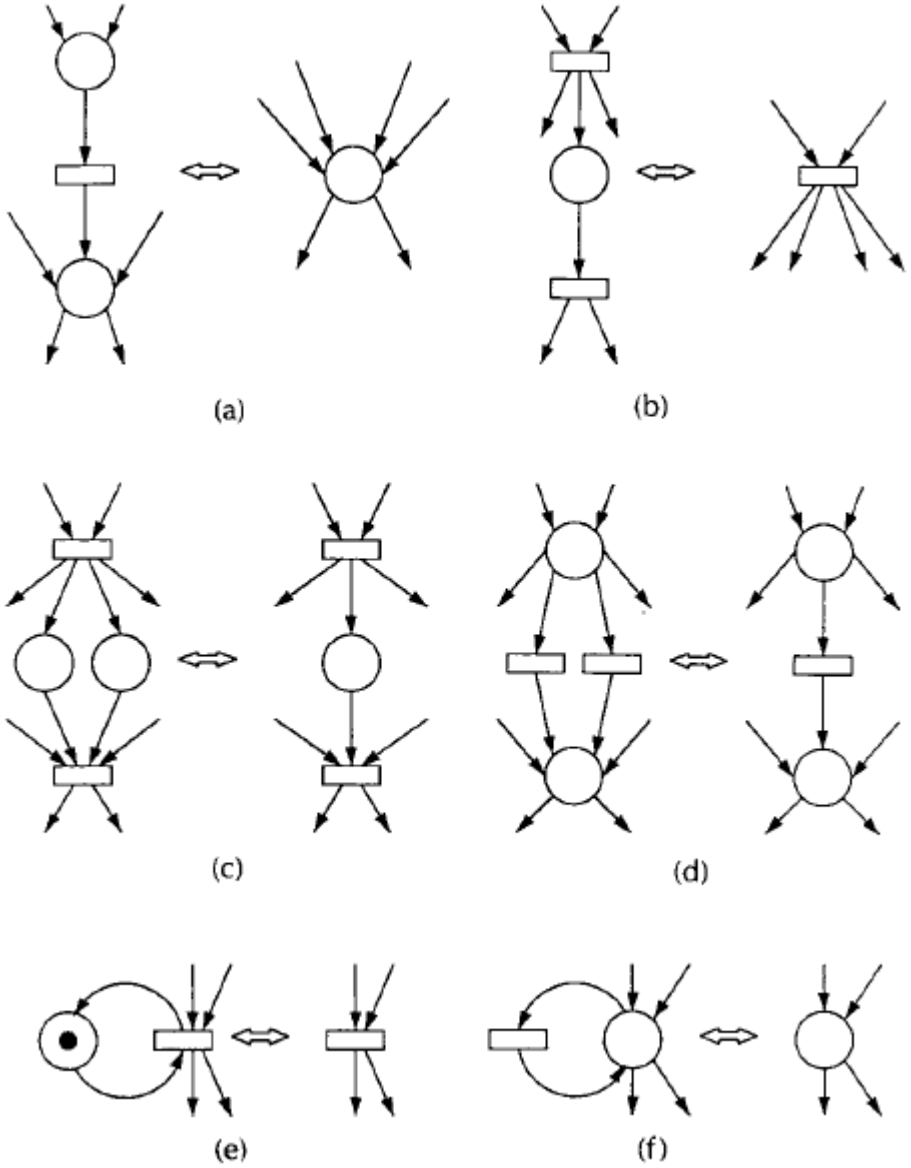


Figura 1.4 Reglas de reducción que preservan vivacidad, seguridad y acotado [9].

a) Fusión de lugares en serie
b) Fusión de transiciones en serie
c) Fusión de transiciones paralelas
d) Fusión de lugares paralelos
e) Eliminación de auto-bucles en lugares
f) Eliminación de auto-bucles en transiciones

1.2.3.3 Matriz de incidencia y ecuaciones de estado

Este método busca representar el comportamiento dinámico de la PN por medio de sistemas de ecuaciones e inecuaciones los cuales se presentan en forma matricial. La parte fundamental del método radica en hallar la matriz de *incidencia* de la PN, la cual especifica las posibles interconexiones que puede haber entre los nodos de la PN.

La matriz de incidencia D_{nxm}, cuyos componentes serán enteros, está definida por:

$$D_{nxm} = D^{+}_{nxm} - D^{-}_{nxm}$$

donde, D^{+}_{nxm} también conocido como matriz del ***Post***, son los pesos de los arcos que conectan las transiciones con los lugares de salida y D^{-}_{nxm}, también conocido como matriz del ***Pre***, son los pesos de los arcos que conectan los lugares de entrada con las transiciones. Todos los elementos de D^{+}_{nxm} y D^{-}_{nxm} son mayores o iguales a cero [7] y [8].

El disparo de las transiciones sucede en pasos discretos. Los n-elementos del vector columna, no negativo, $\boldsymbol{\mu(k)}$ indican el marcado de la PN después del disparo de las transiciones en el paso k-ésimo. Un vector de disparo $\boldsymbol{q(k)}$ es un vector columna no negativo con *m* elementos de valor cero excepto las posiciones que representan las transiciones que se disparan en el paso *k* [7]. En la Figura 1.5 se muestra la relación en el tiempo entre $\mu(k)$ *y* $q(k)$.

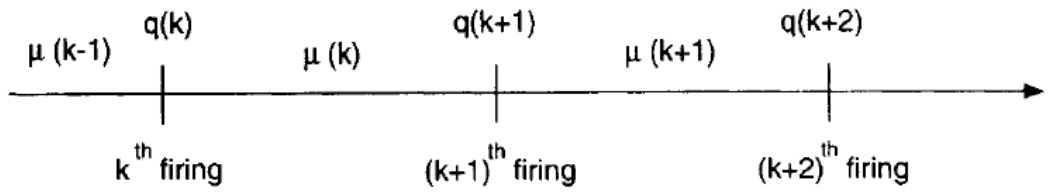


Figura 1.5 Relación entre los vectores de disparo y marcado de una PN [7].

Para determinar que transiciones pueden dispararse en el estado actual de la PN existe la *condición de habilitación.*

$$\mu(k) \geq D^{-} q(k+1) \qquad (1.1)$$

Si la PN dada es pura, no contiene auto-bucles (dentro de una PN un auto-bucle se identifica cuando el lugar de entrada de una transición es el mismo lugar de salida) entonces la condición puede expresarse en términos de la matriz de incidencia. En una PN pura, el disparo de una transición puede añadir o sustraer marcas a un lugar, pero no las dos. De esta manera, las posiciones diferentes de cero de las matrices D^{+} y D^{-} son mutuamente excluyentes. Debido a que los elementos de $D^{+}q$, por definición, son mayores

o iguales a cero y debido a que D^+ y D^- son mutuamente excluyentes, se puede usar la siguiente desigualdad para representar la condición de habilitación [10].

$$\mu + Dq \geq 0 \tag{1.2}$$

Cuando se usen las desigualdades (1.1) y (1.2), debe prestarse atención si el vector q indica el disparo concurrente de varias transiciones ya que la concurrencia no está permitida en todos los casos. Cuando se presenta concurrencia, q debe satisfacer (1.1) o cada vector de disparo, para la respectiva transición a dispararse, debe cumplir por separado la condición (1.2), además la condición (1.2) también se debe cumplir para el vector q completo [10].

Por otra parte, cuando sucede el disparo de las transiciones, el marcado de la PN cambia, el nuevo marcado puede encontrarse con el siguiente sistema:

$$\begin{gathered} \mu(0) = \mu_0 \\ \mu(k+1) = \mu(k) + Dq(k+1) \end{gathered} \tag{1.3}$$

Mediante el uso de las expresiones (1.1), (1.2) y (1.3) se puede describir la dinámica de una PN. A continuación se ilustra con un ejemplo la aplicación de las mismas.

Ejemplo. La PN de la Figura 1.2 tiene la siguiente matriz de incidencia

$$D = D^+ - D^- = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -2 & 2 \end{bmatrix},$$

el marcado inicial de la red es

$$\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

la única transición habilitada para dispararse de acuerdo a (1.2) es

$$q = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

El marcado de la red, posterior al disparo definido por q, será de acuerdo a (1.3), el siguiente:

$$\begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

1.2.4 Invariantes

Una de las más importantes propiedades estructurales de las PN's, son los invariantes. Estos dependen solo de la topología de la PN, permitiendo el análisis de la misma independientemente del proceso que represente [10].

Existen dos tipos de invariantes en una PN: invariantes de lugar e invariantes de transiciones.

1.2.4.1 Invariantes de lugar

Los invariantes de lugar son conjuntos de lugares tal que la suma pesada de sus marcas es constante, para cualquier marcado alcanzable por la PN. Una definición más formal, es considerar un invariante de lugar como un vector x, tal que:

$$x^T \mu_p = x^T \mu_0 \tag{1.4}$$

Donde μ_p y μ_0 son vectores de *n* elementos, tal que *n* es igual al número de lugares de la PN, representando cualquier marcado y el marcado inicial de la misma, respectivamente. El vector x, está conformado de coeficientes enteros, uno para cada lugar, para los cuales se cumple la igualdad.

Todos los invariantes de lugar de la PN, pueden ser hallados resolviendo el siguiente sistema lineal con variables enteras:

$$x^T D = 0 \tag{1.5}$$

un vector x que cumpla la anterior condición, implica que para cualquier vector de disparo q:

$$x^T \mu(k+1) = x^T \mu(k) + x^T D q(k)$$

$$x^T \mu(k+1) = x^T \mu(k)$$

cumpliéndose la propiedad del invariante.

Los invariantes son unas de las principales herramientas en el desarrollo de este proyecto.

1.2.4.2 Invariantes de transición

El análogo de los invariantes de lugar son los invariantes de transiciones [10], estos se definen de manera similar, con la diferencia que en este caso estos cumplen:

$$Dy = 0$$

si y define un conjunto de transiciones de la red, las cuales son disparadas sin importar el orden, esto implica que:

$$\mu(k+1) = \mu(k) + Dy$$
$$\mu(k+1) = \mu(k)$$

De manera que el marcado de la PN volvería a su estado inicial después de finalizar el disparo de todas las transiciones en y. Sin embargo, la propiedad anterior no toma en cuenta el estado de las transiciones, ya que estas pueden estar deshabilitadas para algún marcado intermedio, o simplemente esta pudiese estar bloqueada impidiendo terminar la secuencia.

1.2.5 Sifones y trampas

Los sifones y las trampas son elementos que dependen de la topología de la red y son independientes del marcado. Un sifón es un conjunto de lugares, el cual una vez pierde todas sus marcas permanece vacío, por el contrario, una trampa es un conjunto de lugares, el cual una vez adquiere al menos una marca nunca estará vacía desde ese momento en adelante. [11].

Para la definición de sifón, se hará uso de la notación de pre-set y post-set, entendiéndose el símbolo $\bullet p$, como el conjunto de transiciones con arcos de entrada al lugar p (pre-set de transiciones), y el símbolo $p\bullet$ como el conjunto de transiciones con arcos de salida desde el lugar p (post-set de transiciones). Para un conjunto de lugares, $S \in P$, la notación $\bullet S$ y $S\bullet$, se aplica a las todas las transiciones con entradas y/o salidas a cualquier lugar que pertenezca a S.

Un sifón es un conjunto de lugares $S \in P$, tal que [11]:

$$\bullet S \subseteq S\bullet$$

Lo anterior indica que todas las transiciones de entrada a S son también transiciones de salida. Si S no tiene transiciones de entrada, también se hace válida la definición.

De manera similar, un conjunto de lugares $S \in P$, es una trampa si cumple [11]:

$$S\bullet \subseteq \bullet S$$

Lo anterior indica que todas las transiciones de salida son también transiciones de entrada. Si S no tiene transiciones de salida, la definición también es válida.

Un sifón S es mínimo si no existe otro sifón P, tal que $P \subset S$ [10].

Una trampa S, es mínima si no existe otra trampa P, tal que $P \subset S$ [10].

Los sifones y trampas son ampliamente estudiados en la literatura [12] - [14], su utilidad radica en el análisis de la vivacidad y la prevención del bloqueo en las PN's, por lo que su cálculo es de gran importancia para la implementación de supervisores.

Varios algoritmos para calcular sifones y trampas han sido propuestos en la literatura [15] – [19]. Aunque la herramienta seleccionada para el desarrollo de este proyecto (PIPE) permite hallar los sifones mínimos y trampas mínimas de una PN, en el Anexo C se exponen brevemente tres métodos que se encuentran referenciados en la literatura [17] – [19], los cuales son implementados en la herramienta desarrollada en este proyecto.

Métodos de cálculo para los problemas de sifones pueden encontrarse en [20] y [21].

1.2.6 Sifones y prevención del bloqueo

Una buena política para evitar el bloqueo en una PN consiste en evitar que todos los sifones pierdan todas sus marcas. Todo bloqueo está asociado a un sifón insuficientemente marcado [22], este enfoque es ampliamente trabajado en la literatura [11] – [23]. Sin embargo, no es necesario ocuparse de todos los sifones, por ejemplo, considérese la siguiente PN.

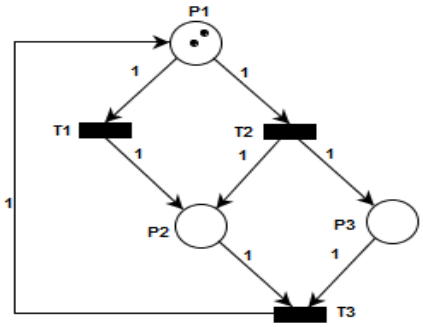


Figura 1.6 Red con bloqueo

La anterior red contiene los sifones $\{p_1, p_3\}$,$\{p_1, p_2\}$ y $\{p_1, p_2, p_3\}$ y trampas $\{p_1, p_2\}$ y $\{p_1, p_2, p_3\}$. No obstante, los dos últimos sifones contienen la trampa

$\{p_1, p_2\}$, el tercero contiene también la trampa $\{p_1, p_2, p_3\}$, además la red contiene un invariante que cubre los lugares p_2 y p_1, de esta manera estos dos sifones están prevenidos de perder completamente sus marcas, cuando esto sucede se dice que el sifón está *controlado*.

Definición 1.6 un sifón se dice **controlado**, cuando una trampa y/o un invariante evitan que aquél pierda todas o las suficientes marcas para bloquear la red.

La red de la Figura 1.6 se bloqueará solo si el sifón $\{p_1, p_3\}$ queda desmarcado, este sifón se dice *no controlado*.

A continuación se lista una serie de condiciones para que una trampa controle un sifón [10].

1. El sifón debe abarcar la totalidad de la trampa.
2. La trampa no puede estar vacía.

Para que un invariante x controle un sifón S, se deben cumplir las siguientes condiciones [10].

1. $x_i > 0 \rightarrow p_i \in S$
2. $x_i < 0 \rightarrow p_i \notin S$
3. $x^T \mu_{p0} > 0$

En el caso anterior, los invariantes de la red que cubren los lugares p_1 y p_2 son:

$$x = k[1\ 1\ 0], k > 0, k \in \mathbb{Z}.$$

En muchos casos prácticos, no es necesario ocuparse de todos los sifones no controlados de la red, por ejemplo, considérese la siguiente red.

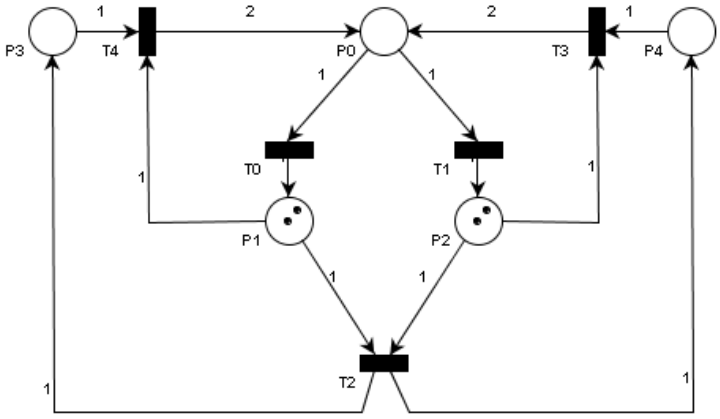


Figura 1.7 Red con sifones no controlados

La anterior red contiene los sifones $\{p_0, p_2, p_3\}$, $\{p_0, p_1, p_4\}$, $\{p_0, p_1, p_2\}$,$\{p_0, p_1, p_2, p_4\}$ y $\{p_0, p_1, p_2, p_3\}$, todos no controlados. Observe que solo los tres primeros sifones son mínimos, si se garantiza que estos no pierdan sus marcas implica que los dos últimos tampoco perderán sus marcas. Debido a la anterior característica, en la mayoría de los casos solo será preciso controlar los sifones mínimos.

En la siguiente sección se presentara un método de control que puede ser utilizado para el control de los sifones no controlados.

2 CONTROL SUPERVISOR

En este capítulo se presenta la técnica para el cálculo del supervisor, basada en restricciones sobre el vector de marcado, vector de disparo y el vector de Parikh, en la cual se fundamentará la herramienta desarrollada en este trabajo. Inicialmente se da la definición y enfoques del control supervisor, luego se aborda el método de control supervisor basado en invariantes de lugar donde se habla de los antecedentes de éste y como se calcula el controlador a partir de dos clases de restricciones que se pueden imponer al modelo del sistema en lazo abierto. Posteriormente se abordan el caso de las transiciones no controlables, no observables, el problema de admisibilidad de las restricciones en este caso, y de tres métodos que permiten transformar las restricciones para que cumplan con los criterios de admisibilidad. Se presenta un ejemplo que aplica el método teniendo en cuenta el problema de las transiciones no controlables y el problema de bloqueo de la PN. Después se habla de las restricciones lineales vectoriales generales, que son la ampliación del método de invariantes de lugar, se muestra cómo se calcula el DEC bajo este nuevo método para las dos clases de restricciones, se habla de los criterios de admisibilidad de las restricciones generales y por último se muestra un ejemplo donde se aplica el método.

2.1 Definición

Los sistemas de eventos discretos son sistemas dinámicos que evolucionan según la ocurrencia de eventos, los cuales suceden de manera abrupta y no necesariamente en un intervalo regular de tiempo. En un proceso, un evento puede ser: daño de una máquina, llegada de un producto, inicio o finalización de una tarea etc. Los DEDS surgen en aplicaciones como manufactura, robótica, tráfico vehicular, entre otras, donde para conseguir los objetivos de estas aplicaciones es necesario llevar a cabo un determinado número de etapas y en cierta forma ordenada [1]. Para que dichas etapas se den de manera ordenada, es necesario el diseño de un controlador para (DEDS). Varios autores y desarrolladores han propuesto distintas teorías y su posible aplicación a diferentes problemas. Ramadge y Wonham, en [1] propusieron uno de los primeros estudios en esta área. En estos modelos el tiempo no está involucrado y se supone que los eventos ocurren espontáneamente. Ellos también abordan el problema de los estados prohibidos, los cuales son estados no deseados o donde se incumplen las especificaciones del problema. El problema del control supervisor es evitar que el sistema entre en estos estados, y para ello este debe seleccionar de entre el conjunto de eventos controlables (aquellos cuya ocurrencia puede forzarse a voluntad),

los que eviten llegar a estados prohibidos, pero teniendo en cuenta que, sin embargo, en cualquier momento pueden ocurrir eventos no controlables (dictados por la naturaleza física del problema, tal como el daño de una máquina o disturbio en cualquier parte del proceso), cuya ocurrencia no se puede impedir. El enfoque establecido por los autores, involucra el modelado en autómatas finitos (FSM), de la planta de eventos discretos (sistema) y del supervisor de eventos discretos (controlador). La planta y el controlador tienen un conjunto de alfabeto idéntico el cual se divide en controlable y no controlable, entonces el comportamiento del modelo se describe como un lenguaje que "habla" el sistema que está soportado en un alfabeto (los eventos). El lenguaje se conforma por palabras (combinaciones de eventos) [3].

El control supervisor es una estructura unificada para el control de DEDS, el cual se basa en lenguajes formales que permiten diseñar, modelar y resolver las especificaciones dadas para un problema de control con algoritmos estándar. Dichas especificaciones se dividen en dos categorías: *problema de estados prohibidos,* en donde las especificaciones de control se expresan como condiciones prohibidas que deben evitarse [24], y *problema de secuencia deseada,* en donde las especificaciones de control se expresan como una secuencia de actividades que deben cumplirse mientras se evita la ocurrencia de secuencias o actividades no deseadas [25].El supervisor hallado se espera que cumpla con las siguientes condiciones:

- Evite que el sistema entre en bloqueos.
- Evite que el sistema alcance estados prohibidos.
- Sea máximamente permisivo esto es, que permita la ocurrencia del conjunto más grande posible de secuencias de eventos que cumplen las restricciones anteriores.

2.2 Enfoques

Para el diseño de DEDS mediante PN hay principalmente dos enfoques propuestos según [3]: *supervisor mapeado,* cuya política de control se calcula de manera eficiente mediante un controlador on-line que hace las veces de una función de realimentación del marcado del sistema, y s*upervisor calculado*, donde la política de control del sistema está representada en la estructura de la PN. Mediante este enfoque las acciones de control se determinan de forma rápida, ya que no es necesario realizar cálculos on-line de forma separada, y además el mismo algoritmo puede ejecutar tanto la PN del sistema como la del supervisor. El modelado del

sistema en lazo cerrado puede hacerse de manera estándar [26]. En el presente proyecto se usa el segundo enfoque.

Además del problema de los estados prohibidos y de las secuencias deseadas, una clase de especificación conocida como *GMEC (generalized mutual exclusion constraints),* es considerada en [27], la cual es útil para condicionar el uso concurrente de recursos finitos. Un enfoque clásico para el modelado y control de sistemas complejos de eventos discretos es considerar al sistema como la interacción de subsistemas. Teniendo en cuenta las tareas particulares que debe cumplir el sistema, y la manera en que se interconectan los subsistemas, se debe imponer ciertas restricciones sobre cómo debe comportarse el sistema total. Una GMEC limita el número de marcas que debe contener un subconjunto de lugares de la PN [3]. En [27] se muestra cómo puede implementarse una GMEC mediante un lugar llamado *monitor*. Una ley de control máximamente permisiva para un conjunto de restricciones siempre se puede implementar por medio de un conjunto de monitores.

Sea $\langle N, \mu_{p0} \rangle$ la PN de un sistema con *n* lugares y marcado inicial μ_{p0}. Una restricción (l, b) define un conjunto de marcados validos:

$$M(l, b) = \{\mu_p \in \mathbb{z}^n, \mu_p \geq 0 | l^T \mu_p \leq b\}$$

Donde μ_p es el marcado del sistema y la ecuación algebraica lineal $l^T \mu_p \leq b$ define un invariante de lugar.

Los monitores son un conjunto de lugares que se conectan a las transiciones del sistema mediante arcos. Cada GMEC requiere un lugar de control y una condición inicial (marcado). En [28] se mencionan los siguientes resultados sobre controladores basados en monitor:

- Un monitor asegura que la proyección del marcado alcanzable por el sistema controlado, sobre el sistema no controlado, satisface la restricción dada.
- La proyección del marcado potencialmente alcanzable del sistema controlado es igual al marcado permitido potencialmente alcanzable del sistema no controlado.
- El monitor restringe de forma mínima el comportamiento de la planta, solo inhibe el disparo de transiciones, controlables, que conducen a marcados prohibidos.
- La vivacidad de la PN no está garantizada mediante la implementación de supervisores basados en monitor. Además, dado un marcado inicial cualquiera, es posible que no se alcancen todos los marcados permitidos, y aun, no es posible retornar al marcado inicial,

pero esto no se debe a la implementación del supervisor mediante monitor, si no al carácter estricto de la restricción impuesta.

2.3 Control Supervisor Basado en Invariantes de Lugar

El control de DEDS basado en invariantes de lugar es similar al enfoque de monitores y se menciona originalmente en [29]. El objetivo de control es hacer cumplir las restricciones impuestas sobre el sistema de la forma.

$$l^T \mu_p \leq b \tag{2.1}$$

donde l es un vector de enteros y b es un escalar entero. Las restricciones de la forma (2.1) son útiles para enfrentar el problema de los estados prohibidos del sistema, y de igual manera sirven para implementar las GMEC's.

Por otra parte, este método de diseño de DEC's es un enfoque muy útil el cual conlleva a la obtención, de manera simple, del supervisor que permite cumplir con los requerimientos de control de la planta, además, la implementación del DEC se puede hacer de manera sencilla. A continuación se nombran algunas ventajas [10]:

1. El método de diseño es transparente por la teoría, análisis e implementación en la que se basa el concepto de invariantes de lugar.
2. El resultado del controlador y en consecuencia el sistema controlado, se describen mediante PN's.
3. Existe una amplia variedad de herramientas para el análisis gráfico y matemático, por ende, la verificación del diseño se hace de manera directa.
4. El supervisor admite que se presente concurrencia en la PN.
5. El método de diseño tiene propiedades numéricas que lo hacen ideal para problemas o aplicaciones de reconfiguración de control, donde debido a fallas, el supervisor debe rediseñarse on-line.

Para desarrollar el concepto de invariantes de lugar se trabajará con una PN de *n* lugares y *m* transiciones, la cual se llamará *red de la planta, red del proceso o sistema.* Su respectiva matriz de incidencia se notará como $D_p \in \mathbb{Z}^{nxm}$. La matriz de incidencia del controlador se notara como D_c, cuya PN estará conformada por las transiciones de la PN de la planta y por los lugares que se añadan para hacer cumplir las restricciones impuestas al sistema. La matriz de incidencia del sistema *controlado* se notará como *D* la cual estará conformada por las matrices de incidencia de la planta y del controlador. La

PN del proceso controlado también se conoce como *sistema controlado* o *sistema en lazo cerrado.*

Mediante este método, el objetivo del controlador es hacer que la planta obedezca las restricciones de la forma (2.1). Por ejemplo, la restricción $\mu_3 + \mu_4 \leq 1$, significa que solo uno de los lugares, p_3 o p_4, puede contener una marca en un momento dado.

La anterior restricción puede ser transformada a una igualdad mediante la introducción de una nueva variable no negativa μ_c. ahora la inecuación se transforma en una ecuación la cual queda así: $\mu_3 + \mu_4 + \mu_c = 1$.

En términos generales se tiene:

$$l^T \mu_p + \mu_c = b \tag{2.2}$$

La variable μ_c representa la marcación de un nuevo lugar c que se añadirá a la PN del sistema el cual contendrá las marcas extras para que la ecuación (2.2) se cumpla. Por cada restricción de la forma (2.1) impuesta sobre la PN del proceso, se añadirá un nuevo lugar a la misma, así, el número de lugares adicionales que tendrá el sistema controlado, dependerá del número de restricciones que se impongan al comportamiento del sistema. Las reglas de evolución de estado de una PN aseguran que μ_c es, por definición, no negativa [10]. De esta manera, la imposición de la restricción (2.1) asegura que la suma pesada de las marcas de los lugares involucrados en la restricción será menor o igual a b, lo que implica que el nuevo lugar es limitado y por tanto si la planta de lazo abierto es limitada, la planta controlada también lo será.

Para hacer un estudio general de los invariantes de lugar, todas las restricciones de la forma (2.1) se pueden agrupar en forma matricial de la siguiente manera:

$$L^T \mu_p \leq b \tag{2.3}$$

Donde μ_p es el vector de marca de la planta, $L^T \in \mathbb{Z}^{n_c \times n}$, $b \in \mathbb{Z}^{n_c}$ y n_c es el número de restricciones del tipo (2.1). La lectura de la desigualdad se debe hacer con respecto a los elementos individuales de los dos vectores $L^T \mu_p$ y b, según corresponda y puede concebirse como conjunciones lógicas separadas en una cantidad igual a n_c. Reescribiendo (2.2), se tiene:

$$L^T \mu_p + \mu_c = b \tag{2.4}$$

Donde $\mu_c \in \mathbb{Z}^{n_c}$, es un vector de enteros que representa el marcado de los lugares de control y $b \in \mathbb{Z}^{n_c}$, es un vector de enteros que representa la suma ponderada máxima de las marcas, que deben contener los lugares involucrados en la restricción junto con los lugares de control.

La matriz de incidencia $Dc \in \mathbb{Z}^{n_c \times m}$ contiene los arcos que conectan los lugares del controlador a la PN del sistema. La matriz $D \in \mathbb{Z}^{(n+n_c) \times m}$del sistema controlado está dada por:

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix}, \tag{2.5}$$

Del mismo modo, los vectores de marcado, $\mu \in \mathbb{Z}^{n+n_c}$, y marcado inicial, μ_0, del sistema en lazo cerrado son:

$$\mu = \begin{bmatrix} \mu_p \\ \mu_c \end{bmatrix}; \mu_0 = \begin{bmatrix} \mu_{p0} \\ \mu_{c0} \end{bmatrix} \tag{2.6}$$

La ecuación (2.4) es de la forma (1.4), por eso los invariantes definidos por la ecuación (2.4) sobre el sistema ((2.5) y (2.6)) deben satisfacer la ecuación (1.5), siendo x reemplazada por X. La siguiente ecuación matricial es la ecuación de invariantes de lugar, definidos por la ecuación (2.4).

$$\begin{aligned} X^T D = [L^T \quad I^T] \begin{bmatrix} D_p \\ D_c \end{bmatrix} = 0 \\ L^T D_p + D_c = 0 \end{aligned} \tag{2.7}$$

donde $I \in \mathbb{Z}^{n_c \times n_c}$ es la matriz de incidencia ya que los coeficientes de las variables μ_c en todas las restricciones es 1.

2.3.1 Cálculo del controlador

Si se cumple,

$$b - L^T \mu_{p0} \geq 0 \tag{2.8}$$

Entonces la matriz de incidencia D_c y el marcado inicial μ_{c0} del controlador se calculan a partir de (2.7) y (2.4).

$$D_c = -L^T D_p \tag{2.9}$$

$$\mu_{c0} = b - L^T \mu_{p0} \tag{2.10}$$

Suponiendo que las transiciones a las cuales van dirigidos los arcos desde los lugares de control, son controlables.

Si (2.8) no se cumple, se deben revisar las restricciones a imponer en el sistema ya que estas deben estar dentro del rango de la condición inicial de la planta. Ver prueba en [10].

El controlador calculado por el anterior método tiene los siguientes alcances de control sobre las transiciones del sistema [10]:

- Habilita e inhabilita varias transiciones en la PN de la planta. Si cualquiera de esas transiciones es no controlable, entonces el controlador definido en esta sección puede no ser válido. La sección 2.4 define la transformación que debe aplicarse a las restricciones en este caso.
- Puede controlar transiciones que hacen parte de auto-bucles.
- Entre los lugares de control y las transiciones del sistema puede generarse auto-bucles cuando se use las técnicas de transformación de la PN para implementar restricciones que involucran el vector de disparo, dichas restricciones se verán en la sección 2.6.3.

2.3.2 Restricciones “mayor o igual”

Las restricciones de la forma (2.1) no garantizan que la suma pesada de las marcas de los lugares de la planta involucrados en la restricción sea mayor a cero, de ahí el problema con la vivacidad y los bloqueos de la PN. Por ello en ocasiones es necesario garantizar que un conjunto determinado de lugares de la PN del sistema permanezca marcado y así garantizar su vivacidad. Ver sección 1.2.5 y 1.2.6.

Una restricción como

$$l^T \mu_p \geq b \tag{2.11}$$

garantiza que la suma de las marcas, de los lugares involucrados en (2.11) en cualquier marcado posterior al marcado inicial μ_{p0}, sea como mínimo igual a b [29].

Este tipo de restricciones al igual que (2.1) están basadas en el concepto de invariantes de lugar, por lo que el controlador debe garantizar que la suma pesada de marcas en los lugares involucrados en (2.11) y del lugar de control sea constante. Suponga que se impone la siguiente restricción: $\mu_1 + \mu_3 \geq 2$, quiere decir que el número de marcas, en cualquier estado de la PN, de los lugares p_1 y p_3 debe ser como mínimo igual a dos.

De nuevo esta restricción se transforma a una igualdad mediante la inclusión de una variable, en este caso negativa, $-\mu_c$: $\mu_1 + \mu_3 - \mu_c = 2$.

En términos generales se tiene:

$$L^T \mu_p - \mu_c = b \tag{2.12}$$

El hecho que $-\mu_c$, quien representa los lugares del controlador, sea negativa cambia un poco las ecuaciones de cálculo del controlador

$$X^T D = [L^T \quad -I^T] \begin{bmatrix} D_p \\ D_c \end{bmatrix} = 0$$
$$L^T D_p - D_c = 0 \tag{2.13}$$

de (2.13) se obtiene la ecuación para hallar la matriz de incidencia del controlador.

$$D_c = L^T D_p \tag{2.14}$$

La ecuación del marcado inicial del controlador es:

$$L\mu_{p0} - \mu_{c0} = b \Leftrightarrow$$
$$\mu_{c0} = L\mu_{p0} - b \tag{2.15}$$

Para imponer la restricción, esta debe cumplir con la condición inicial de la PN del proceso.

$$L\mu_{p0} - b \geq 0 \tag{2.16}$$

Este tipo de condiciones se implementan de forma similar a la vista en la sección anterior. En la sección 2.5 se presenta un ejemplo donde se muestra la aplicación de estas restricciones para el control de sifones mínimos.

2.4 Transiciones no Controlables y no Observables

Es común encontrar en un DEDS eventos, los cuales no pueden ser evitados por el supervisor, como daños en la maquinaria, fallas de calidad en la materia prima, fallas de energía, interferencias con otros procesos o distorsiones en los procesos químicos y/o físicos, los cuales no son controlables. Todas estas situaciones son tomadas en cuenta en el modelo del sistema. En un DEDS basado en PN's, los eventos del sistema están directamente relacionados con el disparo de las transiciones, si el supervisor

es capaz de controlar la ocurrencia de un evento entonces este es controlable y es modelado con una transición corriente, para aquellos otros como los descritos anteriormente, es preciso utilizar transiciones no controlables o no observables, las cuales se explican a continuación.

Una transición es *no controlable* si su disparo no puede ser deshabilitado por una acción externa [10].

Una transición es *no observable* si su disparo no se puede, o es muy difícil, medir [10].

Cuando se diseña un supervisor, se debe tener cuidado que este no pretenda inhibir alguna transición no controlable y/o dependa de alguna transición no observable. La anterior tarea no siempre es sencilla, habiendo la necesidad de considerar muchos casos o un grafo de alcanzabilidad muy grande o infinito, lo que es computacionalmente ineficiente debido al problema del crecimiento exponencial de estados [3]. Una solución computacionalmente eficiente al problema anterior, es considerar el siguiente lema:

2.4.1 Condición suficiente para admisibilidad

Un supervisor que debe hacer cumplir $L\mu_p \leq b$ o $L\mu_p \geq b$ tal que [30]:

$$D_c^-(\cdot, t) = 0, \forall t: t \in T_{uc} \text{ y } D_c(\cdot, t) = 0, \forall t: t \in T_{uo} \qquad (2.17)$$

donde $D_c^-(., t)$ son las columnas de la matriz del *Pre* para los lugares de control, $D_c(., t)$ son las columnas de la matriz del *Post* para los lugares de control, T_{uc} es el conjunto de transiciones no controlables y T_{uo} es el conjunto de transiciones no observables.

Si la condición (2.17) se cumple, implica admisibilidad de la restricción L, asumiendo que esta cumple la condición inicial del proceso.

El anterior lema solo establece una condición suficiente más no necesaria, sin embargo, un análisis más profundo será suficiente para el desarrollo del proyecto.

Es fácil determinar que la condición (2.17) establece la no existencia de un arco desde el lugar de control hacia alguna transición no controlable (evitando cualquier intento de inhibición) y la no alteración del marcado del supervisor por alguna transición no observable, en el caso del control supervisor basado en invariantes de lugar la anterior condición se puede extender a:

2.4.2 Condición suficiente de admisibilidad

Sea una PN con matriz de incidencia D, sea una matriz D_{uc} la cual está conformada por los vectores columna de D correspondientes a transiciones no controlables, sea una matriz D_{uo} conformada por los vectores columna de D correspondientes a las transiciones no observables. Considérese los dos casos.

a.) Sea una matriz con transiciones no controlables y no observables descritas por las matrices D_{uc} y D_{uo}, respectivamente, y la condición $L\mu_p \leq b$, entonces si se cumple:

$$LD_{uc} \leq 0 \text{ y } LD_{uo} = 0 \tag{2.18}$$

Entonces L es admisible [10] y [31].

b.) Sea una matriz con transiciones no controlables y no observables descritas por las matrices D_{uc} y D_{uo}, respectivamente, y la condición $L\mu_p \geq b$, entonces si se cumple:

$$LD_{uc} \geq 0 \text{ y } LD_{uo} = 0 \tag{2.19}$$

Entonces L es admisible.

En la siguiente sección se exponen algunos métodos para transformar las restricciones no admisibles, para que puedan ser impuestas en la PN.

2.4.3 Transformación de restricciones

La presencia de transiciones no controlables y/o no observables en una PN puede impedir la implementación directa de una restricción en el sistema, cuando esto sucede, para aplicar dicha restricción se realiza de una manera indirecta. Como fue expuesto en la sección 2.3, cada restricción impide a la PN alcanzar marcas ilegales, por lo tanto hay que evitar que la PN pueda alcanzar alguna marca, en la cual al disparase alguna transición no controlable o no observable lleve el sistema a un marcado ilegal.

Moody [10] y Antsalkis [32], presentan en sus trabajos el siguiente lema:

2.4.3.1 Estructura de trasformación

Sea R_1 y R_2 que cumplen:

$$R_1 \in \mathbb{Z}^{n_c \times n} \text{ Satisface } R_1\mu_p \geq 0, \forall \mu_p$$
$$R_2 \in \mathbb{Z}^{n_c \times n_c} \text{ Matriz diagonal positiva.}$$

Si se cumple $L'\mu_p \leq b'$, donde L', b' son definidos como:

$$L' = R_1 + R_2 L$$
$$b' = R_2(b + \mathbf{1}) - \mathbf{1}$$

Donde $\mathbf{1}$ es un vector de dimensión n_c compuesto de unos. Entonces se cumple $L\mu_p \leq b$.

Demostración:

$$(R_1 + R_2 L)\mu_p < R_2(b + 1)$$
$$R_2^{-1} R_1 \mu_p + L\mu_p < b + 1$$
$$L\mu_p \leq b$$

Para restricciones $l^T \mu_p \geq b$ un resultado similar se obtiene si se define a R_1 de la siguiente manera:

$$R_1 \in \mathbb{Z}^{n_c \times n}, R_1\mu_p \leq 0, \forall \mu_p$$

De esta manera se consigue la desigualdad:

$$(R_1 + R_2 L) > R_2(b + 1)$$
$$R_2^{-1} R_1 \mu_p + L\mu_p > b + 1$$
$$L\mu_p \geq b$$

La importancia del lema anterior consiste en la posibilidad de implementar en la planta un conjunto de condiciones $L\mu_p \leq b$ o $L\mu_p \geq b$ aplicando otro conjunto de transiciones $L'\mu_p \leq b$ o $L'\mu_p \geq b$. El nuevo conjunto de restricciones determinado por L' y b' es más restrictivo que el original, esto significa que el nuevo controlador puede no ser siempre máximamente permisivo [32], cuando esto sucede, se dice que el supervisor es sub-óptimo.

2.4.3.2 Transformación por operaciones de filas de matriz

En [33], se expone una definición bastante completa sobre transformación de restricciones basado en GMEC. En dicho trabajo se propone un algoritmo basado en operaciones por filas para la transformación de restricciones no

admisibles, el algoritmo solo considera las transiciones no controlables, por lo que es preciso extenderlo para transiciones no observables [31], la estructura básica de este algoritmo se presenta a continuación.

Sea una PN con matriz no controlable D_{uc} y matriz no observable D_{uo}, considérese el conjunto de restricciones no admisibles L. Se construye la siguiente matriz:

$$\begin{bmatrix} D_{uc}D_{uo} & I & 0 \\ LD_{uc}LD_{uo} & R_1 & R_2 \end{bmatrix}$$

Los valores de inicio son $R_1 = 0$ y $R_2 = I$. El objetivo del procedimiento es llevar la parte de la matriz denotada por LD_{uc} y LD_{uo} a un valor que implique admisibilidad, tal como fue expuesto en (2.4.2). Solo son válidas las siguientes operaciones:

a.) No es posible (ni necesario) operar sobre las primeras n filas.
b.) No puede existir la suma o resta entre las últimas n_c filas.
c.) Si $L\mu_p \leq b$, es válido únicamente la suma de filas.
d.) Si $L\mu_p \geq b$, es válido únicamente la resta de filas.

Al terminar el procedimiento, se toman las matrices R_1 y R_2 para calcular el nuevo conjunto de restricciones L', b'.

2.4.3.3 Generación de soluciones calculando el núcleo

Otro método para la transformación de restricciones consiste en hallar el núcleo de la siguiente matriz.

$$\begin{bmatrix} D_{uc} & D_{uo} \\ LD_{uc} & LD_{uo} \\ I & 0 \end{bmatrix}$$

Este método es definido de una manera más completa en [34] aunque omitiendo las transiciones no observables, Moody y Antsalkis [10] utilizan el método en un ejemplo ilustrativo.

Al calcular el núcleo, se obtiene una matriz $[R_1 R_2\ \Delta]$ tal que:

$$[R_1\ \ R_2\ \ \Delta]\begin{bmatrix} D_{uc} & D_{uo} \\ LD_{uc} & LD_{uo} \\ I & 0 \end{bmatrix} = 0$$

Donde la parte de la matriz denotada por Δ corresponde a variables de adición.

Si las restricciones cumplen $L\mu_p \leq b$, el objetivo consiste en conseguir por medio de operaciones por filas que todos los elementos de Δ sean positivos o nulos, de esta manera se garantiza $R_1 D_{uc} + R_2 L D_{uc} = (R_1 + R_2 L) D_{uc} \leq 0$ lográndo calcular un conjunto de restricciones L' admisible, además debe asegurarse un $R_1 \geq 0$ y $R_2 > 0$ para cumplir con los requerimientos de (2.4.2). De manera análoga, si las restricciones a imponer son como $L\mu_p \geq b$ se debe conseguir que los elementos de Δ sean negativos o nulos garantizando $R_1 D_{uc} + R_2 L D_{uc} = (R_1 + R_2 L) D_{uc} \geq 0$ y conseguir $R_1 \leq 0$ y $R_2 > 0$. Nótese la no existencia de variables de adición en la columna de transiciones no observables, esto es a que ya el cálculo del núcleo garantiza $R_1 D_{uo} + R_2 L D_{uo} = (R_1 + R_2 L) D_{uo} = 0$, por tanto en el caso de solo existir transiciones no observables bastará asegura un valor adecuado para R_1 y R_2.

Al terminar el procedimiento, se toman las matrices R_1 y R_2 para calcular el nuevo conjunto de restricciones L', b'.

2.4.3.4 Programación lineal entera para generar una solución óptima

Por último, es posible utilizar la programación lineal de enteros (PLE) para calcular la trasformación de las restricciones. Se define el siguiente problema de PLE para realizar la trasformación [10] y [31].

$$\min_R \left(z(R) = R \begin{bmatrix} \mu_{p0} \\ L\mu_{p0} - b - 1 \\ 0 \end{bmatrix} \right)$$

$$cond. \begin{cases} R \begin{bmatrix} D_{uc} & D_{uo} \\ LD_{uc} & LD_{uo} \\ I & 0 \end{bmatrix} = -L[\, D_{uc} D_{uo} \,] \\ R \geq 0 \end{cases}$$

Donde $R = [R_1 \quad R_2' \quad R_3]$,$R_2' = R_2 - \mathbf{1}$ y R_3 es un vector de variables de adición. Nótese que la anterior definición está estructurada para restricciones que cumplen $L\mu_p \leq b$, para restricciones que cumplen $L\mu_p \geq b$ se redefine el problema de la siguiente manera:

$$\min_R \left(z(R) = R \begin{bmatrix} -\mu_p \\ L\mu_{p0} - b - 1 \\ 0 \end{bmatrix} \right)$$

$$cond. \begin{cases} R \begin{bmatrix} -D_{uc} & -D_{uo} \\ LD_{uc} & LD_{uo} \\ -I & 0 \end{bmatrix} = -L[\, D_{uc} D_{uo} \,] \\ R \geq 0 \end{cases}$$

En la solución del problema, puede ser necesaria la transformación a un problema de mínimo general, esto se puede realizar eliminando la variable de adición R_3 y multiplicando por un valor negativo quedando un número de condiciones como desigualdades y otras como igualdades, además, se debe evitar la solución trivial $R = \mathbf{0}$ y generar criterios para limitar las soluciones en caso de que el problema sea ilimitado.

2.5 La Máquina Poco Fiable

A continuación se presenta un ejemplo donde se ilustra el uso del control supervisor basado en invariantes de lugar, así como métodos de transformación de restricciones.

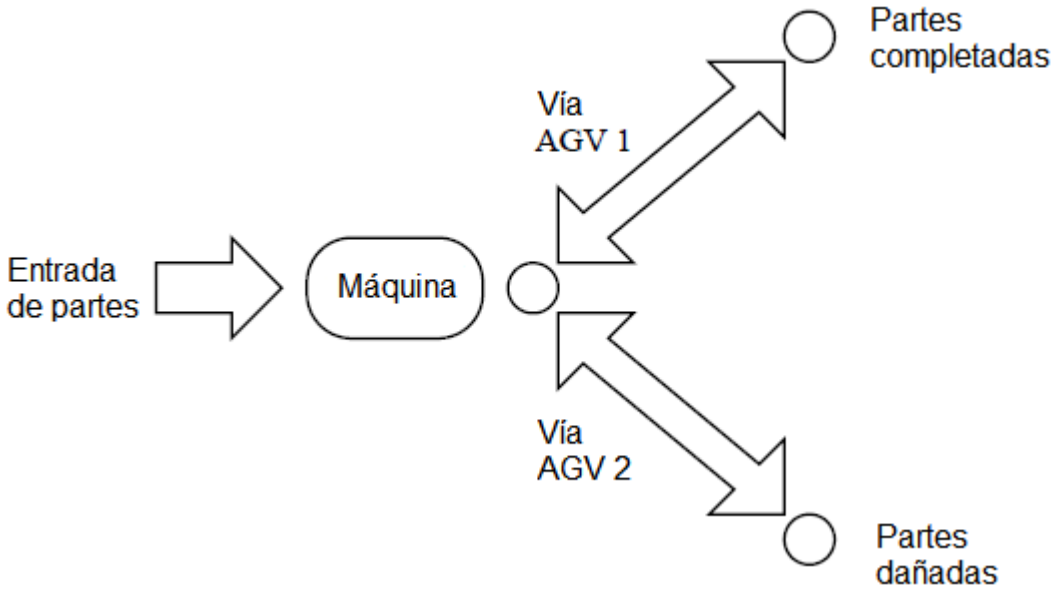


Figura 2.1 Diagrama de operaciones [10]

La Figura 2.1 representa el diagrama de operaciones para una determinada pieza. La máquina toma una piezas desde la cola de entrada y la somete a una serie de operaciones, durante el procesado de la pieza, la maquina puede dañarse y arruinar la pieza en proceso. Cuando esto suceda, un vehículo guiado automáticamente (AGV 2) debe retirar la pieza dañada y ponerla en el sitio de almacenamiento de partes dañadas. Las piezas que son procesadas satisfactoriamente son retiradas por el AGV 1 y puestas en el sitio de almacenaje de piezas completadas. A continuación se muestra el modelo del proceso en PN's.

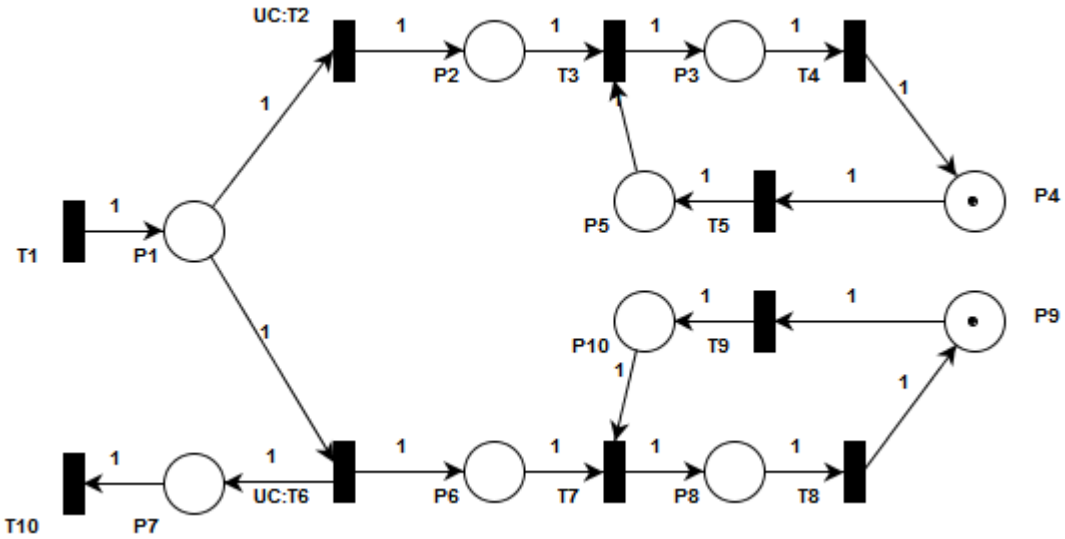


Figura 2.2 Modelo en PN para el proceso de la máquina poco fiable [10]

P1	Máquina procesando una pieza
P2	Pieza esperando su traslado a piezas terminadas
P3	AGV 1 trasladando pieza terminada
P4	AGV 1 pone pieza en piezas terminadas
P5	AGV 1 esperando que la pieza sea procesada
P6	Pieza esperando su traslado a piezas dañadas
P7	Máquina en espera de ser reparada
P8	AGV 2 trasladando pieza dañada
P9	AGV 2 pone pieza en piezas dañadas
P10	AGV 2 esperando pieza dañada

Tabla 2.1 Descripción de los lugares de la PN

T1	Parte removida desde la cola de entrada
UC:T2	Parte procesada satisfactoriamente (no controlable)
T3	Parte tomada por AGV 1 para su traslado
T4	Parte depositada en piezas terminadas
T5	AGV 1 se posiciona para esperar pieza procesada
UC:T6	Fallo de máquina, pieza dañada (no controlable)
T7	Parte tomada por AGV 2 para su traslado
T8	Parte depositada en piezas dañadas
T9	AGV 1 se posiciona para esperar pieza dañada
T10	Máquina reparada

Tabla 2.2 Descripción de las transiciones la PN

La matriz de incidencia de la PN es:

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

2.5.1 Restricciones físicas del proceso

La máquina solo puede almacenar una pieza en el buffer de salida, entonces la primera restricción a imponer sobre el sistema es:

$$\mu_2 + \mu_6 \leq 1$$

Solo un AGV puede acceder a extraer la pieza del buffer. La segunda restricción es:

$$\mu_5 + \mu_{10} \leq 1$$

La tercera restricción es evidente, pues la máquina no puede operar si está dañada:

$$\mu_1 + \mu_7 \leq 1$$

A continuación se muestran las restricciones en forma matricial:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Ahora se chequea la condición de admisibilidad 2.18, correspondiente a transiciones no controlables (T2 y T6), para determinar si se debe transformar alguna restricción:

$$LD_{uc} \leq 0 \rightarrow \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ -1 & 0 \end{bmatrix}$$

La condición no se cumple en la primera restricción impuesta al sistema lo que implica que ésta debe ser transformada. La transformación de la restricción se hace utilizando el método visto en la sección 2.4.3.2, mediante la operación por filas de matriz. Los valores para R_1 y R_2 son:

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} R_2 = I$$

Verifíquese la siguiente ecuación para la matriz R_1 y R_2, que implica la admisibilidad de la nueva restricción:

$$(R_1 + R_2 L) D_{uc} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \end{bmatrix} \leq 0$$

La matriz de incidencia del controlador y su marcado inicial son:

$$D_c = -L' D_p = -(R_1 + R_2 L) D_p = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mu_{c0} = b' - L' \mu_{p0} = R_2(b + \mathbf{1}) - \mathbf{1} - (R_1 + R_2 L)\mu_{p0} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

A continuación se muestra la PN del proceso en lazo cerrado.

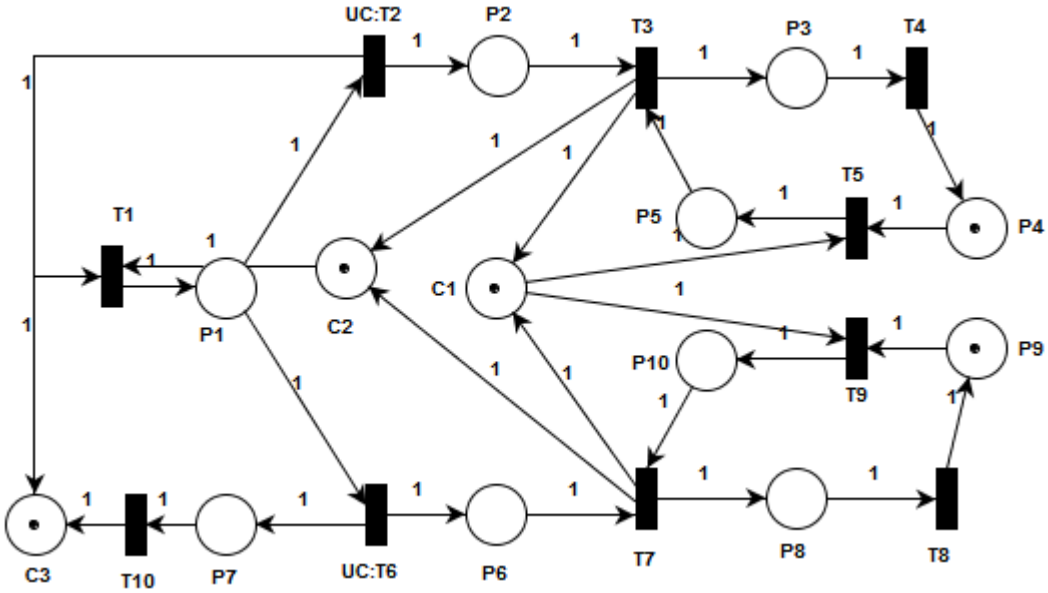


Figura 2.3 PN del proceso controlado

Note que el controlador obtenido, representado por los lugares C1, C2 y C3, no posee arcos de salida hacia transiciones no controlables, lo que muestra la admisibilidad del control.

2.5.2 Prevención del bloqueo

Suponga que antes de definir si la pieza en proceso saldrá buena o mala, el AGV 2 ingresa hasta el buffer de la máquina y sucede que la pieza es bien procesada por la máquina, siendo marcados los lugares P2, P4, P10 y C3. En cuyo marcado la PN se encuentra bloqueada.

Esta situación se presenta porque la PN del sistema controlado contiene sifones los cuales, ante el anterior marcado, impiden la evolución del mismo, por ello deben ser buscados y controlados.

De acuerdo a la sección 1.2.5 y 1.2.6 la PN en lazo cerrado cuenta con siete sifones mínimos pero solo dos no están controlados, {P1, P5, P6, C1, C2} y {P1, P2, P10, C1, C2}, para controlarlos mediante invariantes de lugar, es necesaria la imposición de las siguientes restricciones:

$$\mu_1 + \mu_5 + \mu_6 + \mu_{11} + \mu_{12} \geq 1$$
$$\mu_1 + \mu_2 + \mu_{10} + \mu_{11} + \mu_{12} \geq 1$$

De acuerdo a 2.20, las anteriores restricciones son inadmisibles, por lo que deben ser transformadas. El siguiente es el resultado de la transformación:

$$L' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$b' \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

La matriz de incidencia y el marcado de los nuevos lugares de control son:

$$D_c = -L' D_p = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ -1 & 1 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mu_{c0} = b' - L'^{T} \mu_{p0} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Note que μ_{11} y μ_{12} son el marcado de C1 y C2, respectivamente, en las restricciones impuestas para prevenir el bloqueo del proceso.

La nueva PN del sistema controlado es la siguiente:

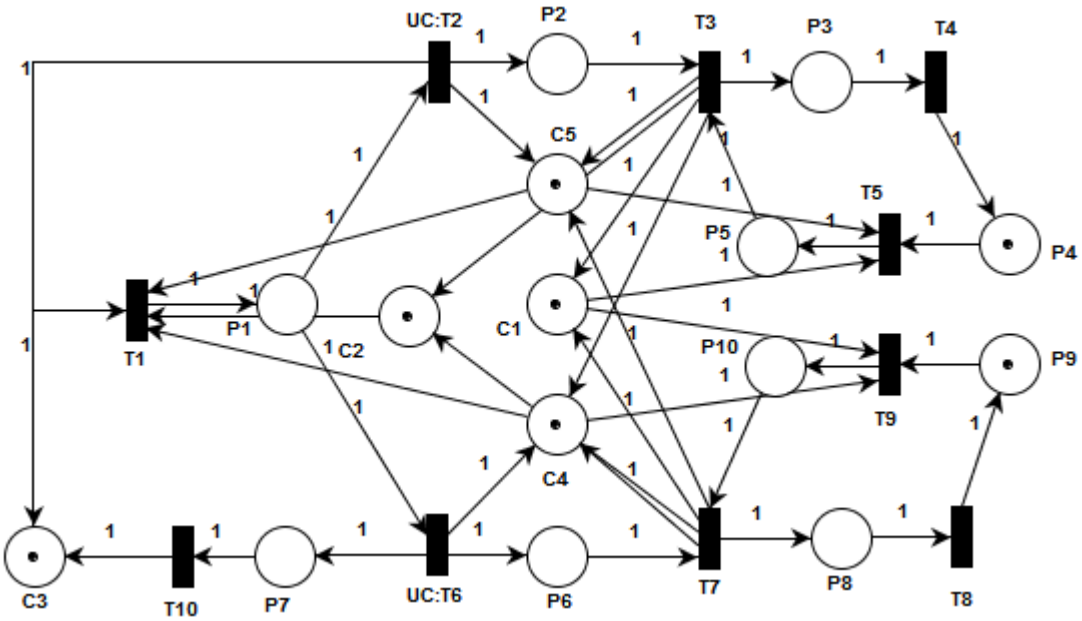


Figura 2.4 PN del proceso controlado sin bloqueo

La PN anterior no está exenta de bloqueo ya que cada vez que se adiciona un nuevo lugar es probable que este, al conectarse a la PN, genere nuevos sifones, de hecho la PN de la Figura 2.4 cuenta ahora con 15 sifones mínimos. Para garantizar que el supervisor permita al proceso comportarse como se desea y evitar que este llegue a bloqueo, es necesario analizar de nuevo los sifones mínimos y controlarlos, pero esto puede convertirse en una tarea complicada e ineficiente de acuerdo a la topología y dimensión de la PN. Por otra parte, cada restricción que se imponga sobre el sistema hace que el controlador obtenido sea menos permisivo.

Sin embargo, una manera de tratar de garantizar que el sistema se comporte como se desea y evite el bloqueo, es hacer menos permisivo el supervisor. Mediante simulación del supervisor de la Figura 2.4 se logró determinar que si se imponen las siguientes restricciones $\mu_5 + \mu_{10} + \mu_{12} \leq 1$, $\mu_2 + \mu_{10} \leq 1$ y $\mu_5 + \mu_6 \leq 1$:

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

el proceso en lazo cerrado se comportara como se desea aunque el supervisor obtenido será menos permisivo.

2.6 Síntesis del Supervisor Aplicando Restricciones Lineales Vectoriales Generales

Las restricciones lineales vectoriales generales (LVGC) son una evolución del control supervisor basado en invariantes de lugar (SBPI). Este tipo de restricciones aparecen en [35] y más adelante en [30]. Las LVGC tienen la siguiente estructura [35]:

$$L\mu + Hq + Cv \leq b \tag{2.20}$$

Donde μ es el vector de marcas, q el vector de disparo y v es el vector de Parikh, el cual actúa como un contador de los disparos de las transiciones de la PN. La Figura 2.5 ilustra la evolución de la red después del disparo de T0 y T1 y el significado de cada vector [32]. La Figura 2.5 a) muestra el disparo de T0 y el marcado previo, el vector de Parikh es cero, pues no se ha disparado ninguna transición antes. En la Figura 2.5 b) se ilustra el disparo de T1, con el nuevo marcado, el vector de Parikh ya registra el disparo de T0. La Figura 2.5 c) ilustra la red en su estado final.

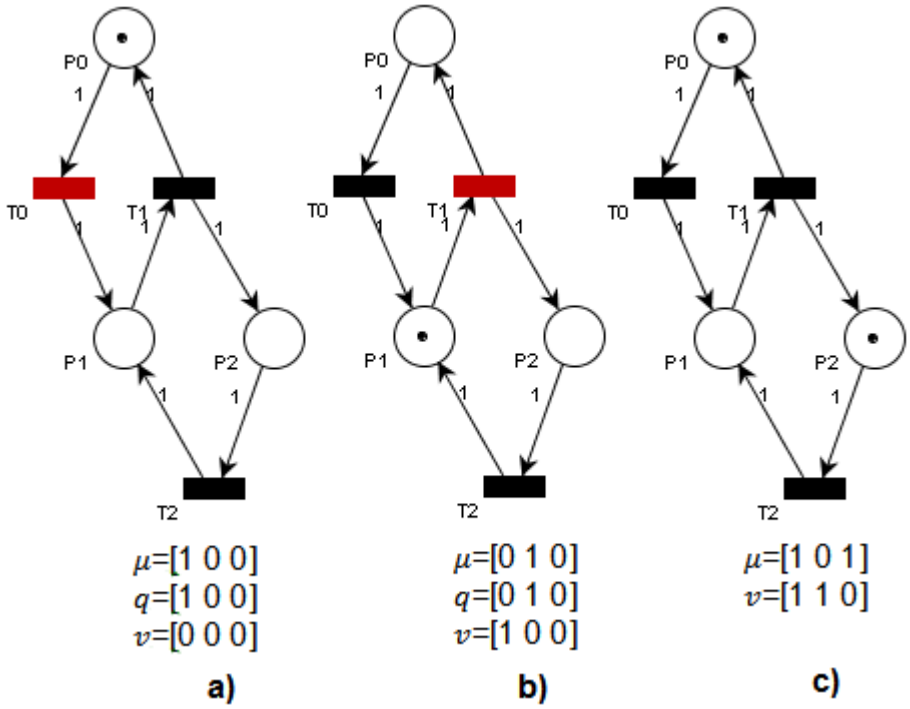


Figura 2.5 Red inicial. b) Red luego del disparo de T0. c) Red final luego del disparo de T1

El término Cv en (2.20) cambia tanto el significado de las restricciones vistas en la sección 2.3, como el concepto del supervisor hallado mediante dichas restricciones. La inclusión del vector de Parikh en (2.20) permite llevar el

historial de los disparos de las transiciones incluidas en (2.20), lo cual hace que el controlador a implementar sea dinámico, a diferencia del supervisor visto en la sección 2.3 el cual se considera estático. Por otro lado, la implementación de restricciones de la forma (2.20) no genera un invariante en la PN del sistema en lazo cerrado [30].

Para hallar la matriz de incidencia del controlador se utilizan las siguientes expresiones:

$$D_{lc}^{+} = \max(0, -LD_p - C)$$
$$D_{lc}^{-} = \max(0, LD_p + C)$$

$$D_c^{+} = D_{lc}^{+} + \max(0, H - D_{lc}^{-}) \tag{2.21}$$
$$D_c^{-} = \max(D_{lc}^{-}, H)$$

En este caso la función max se implementa elemento a elemento de la siguiente manera: Por ejemplo la operación.

$$\mathrm{C} = \max\ (A, B)$$

Sean a_{ij}, b_{ij} y c_{ij}, los elementos en la posición i y j de las matrices A, B y C, respectivamente. Entonces, c_{ij} es igual a b_{ij}, si $b_{ij} > a_{ij}$, en caso contrario c_{ij} es igual a a_{ij}.

El marcado inicial de los lugares de control se obtiene mediante:

$$\mu_c = b - L\mu_0 - Cv_0 \tag{2.22}$$

Las anteriores ecuaciones son válidas si:

$$b - L\mu_0 - Cv_0 \geq 0 \tag{2.23}$$

2.6.1 Teorema: Diseño óptimo del supervisor

Un supervisor diseñado con (2.21) y marca inicial (2.22), refuerza de manera óptima (2.20) [30] y [35].

Demostración: Primero obsérvese que las transiciones descritas por H están habilitadas sí:

$$Hq \leq b - L\mu + Cv$$

Es fácil mostrar que $D_c = -LD_p - C$, en todos los casos. Supóngase una secuencia de disparos q_a habilitados que lleva a una infracción de (2.20), la anterior consigna implica que

$$\left(LD_p + C\right)q_a > b - L\mu - Cv$$

El nuevo marcado del lugar de control es:

$$\mu_c(k+1) = (b - L\mu - Cv) + (-LD_p q_a - Cq_a) < 0$$

Lo cual no es posible, además implica $D_c^- q_a > b - L\mu - Cv$, lo cual es una contradicción. Por tanto se establece que la secuencia q_a no está habilitada si lleva a una infracción de (2.20).

La definición de reforzamiento óptimo se puede encontrar en [22], [23], [30], y [35] por motivos de espacio es omitida aquí, se puede comparar con la idea de “máximamente permisivo”.

2.6.2 Restricciones lineales vectoriales generales “mayor o igual”

Es posible aplicar una restricción de la forma:

$$L\mu + Hq + Cv \geq b \tag{2.24}$$

Mediante la utilización de las siguientes fórmulas se obtiene la matriz de incidencia de los lugares de control:

$$\begin{gathered} D_{lc}^+ = \max(0, LD_p + C) \\ D_{lc}^- = \max(0, -LD_p - C) \\ \\ D_c^+ = D_{lc}^+ + \max(0, H - D_{lc}^-) \\ \\ D_c^- = \max(D_{lc}^-, H) \end{gathered} \tag{2.25}$$

La marca inicial del lugar de control se obtiene mediante:

$$\mu_c = L\mu_0 + Cv_0 - b \tag{2.26}$$

Las anteriores ecuaciones son válidas si

$$L\mu_0 + Cv_0 - b \geq 0 \tag{2.27}$$

Un supervisor diseñado descrito por D_c^+ y D_c^- de la fórmula (2.25) y marca inicial (2.26) aplica (2.24) de manera óptima.

Demostración: Primero obsérvese que las transiciones descritas por H están habilitadas sí:

$$Hq \leq L\mu + Cv - b$$

La matriz de incidencia del controlador es $D_c = LD_p + C$. Una secuencia de disparos q_a habilitados que llevan a una infracción de (2.24) implicaría

$$\left(LD_p + C\right)q_a < b - L\mu - Cv$$

Nótese que $b - L\mu - Cv \leq 0$ debido a que $L\mu + Cv - b \geq 0$, esto a la vez implica $(LD_p + C)q_a < 0$. De la anterior ecuación se deriva.

$$abs\left(\left(LD_p + C\right)q_a\right) > L\mu + Cv - b$$

La nueva marca del lugar de control después del disparo de q_a, es.

$$\mu_c(k+1) = (L\mu + Cv - b) + \left(LD_p q_a + C q_a\right) < 0$$

Lo cual no es posible, e implica $D_c^- q_a > L\mu + Cv - b$, lo cual es una contradicción. Por tanto se establece que cualquiera que sea la secuencia de disparos, no está habilitada si lleva a una infracción de (2.24).

2.6.3 Admisibilidad de las restricciones generales

Determinar la admisibilidad de las LVGC es más complicado que en el caso del SBPI, sin embargo aún es posible utilizar la condición suficiente (2.4.1) como criterio, cambiando precisión por eficiencia computacional [30].

No existen métodos directos para transformar restricciones generales inadmisibles en otras, las cuales apliquen indirectamente las primeras, sin embargo es posible la transformación de una restricción de la forma (2.20) y (2.24) para que solo involucre el vector de marcas como (2.3) o (2.11). Al realizar la anterior transformación es posible hallar una solución con los métodos expuestos en la sección (2.4.3) para control basado en invariantes de lugar, después de esto, se aplica una transformación inversa para hallar las soluciones de la forma (2.20) y (2.24).

A continuación se exponen dos transformaciones, encontradas en [30] y [35], de la PN y sus respectivas restricciones, útiles en el proceso de encontrar soluciones admisibles. Dichas transformaciones permiten hacer uso de los métodos desarrollados para el control basado en invariantes de lugar.

2.6.3.1 La transformación C

Esta trasformación convierte restricciones de la forma $L\mu + Hq + Cv \leq b$ a restricciones de la forma $L_C\mu + H_C q \leq b$.

Entradas: La PN $N = (T, P, Pre, Post)$, las restricciones $L\mu + Hq + Cv \leq b$. De manera opcional el marcado inicial μ_0.

Salidas: La PN transformada $N_C = (T_C, P_C, Pre_C, Post_C)$ y el conjunto de restricciones $L_C\mu + H_C q \leq b$.

1. Inicialice $N = N_C$, sea $k = |P|$.
2. Para: $i\ a\ |T|$
   a.) Si la columna i-ésima de C no es nula
   - Igualar $k = k + 1$
   - Adicionar un nuevo lugar p_k a N_C, tal que $\bullet p_k = t_i$ y $p_k \bullet = \emptyset$.
   - Igualar $L_C = [L,\ C_i]$
   - Igualar $\mu_{C0} = [\mu_C^T,\ 0]^T$

Recuérdese la notación de pre-set y post-set: $\bullet p_k$ es el conjunto de transiciones con arcos de salida hacia p_k y $p_k \bullet$ es el conjunto de transiciones con arcos de entrada desde este lugar.

2.6.3.2 La transformación C – inversa

Entradas: La PN original y transformada $N_C = (T_C, P_C, Pre_C, Post_C)$, $N = (T, P, Pre, Post)$ las restricciones transformadas $L_C\mu + H_C q \leq b$.

Salidas: las restricciones $L\mu + Hq + Cv \leq b$.

1. Inicialice L a las primeras $|P|$ columnas de L_C, y C como una matriz nula.
2. Para: $i = |P| + 1$ a $|P_C|$.
   - Sea j el índice tal que $\bullet p_i = t_j$.
   - Igualar $C_j = L_{C,j}$.

2.6.3.3 La transformación H

Esta transformación convierte restricciones de la forma $L\mu + Hq \leq b$, a restricciones de la forma $L_H\mu_H \leq b$.

Entradas: La PN original $N = (T, P, Pre, Post)$, las restricciones $L\mu + Hq \leq b$. Opcionalmente la marca inicial μ_0.

Salidas: La PN transformada $N_H = (T_H, P_H, Pre_H, Post_H)$, las restricciones transformadas $L_H\mu_H \leq b$ y el marcado inicial μ_{H0} de N_H.

1. Inicializar $N_H = N$, $L_H = L$ y sea $j = |T|$ y $k = |P|$.

2. Para: $i = 1$ a $|T|$.
   a.) Si la columna i-ésima de H no es cero.
   - Igualar $j = j + 1$ y $k = k + 1$.
   - Adicionar un nuevo lugar p_k y una nueva transición t_j
   - Igualar D_{Hj}^{+}, $D_{Hj}^{-} = e_k$, $D_i^{+} = e_k$, donde e_k es un vector de ceros con un uno en la posición k.
   - Configurar los mismos atributos de controlabilidad y observabilidad para la transición t_j de la transición t_i.
   - Igualar $L_H = [L_H, H_i + LD_i^{-}]$ y $\mu_{H0} = [{\mu_{H0}}^T, 0]^T$

2.6.3.4 La transformación H – inversa

Entradas: La PN original $N = (T, P, Pre, Post)$, las restricciones $L_H \mu_H \leq b$ la PN transformada $N_H = (T_H, P_H, Pre_H, Post_H)$.

Salidas: Las restricciones $L\mu + Hq \leq b$.

1. Igualar L a L_H, con respecto a las primeras $|P|$ columnas, y H como una matriz nula.
2. Para: $k = |P| + 1$ a $|P_H|$.
   a.) Sea i el índice de la transición tal que $\bullet p_k = t_i$.
   b.) Igualar $H_i = L_{Hk} - L_H D_{Hi}^{-}$.

Las anteriores transformaciones están definidas para restricciones de la forma (2.20), pero pueden ser usadas, sin cambios, para restricciones de la forma (2.24) y obtener restricciones transformadas del tipo (2.3) o (2.11).
A continuación se presenta un teorema útil para la transformación de restricciones, este está definido para restricciones (2.20), sin embargo puede ser extendido fácilmente para (2.24), se omite la demostración de los dos casos por extensión.

2.6.3.5 Exactitud del diseño del supervisor

Sea $L_{HC}\mu_{HC} \leq b$ las restricciones obtenidas después de aplicar la transformación C y posteriormente la transformación H a la PN N y las restricciones $L\mu + Hq + Cv \leq b$, obteniéndose la PN N_{HC}. Sean $L_{HCa}\mu_{HCa} \leq b_a$ restricciones admisibles sobre N_{HC} tal que $\forall \mu_{HC}: L_{HCa}\mu_{HCa} \leq b_a \Longrightarrow L_{HC}\mu_{HC} \leq b$, es decir se cumple el lema (2.4.3.1), se obtiene $L_a\mu + H_a q + C_a v \leq b$ al aplicar las transformación H – inversa y posterior la transformación C – inversa a $L_{HCa}\mu_{HCa} \leq b$, entonces el conjunto de restricciones $L_a\mu + H_a q + C_a v \leq b$ es admisible, y si son reforzadas por un supervisor también se refuerza $L\mu + Hq + Cv \leq b$ [30] y [35].

Ejemplo. Transformada C y C – Inversa.

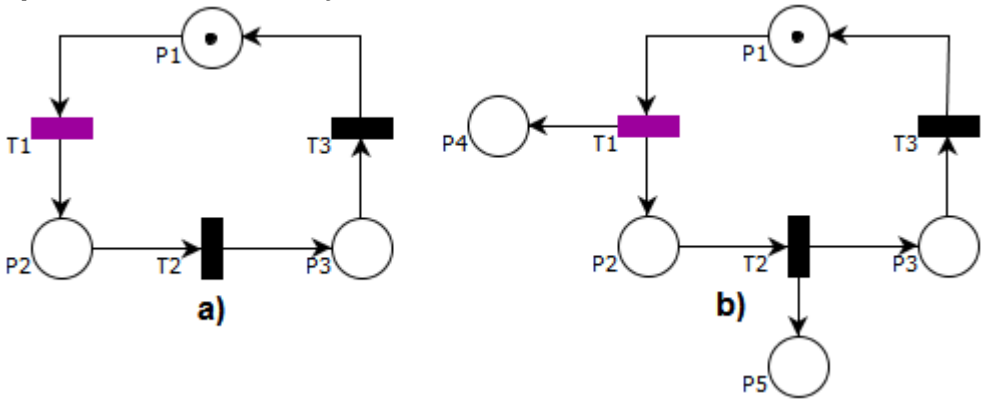
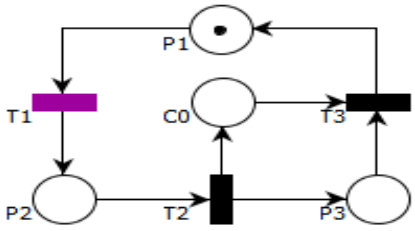


Figura 2.6 Ilustración transformada C

Suponga que se desea imponer la restricción $v_1 - v_2 \leq 1$ sobre la PN de la Figura 2.6 a), donde T1 es no controlable. Esta es equivalente a $\mu_4 - \mu_5 \leq 1$ en la PN de la Figura 2.6 b), en este caso se habla de la transformación C, tanto de la PN como de la restricción. Observe que los lugares P4 y P5 en la Figura 2.6 b) actúan como contadores de los disparos de las transiciones T1 y T2 respectivamente, además, note que la trasformación C hace que la restricción solo involucre el vector de marcas. La restricción $v_1 - v_2 \leq 1$ es inadmisible por lo que $\mu_4 - \mu_5 \leq 1$ también lo será, esto significa que debe transformarse, con cualquiera de los métodos vistos en la sección 2.4.3, para poder ser aplicada en la PN. Al transformar $\mu_4 - \mu_5 \leq 1$ resulta $\mu_1 + \mu_4 - \mu_5 \leq 1$ la cual es equivalente a $\mu_1 + v_1 - v_2 \leq 1$ en la PN de la Figura 2.6 a). En este caso se habla de la transformación C – Inversa. De esta manera el DEC que hace cumplir $v_1 - v_2 \leq 1$ es:



Ejemplo. Transformada H y H – Inversa.

Suponga que se impone la restricción $\mu_2 + q_3 \leq 1$ a la PN de la Figura 2.7 a), donde T3 es no controlable. Dicha restricción es equivalente a $\mu_2 + \mu'_3 \leq 1$ impuesta sobre la PN de la Figura 2.7 b); en este caso se habla de la transformación H, tanto de la PN como de la restricción.

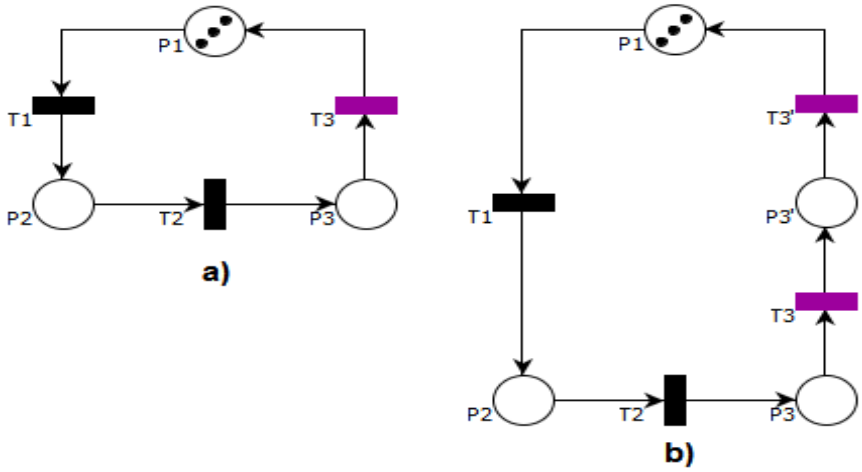
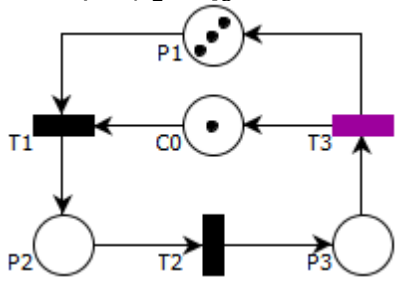


Figura 2.7 Ilustración transformada H

Note que la trasformación H hace que la restricción solo involucre el vector de marcas. La restricción $\mu_2 + q_3 \leq 1$ es inadmisible por lo que $\mu_2 + \mu'_3 \leq 1$ también lo será, esto significa que debe transformarse, de acuerdo a los métodos vistos en la sección 2.4.3, para poder ser aplicada en la PN. Al transformar $\mu_2 + \mu'_3 \leq 1$ resulta $\mu_2 + \mu_3 + \mu'_3 \leq 1$ la cual es equivalente a $\mu_2 + \mu_3 \leq 1$ en la PN de la Figura 2.7 a) (Nótese que la transformación H inversa elimina el vector de disparo, si este está dirigido a una transición no controlable). En este caso se habla de la transformación H – Inversa. De esta manera el DEC que hace cumplir $\mu_2 + q_3 \leq 1$ es:



2.7 Supervisión de un Sistema Compresor-Acumulador de Aire

La planta de la Figura 2.8 es un caso típico del suministro de aire comprimido a un proceso en la industria [36]. El tanque acumulador de aire cuenta con dos sensores binarios para medir una alta y baja presión, PS1 y PS2 respectivamente. Mediante dos compresores, A y B; se surte al tanque de aire, dichos compresores cuentan con un sensor interno el cual envía una señal en caso de disturbio. Los requerimientos de control son los siguientes:

1. Si la presión en el tanque es menor a 10 bar (señal PS1 off) y superior a 9.7 bar (señal PS2 off) solo debe funcionar un compresor (rango de presión deseado).
2. Si la presión aumenta por encima de 10 bar (señal PS1 on), ningún compresor debe funcionar.
3. Si la presión desciende por debajo de los 9.7 bar (señal PS2 on), los dos compresores deben funcionar
4. Si uno de los compresores presenta disturbio, cuando la presión este dentro del rango deseado, el otro debe sustituirle.
5. Cuando la presión se encuentre dentro del rango deseado, los compresores deben funcionar alternadamente en periodos de ½ hora.

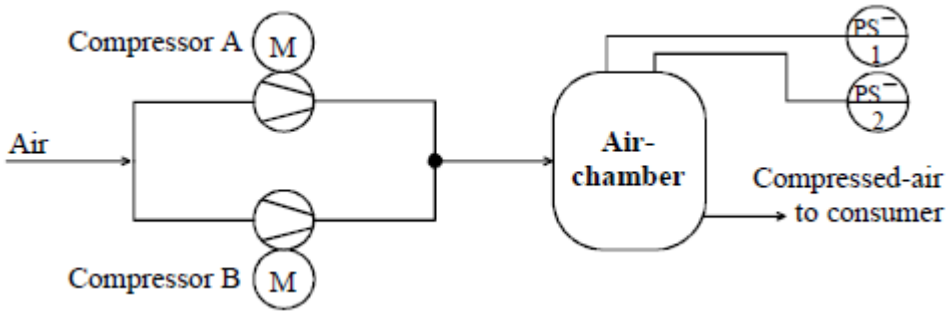


Figura 2.8 Compresor y acumulador de aire [36]

Se supondrá que la planta inicia el proceso con el tanque dentro de la presión deseada, el compresor A On y B Off.

Restricciones a imponer:

1. $q_1 + \mu_5 - \mu_6 + \mu_8 \leq 2$: Esta restricción se impone para impedir que el compresor A funcione si la presión del tanque está bien y si está operando el compresor B; además permite al compresor A entrar en operación, en caso que la presión descienda a un valor no permitido.
2. $q_1 + \mu_4 + q_9 \leq 1$: Esta condición se impone para impedir que ninguno de los compresores se active cuando la presión supere el límite máximo.
3. $\mu_2 + \mu_5 + q_9 \leq 2$: Mediante esta condición se impide que el compresor B se active si la presión está bien y si el compresor A está funcionando.
4. $v_9 - v_1 \leq 1$
5. $2v_1 - 2v_9 \leq 1$: Mediante las restricciones 4 y 5 se busca que la operación de los compresores sea de manera alternada, es decir, actúan como un “árbitro”.
6. $\mu_2 + \mu_8 + 2q_{13} + 2q_{14} + v_{13} + v_{14} - v_6 - 2v_8 \leq 2$: Con esta restricción se busca que los compresores se apaguen de inmediato cuando la presión supere el límite máximo permitido en el tanque, y que solo

funcione uno cuando el tanque recupere la presión normal desde el límite inferior.

P1	Compresor A Off	T1	Encender compresor A
P2	Compresor A On	T2	Apagar compresor A con retardo de 1 hora
P3	Compresor A en disturbio	UC:T3	Disturbio en compresor A
P4	Presión mayor a 10 bar	T4	Apagar compresor A
P5	Presión entre 10 – 9.7 bar	T5	Descenso de presión desde más de 10 bar
P6	Presión menor a 9.7bar	T6	Aumento de presión desde menos de 10 bar
P7	Compresor B Off	T7	Descenso de presión desde más de 9.7 bar
P8	Compresor B On	T8	Aumento de presión desde menos de 9.7 bar
P9	Compresor B en disturbio	T9	Encender compresor B
C0	Lugar de control para restricción 1	T10	Apagar compresor B con retardo de una hora
C1	Lugar de control para restricción 2	UC:T11	Disturbio en compresor B
C2	Lugar de control para restricción 3	T12	Apagar compresor B
C3	Lugar de control para restricción 4	T13	Apagar compresor A
C4	Lugar de control para restricción 5	T14	Apagar compresor B
C5	Lugar de control para restricción 6		

Tabla 2.3 Descripción de lugares y transiciones

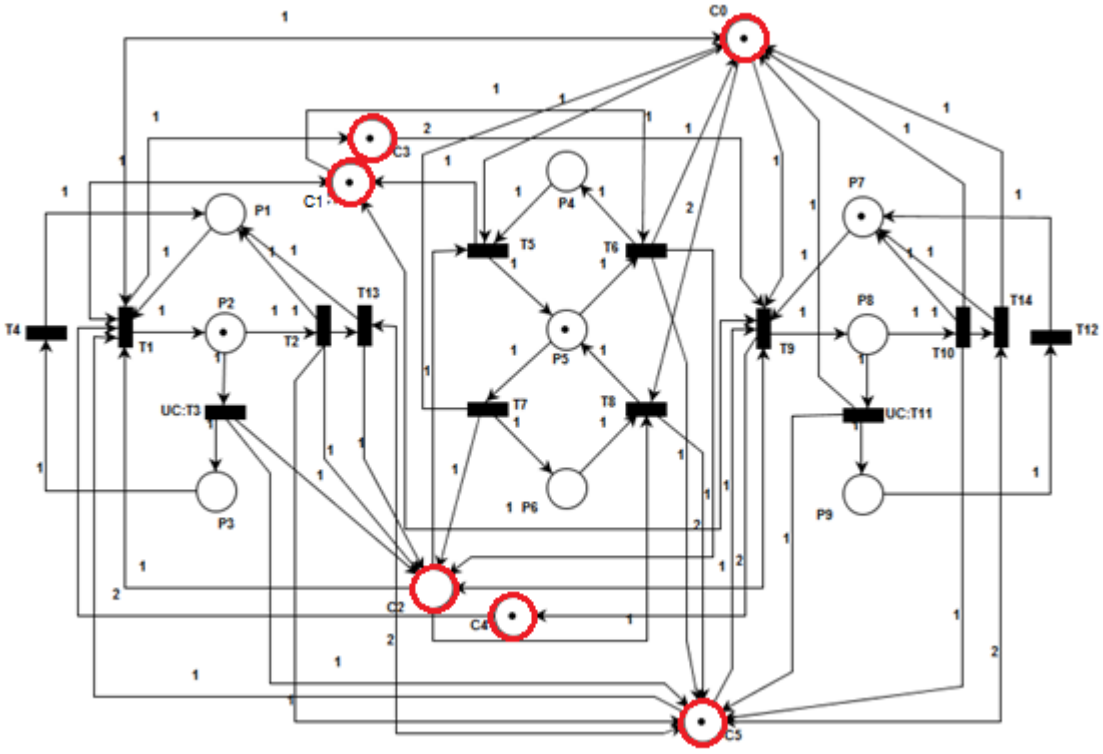


Figura 2.9 Supervisor del compresor-acumulador de aire

Observe que las transiciones T5, T6, T7, T8, T3 y T11 son no controlables, y solo T3 y T11 no tienen arcos dirigidos desde los lugares de control. El hecho que las transiciones T5, T6, T7 y T8 tengan dirigidos arcos desde los lugares de control no hace inadmisible al supervisor, ya que los lugares de control no inhiben en ningún momento su disparo, lo que hace admisible el supervisor [37] y [10]. En síntesis, todas las restricciones son admisibles y su actuación en conjunto, sobre el modelo del sistema, permiten a este comportarse conforme a los requerimientos de control. La Figura 2.9 muestra el sistema en lazo cerrado.

3 GENERACIÓN DE CÓDIGO LADDER A PARTIR DE REDES DE PETRI

Una vez diseñado el supervisor el siguiente paso es implementarlo, en este caso se hará mediante un Controlador Lógico Programable (PLC). El código *ladder* o escalera es uno de los cuatro lenguajes estándar especificados en la norma IEC 61131-3 para la programación de PLC's, llegando a ser el más popular gracias a la facilidad de uso y compresión de sus estructuras. En este trabajo se aplicará un método para realizar la implementación del supervisor, basado en PN's, en un PLC mediante la generación automática de código *ladder*.

3.1 Enfoques

La generación de código *ladder* a partir de PN ha sido trabajada por muchos investigadores, lo que ha dado como resultado una diversa cantidad de enfoques metodológicos y heurísticos. Se revisarán algunos de estos enfoques en esta sección.

Existe una herramienta de generación de código *ladder*, a partir de una PN, llamada Petri-LLD [38], este software produce código para controladores Allen Bradley (5000), Omron y Siemens, a partir de PN's seguras, sin embargo, la herramienta presenta limitaciones como, la restricción de la PN a ser de tipo segura, además modifica las reglas de comportamiento de la PN, lo que imposibilita un análisis formal del supervisor con las técnicas expuestas, por último, se genera solo la estructura del *ladder*, lo que hace necesario declarar manualmente todas las variables necesarias y asignar las entradas y salidas, en el código *ladder*, tarea que puede ser de cuidado según el tamaño de la PN.

Un enfoque parecido al anterior, es el presentado por Georg Frey [39] y [40] con las redes SIPN (Signal interpreted Petri Nets) usadas para representar procesos secuenciales y concurrentes, por lo que se definen como PN's seguras, en este enfoque se considera la PN más como un lenguaje de programación de PLC que como una herramienta para modelar sistemas y construir supervisores. Cada SIPN se implementa en el PLC como una función, por lo que cada PN debe contener una condición inicial y una condición de salida hacia la PN principal.

Mientras los enfoques anteriores utilizan las PN's como un lenguaje de programación, una técnica trabajada por Uzam, Jones y Ajlouni en [3] y [41] denominada *Token Passing Logic* está enfocada en la implementación en el

PLC de una PN cualquiera, eliminando la limitación a PN's seguras. La metodología considera redes generales, temporizadas (lugares o transiciones) y coloreadas. Además, la existencia de elementos no ordinarios como arcos inhibidores y habilitadores.

La PN obtenida en el diseño del supervisor mediante la técnica adoptada en este proyecto no necesariamente es segura ni ordinaria, puede ser temporizada, por medio de las transiciones, y no pueden contener arcos inhibidores y/o habilitadores[2], a continuación se describe la parte de esta metodología que permite modelar el tipo de PN's con las cuales se trabajará.

3.1.1 Modelado de lugares

Uzam [3], propone utilizar una bandera (bit) para modelar lugares con capacidad igual a uno y un contador (variable entera) para modelar lugares con capacidad mayor que uno. Cuando la bandera contenga el valor 1 indica la presencia de marca y cero indica vacío, el valor de la variable entera indica el número de marcas en el lugar con capacidad mayor. El marcado inicial de estos lugares se establece con la siguiente estructura *ladder* utilizando el bit de F0 como una bandera de inicialización. En la red de petri, se puede determinar la capacidad de un lugar mediante el cálculo de los invariantes positivos.

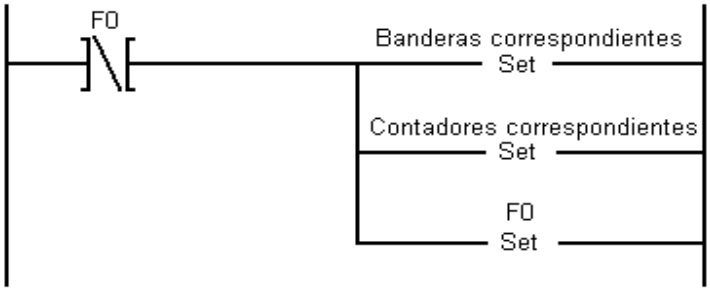


Figura 3.1 Establecimiento del marcado inicial

Con la manipulación de las variables de los lugares se emula los estados que puede alcanzar la PN, las estructuras que modifican estas variables se describen a continuación.

[2] La razón de esto, es que, aunque la teoría de control por medio de invariantes y restricciones generales aun es válida para aquellas redes, no es posible realizar un análisis de bloqueo fiable. De todas formas, en todos los casos de estudio que se han trabajado nunca ha habido necesidad de este tipo de arcos, además siempre hay formas de realizar las mismas funciones con arcos normales.

3.1.2 Modelado de arcos y transiciones

Las transiciones son las encargadas de actualizar los estados de la PN, estas se disparan si todos los lugares con arcos de entrada a la transición tienen al menos igual número de marcas que el peso del arco. Posterior al disparo, se suma a cada lugar de salida un número de marcas especificado por los arcos de salida. Teniendo en cuenta que los arcos considerados en este trabajo tienen peso mayor o igual a uno se considera una variable entera o bit para cada uno, la siguiente estructura basada en [3] es utilizada para modelar los arcos de una transición.

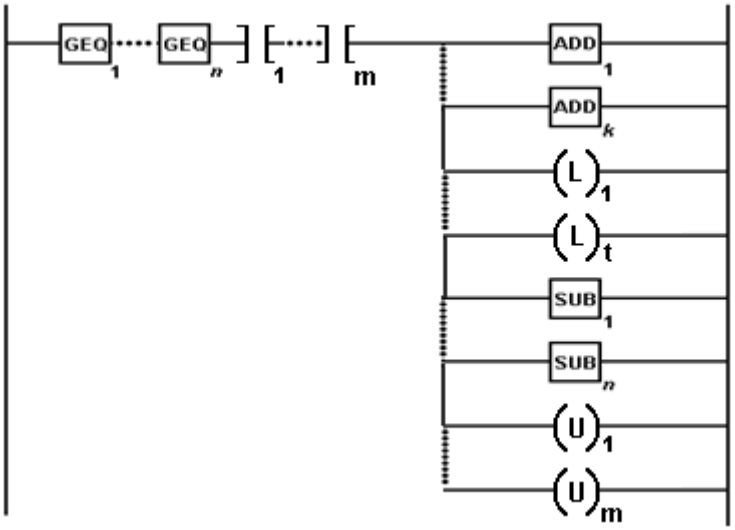


Figura 3.2 LLD para arcos con peso mayor o igual a 1

Los comparadores (GEQ: *Mayor o igual*) son utilizado para evaluar cada variable con respecto al peso de cada uno de los n arcos de entrada a la transición para los lugares con capacidad mayor que uno, en seguida se usa un examinador de cerrado (XIC) para los m lugares con capacidad uno. En caso que todos los comparadores y examinadores resulten verdaderos, se procede a adicionar (ADD: *adicionar*) el valor de cada uno de los k arcos hacia a la respectiva variable de cada lugar de salida con capacidad mayor y con un enganche (*OTL*:,(L) enganche) poner la marca para los t lugares con capacidad uno, por último solo resta substraer (SUB: *substraer*) el valor de cada n arco a la variable del lugar de entrada correspondiente y quitar la marca para cada uno de los m lugares con capacidad uno mediante un des enganche (OTU: (U), desenganche).

3.1.3 Modelado de eventos

La manera más natural de asociar eventos a un DEDS basado en PN's es por medio de las transiciones, en la cual la ocurrencia de un evento equivale al disparo de una transición. Uzam en sus trabajos [3] y [41], propone el uso

de las APN (Automated Petri Nets) las cuales siguen la convención anterior, el modelado del evento en la transición es llevado a cabo con la siguiente estructura.

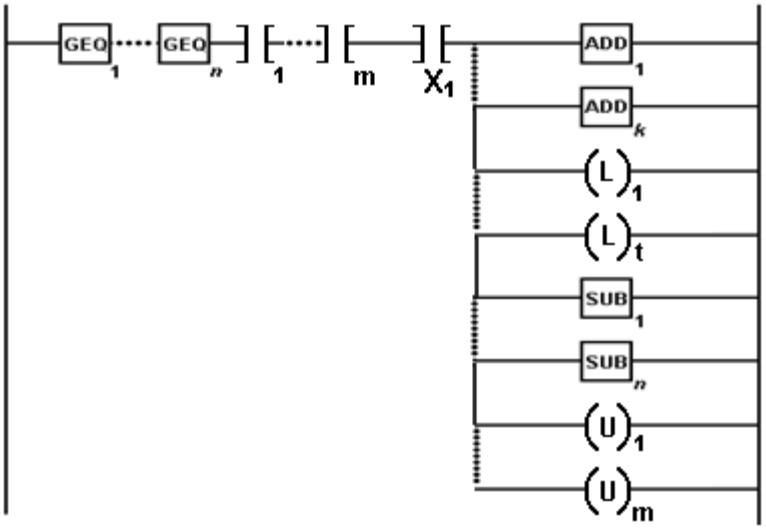


Figura 3.3 LLD para un evento

Donde X_1 es una entrada de sensor al PLC la cual indica la ocurrencia del evento (o entrada). Como lo muestra la Figura 4.3, la transición debe estar habilitada y el evento (o entrada) debe estar activo para que pueda disparase.

Es necesario advertir al lector que en este trabajo *evento* será referido a un suceso cualquiera, que puede ser medido y detectado por el PLC mediante una de sus entradas, diferente a *transición*, que es un elemento de la red de petri. Aunque esta definición pueda resultar equivalente para muchos casos prácticos en los cuales las transiciones son disparadas cuando ocurre la activación de una señal, no es estrictamente igual con la definición de la teoría tradicional de DEDS, donde estos dos conceptos son prácticamente equivalentes.

3.1.4 Modelado de acciones

De manera análoga a las entradas, la forma más práctica de asociar una acción a un DEDS basado en PN es por medio de los lugares, pero no todos los lugares tendrán necesariamente una acción asociada. La siguiente estructura es usada para modelar el uso de las acciones en la PN.

Los comparadores y examinadores evalúan la variable correspondiente a cada lugar con la acción asociada, este devuelve verdadero si encuentra algún comparador mayor o igual a uno (lugar con capacidad mayor) o algún

examinador cerrado, de esta manera, la acción se activa si al menos un lugar asociado contiene al menos una marca.

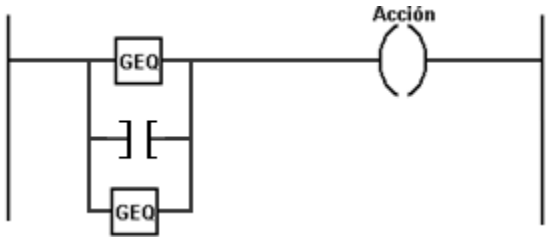


Figura 3.4 LLD para una acción

Nótese que en la figura se toma como ejemplo una acción asociada a tres lugares, dos con capacidad mayor a uno y uno con capacidad igual a uno. No existe límite en la metodología para los lugares asignados a la acción, aunque el PLC si lo puede tener, por ejemplo, la familia SLC solo permitiría un máximo de 76 ramas paralelas lo cual significa un máximo de 76 lugares por acción.

3.1.5 Modelado de transiciones temporizadas

Se modela el tiempo por medio de un retraso en el disparo de las transiciones, una vez habilitada la transición y lista para disparar, empezara a contar un temporizador el cual una vez terminado activara el disparo, siempre y cuando la transición continúe habilitada y el evento siga presente, en caso de tener asociado alguno. La estructura *ladder* que realiza esto se ve en la Figura 3.5.

Cada vez que la transición se dispara se reinicia el temporizador, de esta manera se vuelve a iniciar el ciclo. Si la transición además contiene un evento, es necesario agregar el bit del evento después de los comparadores y examinadores de las dos primeras líneas.

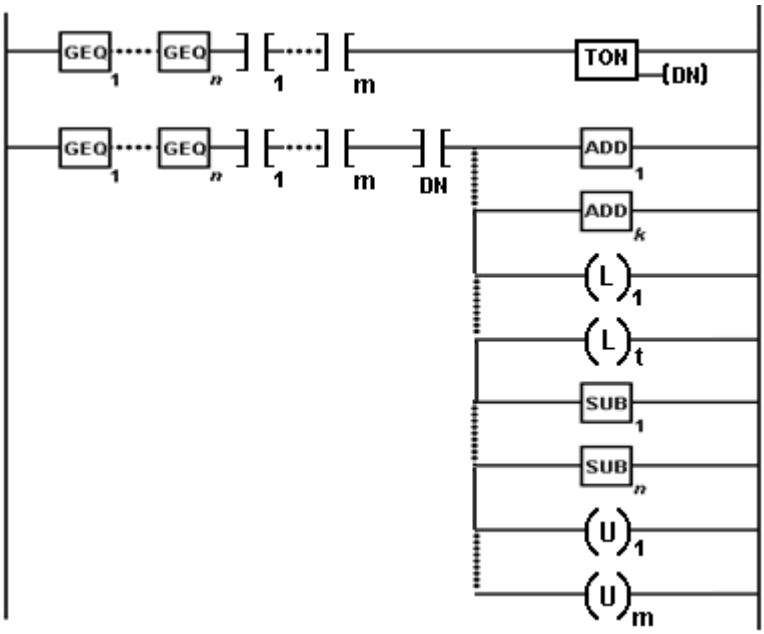


Figura 3.5 LLD para transiciones temporizadas

3.2 Modificaciones y Extensiones

La metodología utilizada en este proyecto para la generación de código *ladder* es la *Token Passing Logic*, explicada anteriormente. Es necesario plantear algunas variantes y extensiones de la metodología para resolver determinadas falencias, las cuales son explicadas a continuación.

3.2.1 Marcado inicial

La forma original propuesta por la metodología para establecer el marcado inicial es por medio de una estructura como la expuesta en 3.1.1, sin embargo, esto no es necesario, ya que es posible establecer el valor inicial de cualquier variable escribiendo directamente en el archivo que contiene el código *ladder*.

3.2.2 Interruptores para la generación de eventos

Para trabajar sobre una planta física, es necesario determinar la ocurrencia de eventos en la red de Petri basándose en la evolución de señales lógicas en el sistema real, estas señales pueden ser salidas de sensores u otros componentes, también podrían relacionarse con instrucciones del operario.

Como se ilustra en la sección 3.1.3, la generación de eventos en la red de Petri está asociada directamente a la activación de una señal lógica, bajo este enfoque, se considera que el evento ocurre *continuamente* mientras x_1 continúe activo. Para trabajar sobre algunos casos prácticos, es necesario adicionar variantes sobre la metodología original, en la cual se generen eventos basándose en otros estados de las señales.

3.2.2.1 Eventos generados por un cambio de estado

Como se observa en la estructura propuesta en 3.1.3, es posible disparar la transición siempre y cuando el evento generado por la entrada x_1 esté activado, si los lugares contienen suficientes marcas para habilitar la transición esta se disparará indefinidas veces hasta que x_1 pase a un estado de apagado o hasta que se agoten las marcas.

En muchos casos prácticos se requiere que las transiciones se disparen solo una vez cuando se activa una entrada, es decir, el evento solo ha ocurrido una vez, de esta manera para conseguir un nuevo disparo la entrada tendría que pasar a estado apagado y volver a activarse. En seguida se muestra la lógica *ladder* para implementar esta variante.

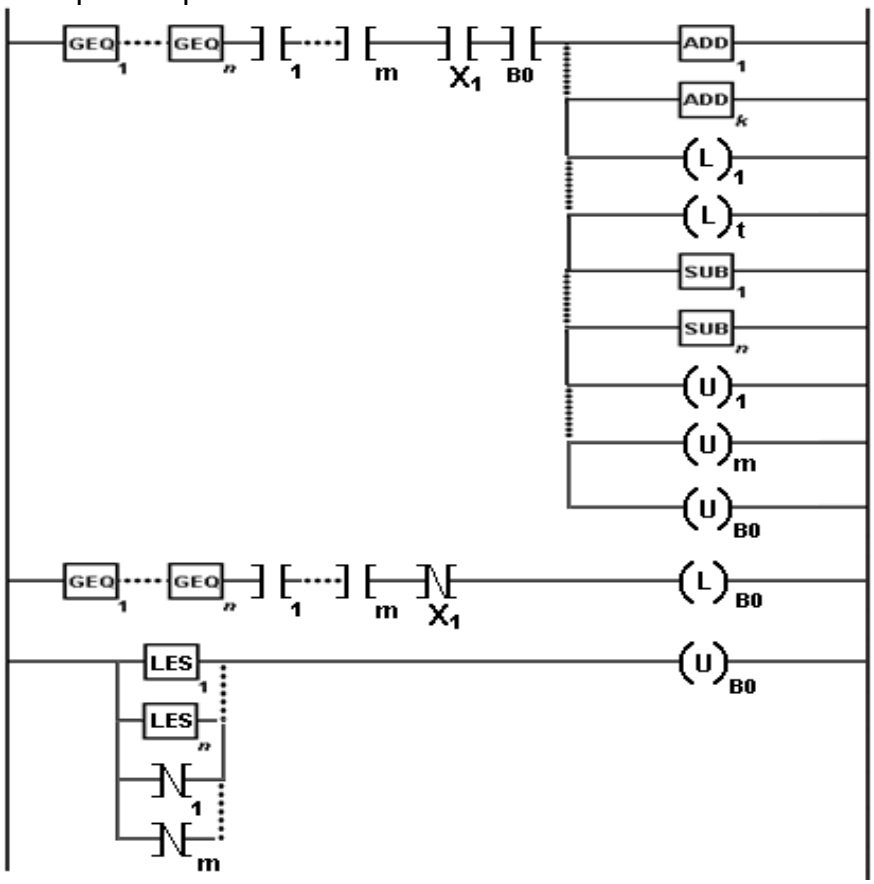


Figura 3.6 LLD propuesto para una transición con un evento asociado

La transición es disparada si está habilitada, el evento activado y el bit B0 es verdadero, nótese que al dispararse la transición se desactiva el bit B0 ocurriendo un solo disparo, además para activar este bit es necesario que el evento x_1 esté antes desactivado, de este modo se garantiza que el disparo ocurra cuando el evento cambia de estado. Por último, es preciso desactivar B0 cuando la transición no está habilitada.

Para utilizar la estructura anterior es necesario el uso de un bit de memoria del PLC por cada transición con este tipo de evento. Si además la transición es temporizada, es preciso agregar la línea para implementar el temporizador y reiniciarlo después del disparo, esta línea también debe ser condicionada por el bit de memoria (B0).

3.2.2.2 Eventos generados por entradas apagadas

En muchos casos prácticos, un evento consistirá en la desactivación de una entrada, esto es muy común en sensores de tipo 1 y 0. Para modelar este hecho, es necesario modificar los examinadores de cerrado por examinadores de abierto en la estructura propuesta en 3.1.3.

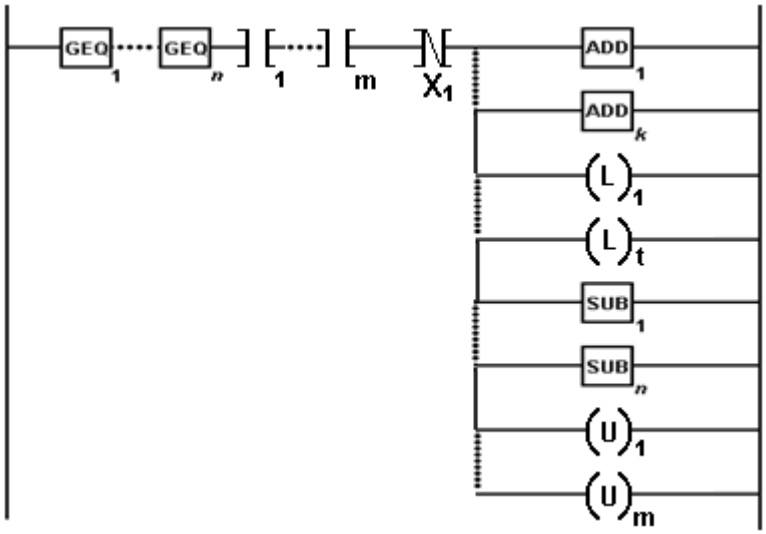


Figura 3.7 LLD para eventos negados

Es preciso realizar el mismo cambio en todos los examinadores que se puedan agregar, en caso de que la transición sea temporizada y/o el evento este denotado como de cambio de estado.

3.3 Generación de Código *Ladder* Para RSLogix500

Para generar código para RSLogix 500, la manera más práctica es editar un archivo de exportación de Rockwell (SLC). El fichero SLC es un archivo de texto, de esta manera es posible escribirlo usando funciones para cadenas y escritura de archivos.

La estructura general del archivo SLC es la siguiente:

1. Configuración física del PLC
2. Información de la estructura *ladder*.
3. Configuración de parámetros del controlador
4. Declaración de las variables usadas en el *ladder*.

En esta sección se enfocara la estructura del fichero SLC, en especial las partes dos y cuatro, las cuales permiten escribir un código para RSLogix500, las otras partes, no se deberían modificar, solo detectar y copiar al código sin ningún cambio, la herramienta “PLC Detector” ha sido diseñada para realizar esto de manera automática, en el anexos A se encuentra más información respecto a esta aplicación.

3.3.1 Definición de la estructura *ladder*

Una vez definida la configuración física del PLC, una secuencia de estructura *ladder* empieza con una nueva línea con la palabra clave *ladder* y el número identificador de la secuencia. Por ejemplo:

```
%Arriba configuración física%

LADDER 2

%Abajo empiezan las líneas del ladder%
```

Los comentarios en este archivo vienen denotados por los caracteres ‘%’, cualquier texto entre estos caracteres se considera comentario.

Después de la declaración, cada peldaño de *ladder* debe estar en una línea de texto, y sus instrucciones entre las palabras claves de inicio y fin: SOR (start of rung) y EOR (end of rung), respectivamente. En el siguiente cuadro de dialogo se muestran tres peldaños de *ladder* vacios, como ejemplo.

```
LADDER 2

SOR %Contenido del peldaño% EOR
SOR %Contenido del peldaño% EOR
SOR %Contenido del peldaño% EOR
```

Para cada instrucción en código *ladder* existe una palabra clave, los parámetros de la instrucción se colocan justo después de la declaración de la instrucción. En la siguiente tabla, se muestran las palabras claves de las instrucciones más comunes.

DESCRIPCION	SIMBOLO	PALABRA CLAVE ROCKWELL	PARAMETROS	OPERACION
Examinador de cerrado		XIC	<Bit>	Verdadero si el bit es 1
Examinador de abierto		XIO	<Bit>	Verdadero si el bit es 0
Enganche		OTL	<Bit>	El bit es colocado a 1 (retentivo)
Des-enganche		OTU	<Bit>	El bit es colocado a 0 (retentivo)
Conteo ascendente	CTU	CTU	<Contador> <preset> <Acum>	Acum incrementa, preset es el valor inicial
Conteo descendente	CTD	CTD	<Contador> <preset> <Acum>	Acum decrementa, preset es el valor inicial
Activar		OTE	<Bit>	Activa el bit mientras hay paso
Mayor igual	GEQ	GEQ	<entero1> <entero2>	verdadero si entero1≥entero2
Resta	SUB	SUB	<entero1> <entero2> <entero3>	entero3= entero1-entero2
Suma	ADD	ADD	<entero1> <entero2> <entero3>	entero3= entero1+entero2
Temporizador retraso	TON	TON	<temporizador> <tiempo base> <No. ciclos> <preset>	Espera No. ciclos, cada uno tarda “tiempo base”. preset es el valor incial.
Reset		RES	<elemento>	Devuelve el

				elemento al valor inicial

Tabla 3.1 Palabras claves comunes en cada instrucción

Para usar las instrucciones, es necesario declarar las variables que se usaran. En RSLogix, existen 6 tipos de variables que se utilizarán para la generación automática de código *ladder*: Controladores, temporizadores, contadores, enteros, flotantes y binarios. Cada tipo de variable tiene una letra y número de identificador, de la siguiente manera.

B3: binarios
T4: Temporizadores
C5: Contadores
R6: Controladores
N7: Enteros
F8: Flotantes

Estas variables son declaradas en un solo vector para cada tipo, excepto los binarios, los cuales se declaran en conjuntos de 16 elementos, de esta manera se referencia a cualquier elemento en el archivo SLC, por ejemplo:

N7:1 %entero en la posición dos%
F8:0 %flotante en la primera posición%.
T4:8 %Noveno temporizador%
C5:0 %contador en la primera posición%
R6:1 %Controlador en la segunda posición%.

Nótese que se sigue un índice empezando en cero. Los binarios se referencian de manera similar, excepto que se referencia el conjunto y posteriormente se agrega un slash para indicar el elemento:
B3:0/1 %Binario del primer elemento bit segundo%
B3:1/0 %Binario del segundo elemento bit primero%

Algunos elementos, como los temporizadores o contadores contienen bits indicativos usados para señalar a la rutina la finalización de una espera o terminación de una tarea, a estos bits se accede con un slash después del elemento y especificando el bit posteriormente, por ejemplo.

T4:0/DN %bit de terminación del primer temporizador%.
R6:7/EN %bit de habilitación del octavo controlador%.

Para referenciar entradas y salidas, se utiliza de clave la letra I (Input) como clave para las entradas y la letra O (Output) como clave para las salidas, después se hace referencia al slot en el cual se encuentran y posteriormente al elemento, por ejemplo.

I:2/8 %Novena entrada del Slot dos%
O:1/0 %Primera salida del Slot uno%.

En caso de que la entrada o salida sea propia del controlador, y no sea de un slot, se coloca el número cero en la referencia, por ejemplo:

I:0/15 %Entrada dieciséis, nativa del dispositivo.%

Para definir una línea de código en *ladder*, se escribe las palabras claves de las instrucciones seguido de las variables que se utilizarán de parámetros, por ejemplo considérese el siguiente programa en *ladder*.

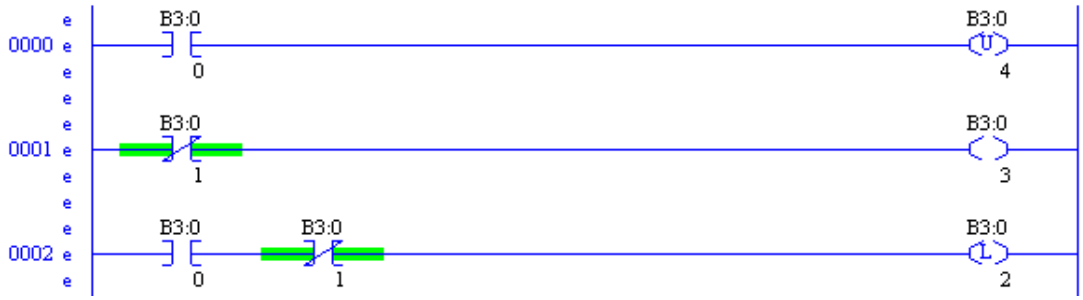


Figura 3.8 Programa en *ladder*

El programa contiene tres peldaños usa 7 instrucciones y 5 variables binarias, el texto que define este *ladder* sería el siguiente:

```
LADDER 2

SOR XIC B3:0/0 OTU B3:0/4 EOR
SOR XIO B3:0/1 OTE B3:0/3 EOR
SOR XIC B3:0/0 XIO B3:0/1 OTL B3:0/2 EOR
```

3.3.2 Ramas paralelas y anidadas

Las ramas del *ladder* se agregan mediante las instrucciones BST (Branch start) y BND (Branch end), los distintos caminos se separan con una instrucción NXB. Por ejemplo, el siguiente código muestra un peldaño conteniendo tres ramas paralelas.

```
SOR BST %rama uno% NXB %rama dos% NXB %rama tres% BND
EOR
```

La familia de controladores SLC-500 soporta un máximo de 76 ramas paralelas y 4 ramas anidadas. Mediante este texto se puede ya crear cualquier programa *ladder* mediante la escritura directa en un archivo SLC, e implementar la metodología explicada en la primera sección del capítulo. De manera ilustrativa, considérese el siguiente programa *ladder* con distintas instrucciones y ramas paralelas y anidadas.

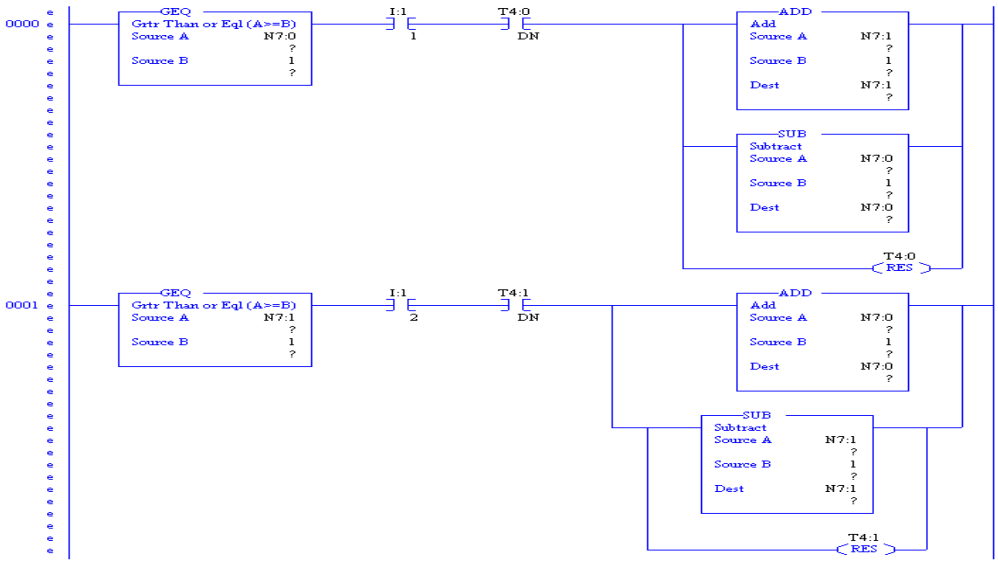


Figura 3.9 Ramas anidadas y paralelas en *ladder*

Escrito en texto, este código *ladder* quedaría de la siguiente manera.

```
LADDER 2
SOR GEQ N7:0 1 XIC I:1.0/1 XIC T4:0/DN BST ADD N7:1 1
N7:1 NXB SUB N7:0 1 N7:0 NXB RES T4:0 BND EOR
SOR GEQ N7:1 1 XIC I:1.0/2 XIC T4:1/DN BST ADD N7:0 1
N7:0 NXB BST SUB N7:1 1 N7:1 NXB RES T4:1 BND BND EOR
```

3.3.3 Declaración de las variables

Las variables utilizadas en el *ladder* se declaran con la palabra clave DATA seguido por el identificador del tipo de variable. La forma de definición de las variables se puede dividir en dos formas, los controladores temporizadores y contadores se declaran en líneas separadas, y los flotantes, enteros y binarios en una sola línea, separando los distintos elementos con espacios.

Los temporizadores, contadores y controladores se declaran con un número hexadecimal que indica el valor de las variables del elemento y dos enteros indicando el preset y acumulador, en el caso de los temporizadores y

contadores, y la posición y longitud en el caso de los controladores. Por ejemplo.

DATA T4:0
0x0000 99 0 %Temporizador con base 0.01s y 99 ciclos (retraso= 0.99s)%
0x0200 42 0 %Temporizador con base 1s y 99 ciclos (retraso= 42segs)%

Nótese que el tiempo base solo puede contener 3 valores, 1, 0.01, y 0.001, por tanto no se guarda como un valor, sino como un indicativo. Todas estas banderas especificadas por el valor hexadecimal. Los binarios, flotantes y enteros se declaran de forma decimal. En el caso de los binarios, el valor decimal especifica el estado de los dieciséis bits, convirtiendo el valor a binario, por ejemplo.

DATA B3:0
0 0 0 4 512 %5 elementos binarios, en total 80 bits, solo dos bit son uno inicialmente B3:3/4 y B3:4/10%

DATA N7:0
1 5 6 87 9 0 0 0 %8 enteros con sus valores iniciales%

DATA F8:0
23.2 4.0 5.236 % tres flotantes con sus valores iniciales%

3.3.4 Parámetros del controlador y configuración física

Los parámetros de controlador y la configuración física del PLC también es escrita en el archivo SLC. Cada dispositivo tiene su propia configuración, esta es guardada en la parte S:0, la cual indica variables de status. Cada controlador puede disponer de valores de distintas variables, por lo que no es práctico discutir cada una, en general, solo será necesario pasar estos valores intactos como los guarda RSLogix.

La parte inicial del archivo SLC contiene la configuración física de PLC, en general, la información física se define por 3 palabras clave:

INSTRUCCION	PARAMETROS	DESCRIPCION
START	<identificador>	Indica el dispositivo el cual se programará
SLOT	<posición> <identificador>	Indica la presencia de un slot y sus posición en el rack
RACK	<id> <…..>	Indica la configuración del rack, esta opción puede contener distintos

		parámetros para cada controlador.

Tabla 3.2 Instrucciones para identificar la información física del PLC

Por ejemplo, la siguiente es la declaración de un dispositivo Micrologix 1500 LRP Serie C, con 2 módulos de 4 canales de entrada analógicos y uno de 8 salidas de 24 voltios continuos.

START Bul.1764
SLOT 0 Bul.1764
SLOT 1 1769-IF4
SLOT 2 1769- IF4
SLOT 3 1769- OB8

Los módulos de entrada se encuentran en los dos primeros slots, el de salida está en el tercer slot. Este dispositivo no puede tener configuración de rack. Las etiquetas que están después de la instrucción son identificadores usados por Rockwell para identificar un módulo o dispositivo, véase el anexo B para ver todas las etiquetas y el módulo o dispositivo que identifican. El siguiente dispositivo es un PLC SLC-500 de 4k de memoria, similar al encontrado en la sala de automática en la Universidad del Cauca.

START 1747-L514
RACK 1 1746-A4 % Rack de 4 slots%
SLOT 1 1746-IB32
SLOT 2 1746-OB32

Este dispositivo está instalado con un rack de 4 slots, de los cuales los primeros dos están ocupados por un módulo de 32 entradas y otro módulo de 32 salidas, respectivamente. Leer la información física es importante, ya que permite relacionar esta información con la planta real para la cual se crea el supervisor.

La aplicación auxiliar “PLC Detector” fue diseñada para detectar la configuración física y parámetros a partir del archivo SLC de manera automática, de esta manera no será necesario declarar esta parte. Un manual completo de “PLC Detector” está en el anexo B, junto con la lista de dispositivos y módulos soportados.

El resultado de la detección, es un archivo PLC el cual ya es fácil de utilizar para generar los ficheros con la estructura *ladder* que se desee. Mediante cualquier rutina de lectura de ficheros, se puede leer este archivo y, por medio de un algoritmo de manejo de cadenas (que genera el *ladder*), ensamblar el archivo SLC completo para cualquier dispositivo.

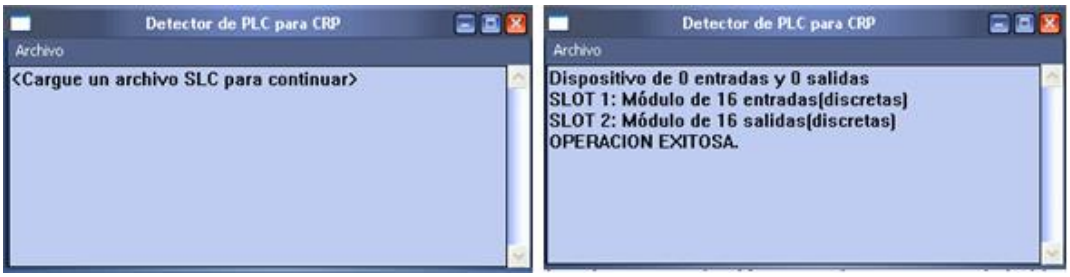


Figura 3.10 PLC Detector, detección del PLC de la sala de automática

En la siguiente tabla se explica el formato del archivo PLC generado por "PLC Detector". Los primeros 4 bytes del fichero deben leerse como un dato entero, el valor de este número indica el número de Slots de entrada que están en el dispositivo. Posteriormente, se usan 8 bytes para especificar cada slot, los primeros 4 bytes se leen como un entero indicando la posición física en el controlador, los cuatro restantes son un valor entero que indica el número de entradas del slot.

INICIO	No. slots de entrada	Posición del primer slot de entrada	No. entradas primer slot	Posición del segundo slot de entrada	No. entradas segundo slot
	4 Bytes	4 Bytes	4 Bytes	4 Bytes	4 Bytes

Tabla 3.3 Dispositivo con 2 slots de entradas (20 Bytes del fichero)

El tamaño en bytes de la especificación de los slots de entradas será entonces 8*(NI) bytes, donde NI es el valor del entero especificado en los primeros 4 bytes del fichero (No. slots de entradas). Los slots de salidas se especifican justo después de los slots de entradas, el formato es igual: los primeros cuatro bytes indican un número entero, cuyo valor es el número de slots de salidas en el dispositivo, justo después, se usan 8 bytes por cada slot, los cuatro primeros son un entero cuyo valor indica la posición y los 4 últimos un entero cuyo valor indica el número de salidas del slot.

La parte restante del fichero PLC contiene la declaración de los parámetros y la configuración física del controlador, estos datos se guardan en forma de cadena usando el código ASCII de un byte por carácter (debe tomarse en cuenta en aplicaciones que usen Unicode, como las basadas en Java).La configuración física está escrita después de la especificación de slots de salida, los primeros cuatro bytes son un valor entero que indica la longitud de la cadena en bytes y posteriormente se escribe la cadena. La cadena de parámetros está definida después de terminar la configuración física, los primeros cuatro bytes es un valor entero que indica la longitud de la cadena en bytes, luego está escrita la cadena. Las cadenas de texto no contienen ningún carácter nulo terminador.

INICIO	No. slots de entrada	Posición del primer slot de entrada	No. entradas primer slot	Posición del segundo slot de entrada	No. entradas segundo slot
	4 Bytes	4 Bytes	4 Bytes	4 Bytes	4 Bytes

No. slots de salida	Posición del primer slot de salida	No. salidas primer slot	Posición del segundo slot de salida	No. salidas segundo slot	Longitud cadena de conf. física: L_1
4 Bytes	4 Bytes	4 Bytes	4 Bytes	4 Bytes	4 Bytes

Cadena con la configuración física del dispositivo.	Longitud cadena de parámetros L_2	Cadena con la configuración de los parámetros del dispositivo	FIN
L_1 Bytes	4 Bytes	L_2 Bytes	

Tabla 3.4 Archivo PLC de un dispositivo con dos slots de entrada y de salida

El principal objetivo de explicar esta sección, es facilitar trabajo futuros enfocados en ampliar la traducción a *ladder* de otras extensiones de redes de Petri, o cualquier otro enfoque de implementación de supervisores mediante sistemas de eventos discretos.

Crear un programa *ladder* a partir del archivo PLC es una tarea sencilla, la aplicación debe tener implementado el algoritmo mediante el cual se escribirá el *ladder* mediante texto, como se explicó en las secciones 3.3.1, 3.3.2 y 3.3.3. Posteriormente, leyendo el archivo PLC, se ensambla junto con el *ladder* el archivo SLC completo para ser pasado a RSLogix.

La información de slots de entrada y salidas resulta bastante útil para que la aplicación a crear el *ladder* pueda asignarlas al diseño que está creando, una asignación posterior en RSLogix resulta difícil, ya que el *ladder* puede llegar a ser extenso y ubicar cada elemento es una tarea tediosa y propensa a errores.

Una descripción de las funciones usadas por “PLC Detector” está disponible en el anexo A.

4 HERRAMIENTA BASADA EN REDES DE PETRI PARA DISEÑO DE SUPERVISORES DE SISTEMAS DE EVENTOS DISCRETOS

En este capítulo se presenta el resultado de este proyecto, la aplicación CRP (siglas de Control Redes de Petri), la cual fue desarrollada completamente en este trabajo. A diferencia de otras aplicaciones, CRP fue desarrollado con el fin de realizar todo el proceso de diseño e implementación de supervisores y no solo como herramienta para generar código ladder, para cumplir estos propósitos, CRP fue desarrollado desde cero.. CRP es una herramienta basada en Win32 diseñada para llevar a cabo el diseño, construcción y simulación de supervisores de una manera rápida y eficaz. La aplicación está integrada con el software PIPE, en el cual se edita la red de Petri. Para la creación del supervisor, se implementa la metodología presentada en el segundo capítulo, para la cual se ha desarrollado una librería de clases y funciones capaces de abarcar esta teoría de una manera bastante completa.

La generación automática de código *ladder* se lleva a cabo como fue descrito en el capítulo tercero. Un manual completo de CRP y de la aplicación auxiliar “PLC detector”, la cual también fue desarrollada en este trabajo como una herramienta para obtener la información de la configuración física y parámetros del dispositivo a programar junto con un ejemplo ilustrativo de uso se puede encontrar en el anexo A, en este capítulo se expondrán las características generales de CRP integrado con PIPE.

En el próximo capítulo se muestran casos de estudio que fueron resueltos haciendo uso de esta herramienta.

Se han establecido los siguientes requerimientos para CRP.

- Sistema operativo: Windows7/Vista/WindowsXP/Windows2003 Server/ Windows 2000/ Windows 98, otras plataformas Windows pueden funcionar, más no están verificadas.
- Utiliza en promedio 11MB de RAM, bajo uso típico (debe sumarse los requerimientos del SO, este valor puede subir para redes muy grandes)
- 2.5MB de espacio en disco
- Según la versión de Windows y el tipo de procesador, instalar Java con el JRE.

4.1 Edición de Redes Petri

La edición de la red de Petri se realiza mediante el software PIPE 2.0, el cual está integrado con la ventana principal de CRP, cada vez que se necesita diseñar un nuevo supervisor o se abre desde un archivo existente, PIPE es lanzado para visualizar y editar la red de Petri.
La versión original de PIPE 2.0 permite la edición de redes de Petri con transiciones temporizadas o inmediatas y arcos de cualquier peso, para los requerimientos de este proyecto, se extendió dicho software para que sea posible editar otros atributos de las transiciones y lugares. Los atributos que se adicionaron son los siguientes:

1. Entradas (o eventos): se asignan a transiciones.
2. Acciones: se asignan a lugares
3. Controlabilidad: se asigna a transiciones
4. Observabilidad: se asigna a transiciones

Se recuerda al lector la diferencia entre *evento* y *transición*, descrita en la sección 3.1.3, para evitar confusiones con la teoría de DEDS.

Los dos primeros atributos son nombres, por tanto se editan como una cadena de caracteres, la controlabilidad y observabilidad se pueden tomar como una bandera para cada atributo.

Además, si la transición contiene un evento, se editan dos banderas más para esta transición con respecto a la teoría de generación de *ladder*:

1. Generar el evento una sola vez cuando la señal se active.
2. Generar el evento solo si la señal está apagada.

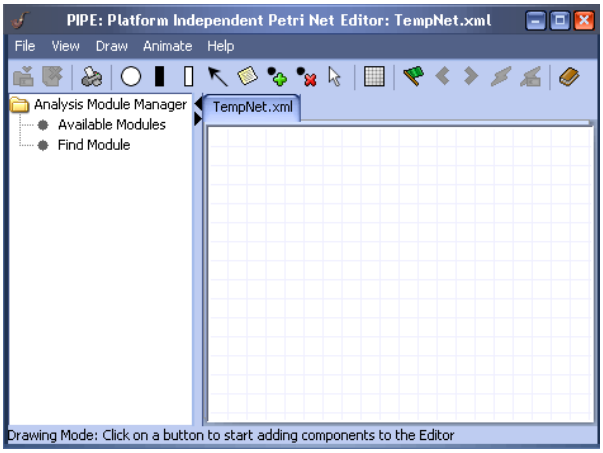


Figura 4.1 Interfaz del software PIPE

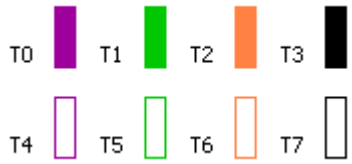


Figura 4.2 Tipos de transiciones editadas en PIPE extendido

Morado: no controlable, *verde*: no observable: *naranja*: no controlable y no observable, *negro:* normal. Arriba inmediatas, abajo temporizadas.

Esta versión modificada, se conecta mediante una tubería con nombre a CRP para compartir datos de la PN, de esta manera, mientras el usuario realiza la edición de la PN también puede acceder a las funciones de análisis de bloqueo, adición de restricciones, cálculo del supervisor y creación de la red supervisada, simulación o generación de *ladder* para la PN que está editando en PIPE. Cuando el usuario se encuentre realizando una simulación o generando código *ladder*, PIPE se ocultará momentáneamente por comodidad.

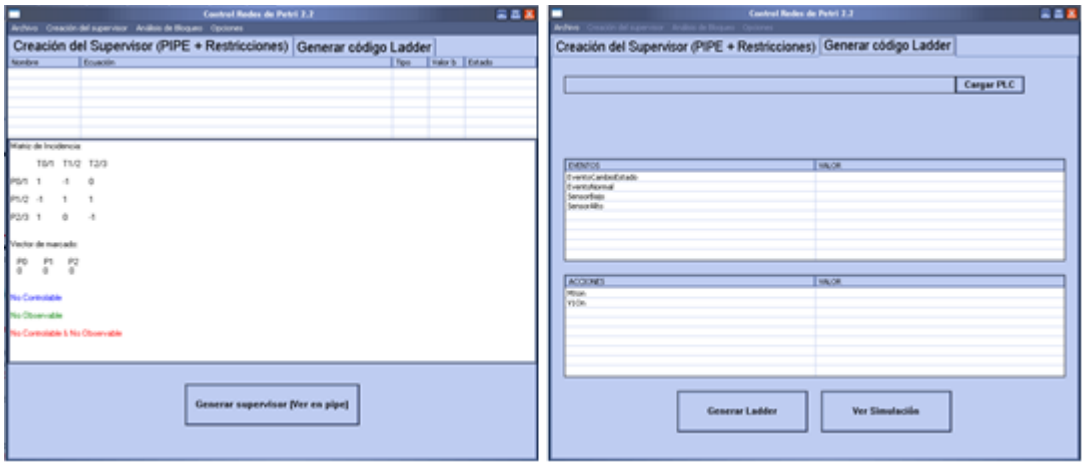


Figura 4.3 Interfaz de CRP para diseño de supervisores

CRP también calculará la matriz de incidencia y el vector de marcado a cada instante de la edición de la PN, en esta matriz, se denotan las transiciones no controlables y no observables con distintos colores, como se observa en la siguiente figura.

Figura 4.4 Matriz de incidencia y vector de marcado calculados por CRP

4.2 Herramientas de CRP

El supervisor en CRP se crea mediante la teoría expuesta en el capítulo segundo, basada en las restricciones generales. Se han implementado todas las funciones necesarias para trabajar esta metodología. La lista de funciones es.

1. Cálculo del supervisor.
2. Evaluación de la admisibilidad en las restricciones.
3. Cálculo de la matriz de incidencia, además es posible observar las columnas no observables y/o no controlables.
4. Cálculo de soluciones, para restricciones no admisibles, mediante el método del núcleo.
5. Cálculo de soluciones, para restricciones no admisibles, mediante el método de operaciones por filas.
6. Cálculo de soluciones, para restricciones no admisibles, mediante el uso de programación lineal de enteros, usando el método branch and bound.
7. Cálculo de sifones y trampas mediante la transformación de la matriz de incidencia.
8. Calculó de sifones y trampas mediante ecuaciones lógicas (método de Thelen).
9. Cálculo de sifones y trampas mediante reducción recursiva de la red.
10. Cálculo de los lugares de sifones cubiertos por invariantes (sifones controlados por invariante), y trampas (controlados por trampas), mediante esto es posible ver los bloqueos de la PN (sifones no controlados).
11. Transformación H de la red de Petri, y su respectiva inversa.
12. Transformación C de la red de Petri, y su respectiva inversa.

Estas herramientas se pueden utilizar en cualquier etapa del diseño del supervisor. En seguida se presenta de manera general las características de CRP, para un manual completo vea el anexo A.

4.2.1 Edición de restricciones

Es posible adicionar cualquier restricción de tipo (2.20) o (2.24) en CRP, una vez agregada se puede editar, eliminar y en caso de ser no admisible, aplicar un método para obtener una solución alternativa o forzarla.

La edición de estas restricciones consiste en una ecuación, en la cual mediante los nombres de lugares y transiciones, se indica los elementos que intervienen. Se usan los caracteres *u*, *q* y *v* como auxiliares para indicar respectivamente, el vector de marcado, el vector de disparo y el vector de Parikh.

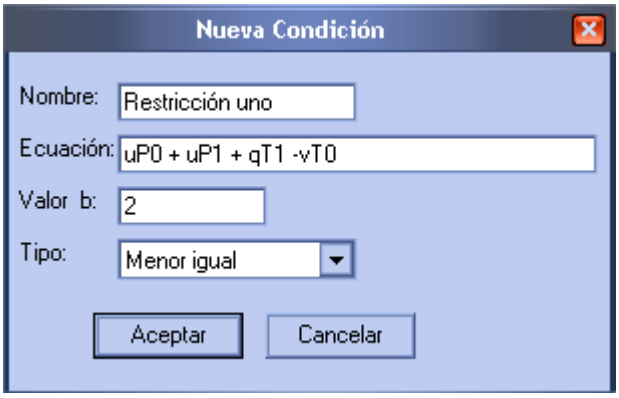


Figura 4.5 Edición de una restricción

Por ejemplo, la ecuación de la Figura 4.5 indica: el marcado del lugar P0 más el marcado del lugar P1 más el disparo de la transición T1 menos el vector de Parikh (número de disparos) de la transición T0, debe ser menor igual a 2 en todo momento. Todas las restricciones que se agreguen, se irán mostrando en la lista superior de CRP, en la pestaña “Creación del supervisor”.

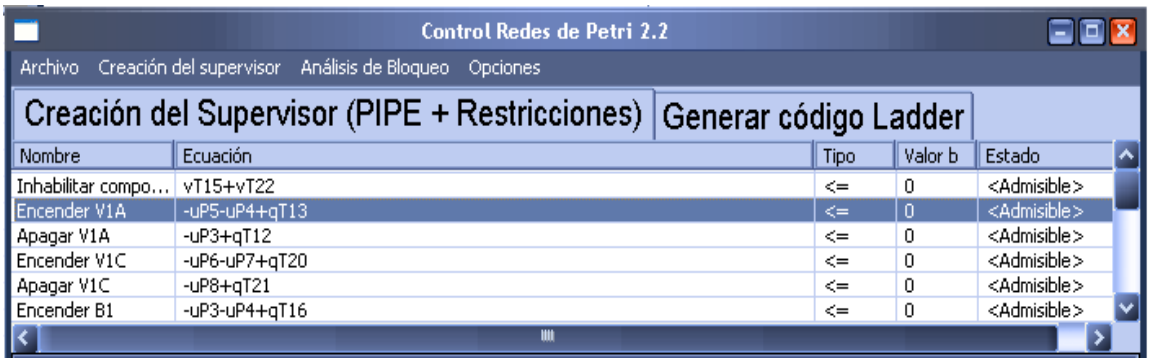


Figura 4.6 Restricciones en la lista superior

En caso de necesitar resolver una restricción no admisible, es posible aplicar los métodos de solución a esa restricción, justo después se desplegará una ventana con todas las posibles soluciones encontradas.

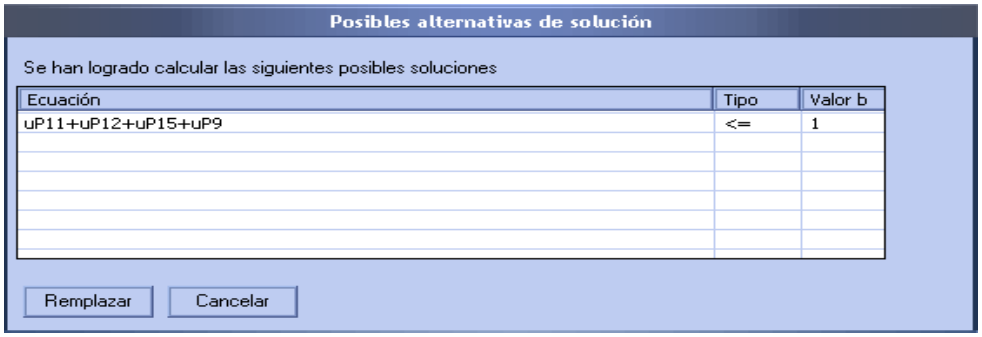


Figura 4.7 Solución hallada por CRP para una restricción

En caso de no existir alguna solución para la restricción, se recomienda buscar una restricción alternativa admisible que pueda servir el mismo propósito. Otra opción es forzar (implementar la restricción aun siendo inadmisible) la condición para que se implemente.

4.2.2 Cálculo y creación de la red supervisada

Se pueden implementar las restricciones agregadas en cualquier momento del diseño del supervisor, siempre y cuando estas sean admisibles y válidas para la red actual (el usuario podría eliminar elementos o cambiar la marca de la red, dejando algunas restricciones sin validez).

Esta opción está disponible desde el menú “Creación del supervisor” o desde el botón inferior de la pestaña con el mismo nombre, Inmediatamente después se mostrará en PIPE, en una nueva pestaña, la red con las restricciones impuestas.

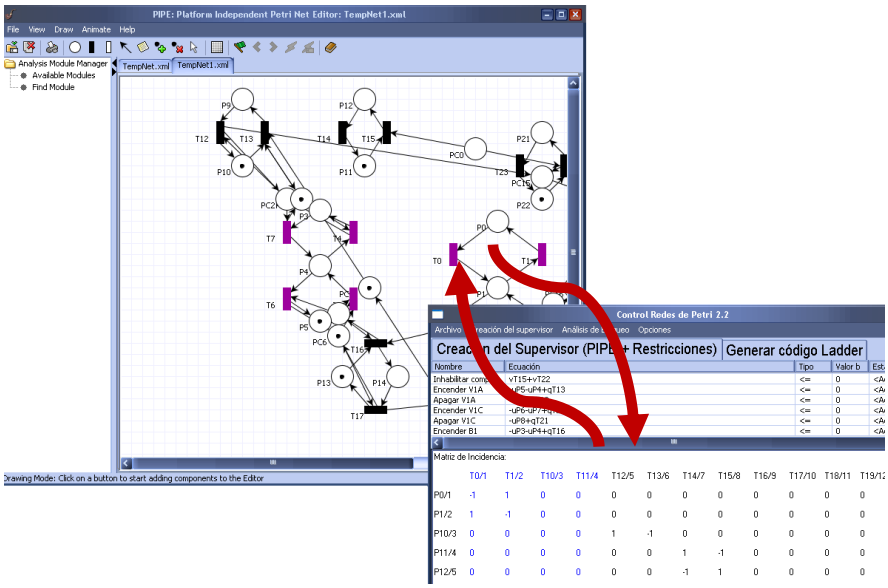


Figura 4.8 Visualización de la red supervisada en PIPE

4.2.3 Cálculo de sifones y trampas

El análisis de bloqueo realizado por CRP se basa en el cálculo de sifones y trampas, el cual es suficiente en la mayoría de los casos, para resolver algún problema de bloqueo. Para controlar sifones (evitar que pierda sus todas sus marcas), se recomienda implementar invariantes “mayor igual”, sin embargo un algoritmo o procedimiento formal para resolverlo no ha sido trabajado en este trabajo.

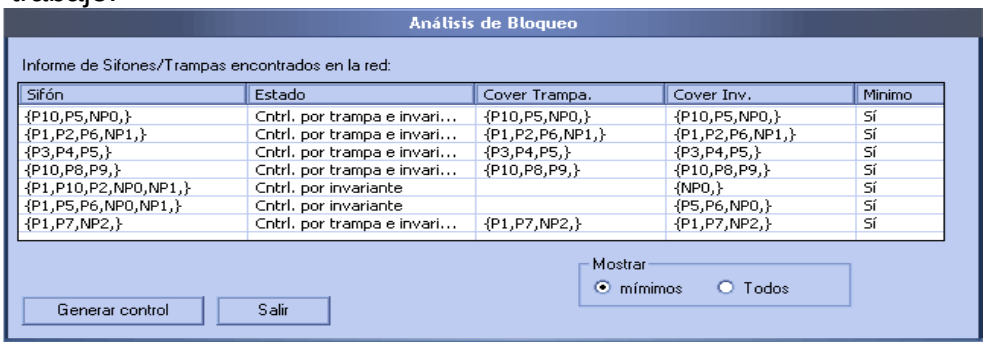


Figura 4.9 Sifones y sus distintas propiedades

CRP calcula sifones (y trampas) mediante el método de transformación de la matriz de incidencia, ecuaciones lógicas y la reducción recursiva de la red (método de Thelen). De todas formas, antes de realizar el cálculo se reduce la red aplicando reducción recursiva [19], quitando elementos innecesarios. Mediante este cálculo, es posible observar las trampas que contienen un sifón y si este es uno mínimo, adicionalmente implementa un algoritmo para determinar los lugares cubiertos por algún invariante, de esta manera es fácil observar los sifones no controlados que pueden ser causa de bloqueo. Es

posible elegir aplicar el análisis a la red original (modelo lazo abierto) o la red supervisada (con las restricciones impuestas).

Con respecto al módulo de sifones y trampas implementado en PIPE, el cálculo de los sifones en CRP ofrece la ventaja de detectar si estos están controlados por invariante de lugar o trampa. En la sección 5.2 se muestra un ejemplo al respecto.

Opcionalmente, es posible adicionar una restricción para controlar un sifón mediante el botón “Generar control”, este desplegará un diálogo como el de la Figura 4.8.

4.2.4 Generación de código *ladder* y simulación del supervisor

CRP genera código *ladder* mediante la metodología expuesta en el capítulo tercero, el resultado de esta operación es un archivo SLC para el software RSLogix 500. Toda la familia de controladores SLC-500 está considerada para ser compatible con CRP.

Antes de generar *ladder*, es necesario crear una especificación para el PLC que se utilizará, ya que este puede contener distintas configuraciones de memoria, slots con módulos de entradas/salidas discretas, o tamaño de rack, además cada dispositivo puede tener su propia configuración interna. Para realizar este procedimiento de manera fácil y rápida, fue diseñada la aplicación auxiliar “PLC Detector”, la cual acomoda la configuración de RSLogix500 para ser leída por CRP mediante un archivo de extensión PLC.
El primer paso, consiste en configurar el controlador de manera normal en el software RSLogix 500, después de esto se guarda esta configuración en un archivo SLC, “PLC Detector” toma este archivo y lo convierte a uno con extensión PLC, el cual ya se puede utilizar para la generación de *ladder*. Los detalles de esta operación fueron descritos en el capítulo tercero.

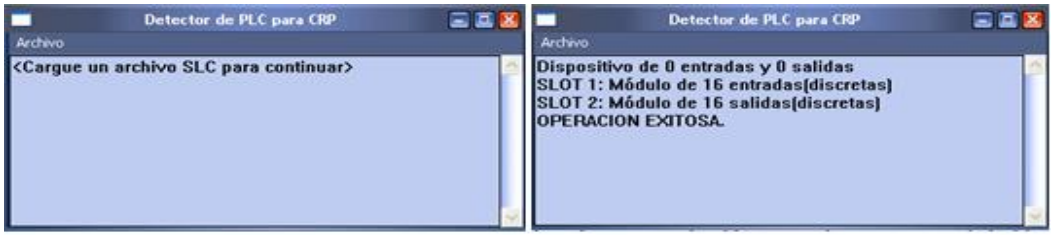


Figura 4.10 Interfaz de PLC Detector

Una vez creado el archivo PLC, solo será necesario volver a crear este archivo si tiene que utilizar un controlador distinto o su configuración cambia, por ejemplo, si se instala un rack de mayor tamaño, se adiciona otro módulo de entradas/salidas o se cambia el orden de los existentes, se configura un

nuevo valor para el watchdog o tiempo de escaneo etc. Generalmente solo será necesario realizar esta operación una vez.

Cuando se haya generado el archivo PLC, se debe cargar a CRP en la pestaña “Generación de código *ladder*”, una vez realizado esto, también es posible asignar las entradas y salidas del controlador a los eventos y acciones de la PN, esto se realiza haciendo doble clic en el evento o salida que aparecerán en las listas inferiores.

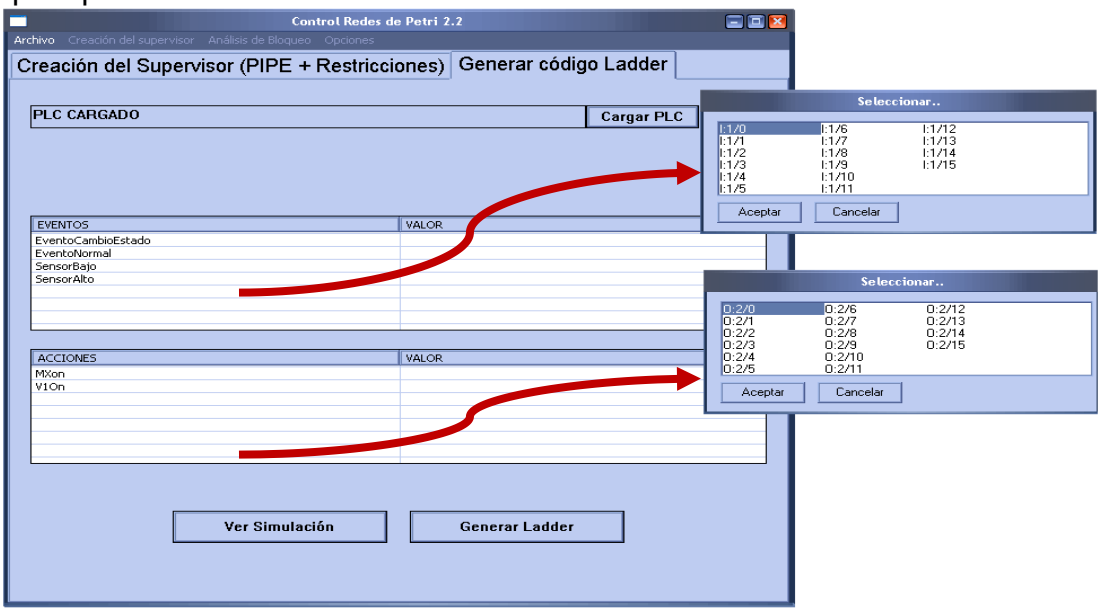


Figura 4.11 Interfaz de generación de código *ladder*

Una vez realizada la asignación de entradas y salidas, pulsando el botón “Generar *ladder*” se creará el archivo SLC para RSLogix. Es posible elegir la PN original o la creada por el supervisor para esta operación. Después de creado el programa, falta elegir la ubicación en la cual se guardará el fichero SLC.

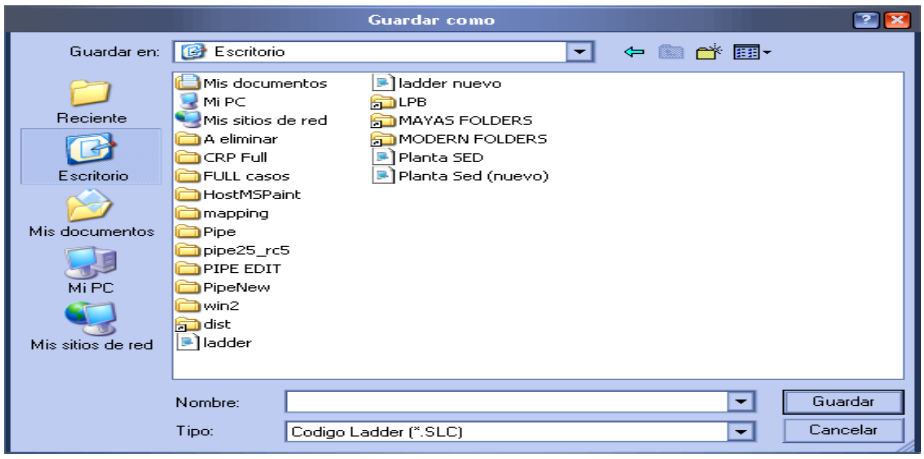


Figura 4.12 Guardado del archivo SLC

Es posible simular previamente el supervisor antes de descargarlo en un PLC, esta simulación ayuda a disminuir el tiempo de diseño detectando errores o bloqueos en el supervisor de manera rápida, sin salir de la misma herramienta. De todas formas, una última simulación en RS – View o RsLogix es recomendada.

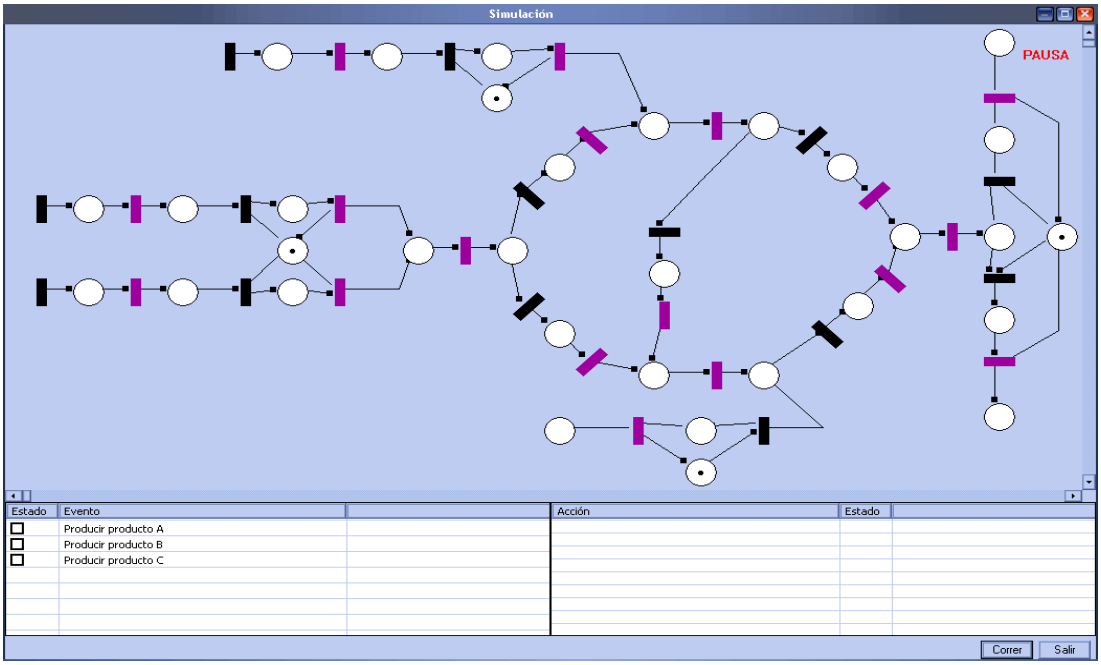


Figura 4.13 Simulación de una celda de manufactura en CRP

En la simulación, se pretende representar el comportamiento del sistema una vez descargado el código al PLC, por ello no consiste en la elección de las transiciones a disparar (como el caso de la simulación en PIPE), en cambio, el usuario puede simular la ocurrencia de eventos (entradas en el PLC) y observar las acciones que toma el supervisor ante esto. Se vuelve a hacer

énfasis en la diferencia entre *evento* y *transición*, como fue expuesto en el tercer capítulo.

La creación del supervisor está diseñada para ser máximamente permisivo, como fue expuesto en los anteriores capítulos, por tanto el controlador hará lo posible por disparar cualquier transición controlable que tenga disponible.

Las secciones a continuación se presentan como referencia a posibles trabajos futuros, que pueden partir de lo desarrollado en este trabajo.

4.3 Comunicación Windows – Java

La comunicación entre Windows (CRP) y Java (PIPE), se realiza mediante mensajes enviados por dos tuberías con nombre. El nombre de estas tuberías es:

PIPE a CRP \\.\pipe\crp-pipe-10425087362-INTERPROCESSESCOMM-I
CRP a PIPE \\.\pipe\crp-pipe-10425087362-INTERPROCESSESCOMM-O
Se usa comunicación sincrónica, por tanto solo es necesario crear un hilo en cada programa para atender la tubería de entrada y avise al proceso principal la llegada de un mensaje. Debido al limitado tamaño de los ficheros creados por PIPE, la creación de un archivo temporal, donde se almacenen los datos a compartir, es suficiente para integrar estas dos aplicaciones. Las tuberías se usan para coordinar que las dos aplicaciones no intenten acceder al mismo fichero al mismo tiempo, lo que provocaría un error, y que cada aplicación pueda notificar a la otra cuando un lote de datos está listo para ser leído.

En caso de realizar alguna extensión a este trabajo, que pueda incrementar sustancialmente el tamaño de los datos a compartir, se recomienda mapear parte de los datos en RAM, de esta manera se acelerará el proceso y no se tendrán que realizar modificaciones extensas. Sin embargo, si se trabaja en sistema operativo diferente a Windows, las funciones de creación de tuberías y ficheros deberían ser sustituidas por su equivalente, aunque PIPE no necesitará modificaciones.

Los archivos temporales usados se crean dentro del directorio de CRP, estos son:

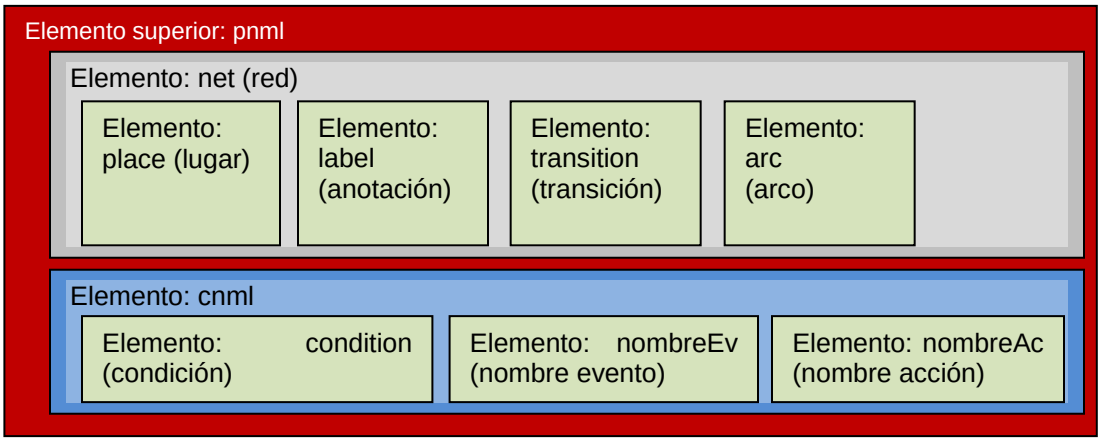
TempNet.xml => Compartir datos entre CRP y PIPE en la edición.
TempNet1.xml => Envío de la red supervisada a PIPE.

No se recomienda modificar o abrir estos archivos mientras CRP esté ejecutándose.

4.4 Ficheros escritos por PIPE y CRP

Los archivos usados por la herramienta están escritos en lenguaje XML (Extensible Markup Language), no es necesario un conocimiento avanzado de este formato para manejarlos. En el siguiente esquema se denotan los elementos existentes en el fichero y su jerarquía. En este formato, todos los datos y valores se escriben como una cadena.

El esquema del fichero XML se puede observar en el cuadro inferior, el elemento superior es “pnml”, el cual contiene dos elementos “net” y “cnml”, “net” contiene toda la información de la red de Petri, mientras “cnml” contiene los datos con respecto a las restricciones y asignación de entradas/salidas.
Una descripción de la arquitectura de los ficheros generados por PIPE y los nuevos atributos adicionados a la red de Petri en este trabajo se puede encontrar en el anexo B, esta es la información que se guarda en el elemento “net”. En esta sección se explicará el formato utilizado para guardar las restricciones y asignaciones de salidas y entradas, correspondiente al elemento “cnml” del fichero.



4.4.1 Formato para restricciones

Cada restricción contiene los siguientes atributos:

- “id” nombre de la restricción.
- “ecuacion” ecuación de la restricción
- “tipo” tipo de la restricción, puede ser mayor o igual o menor igual, para referirse a estos valores se usará la cadena CD_GREATHER_THAN y CD_LESS_THAN, respectivamente.

- “valorLimite” el parámetro b de la restricción, como ha sido explicado anteriormente.

El atributo “id” y su valor se escribe junto con la sentencia de inicio del elemento de restricción, cuya sentencia de declaración es la palabra “condition”. El resto de los atributos se declaran en orden con su valor escrito entre clausulas value. Véase el siguiente ejemplo.

```
<condition id="CondiciónUno">
        <ecuacion>
                <value>-uP8+qT27</value>
        </ecuacion>
        <tipo>
                <value>CD_LESS_THAN</value>
        </tipo>
        <valorLimite>
                <value>0</value>
        </valorLimite>
</condition>
```

La anterior restricción, estricta en forma de texto es $-\mu_8 + q_{27} \leq 0$.

4.4.2 Formato para salidas o entradas asignadas

En estos elementos se guardan las entradas del PLC asignadas a algún evento y las salidas del PLC asignadas a alguna acción.

En el caso de las entradas asignadas, solo son necesarios dos atributos:

- “id” el nombre del evento al cual se asignó la entrada
- “nom” nombre de la entrada asignada.

Estos atributos de escriben en la misma sentencia de declaración de la entrada-asignada, la cual es “nombreEv”.

Para la asignación de salidas, también son necesarios dos atributos.

- “id” el nombre de la acción a la cual se asignó la salida
- “nom” nombre de la salida asignada.

Estos atributos de escriben en la misma sentencia de declaración de la salida-asignada, la cual es “nombreAc”.

En seguida se muestran una entrada y salida asignadas.

```
        <nombreEv id="SensorAlto" nom="I:1/0" />
        <nombreAc id="AcciónUno" nom="O:2/0" />
```

Nótese que el usuario podría realizar un cambio de PLC en cualquier instante del programa, ya que antes de generar *ladder* la aplicación verifica que los nombres de las entradas y salidas asignadas sean válidos para el dispositivo actual.

4.5 Funciones y Clases Implementadas Para Extensiones Futuras

CRP fue implementado usando C/C++, los métodos matemáticos y clases que se usan para trabajar las redes de Petri y la metodología fueron escritos usando únicamente funciones estándar, por tanto es posible compilarlos en cualquier sistema operativo con cualquier compilador de C/C++. Futuras extensiones de este trabajo pueden ser llevadas a cabo fácilmente a partir de la librería desarrollada en este proyecto.

Con respecto a la creación de interfaces gráficas, escritura de archivos y comunicación con PIPE estas funciones si se deben reescribir en caso de hacer uso de otro sistema operativo diferente a Windows. En caso de utilizar alguna librería multi-plataforma, como Qt o wx, se recomienda una implementación cuidadosa para no generar ejecutables muy grandes, además de no dar demasiados problemas de instalación al usuario.

Las funciones y clases principales de la librería, están basados en las siguientes clases “menores” que modelan los diversos elementos para trabajar con redes de Petri: En esta sección se escribe una definición ilustrativa de la librería, en el manual de la librería es posible ver la definición de todas las clases y sus funciones.

Clase TRANSICION
Define una transición inmediata o temporizada de una red de Petri:

Clase LUGAR
Define un lugar de una red de Petri.

Clase ARCO
Define un arco de una red de Petri.

Clase BUCLE
Define un sifón o trampa de una red de Petri.

Clase GNERESTRICT
Define una restricción general en forma de ecuación.

Clase CONDICION

Define una restricción en forma vectorial.

Clase PLC
Define un dispositivo con su configuración de entradas, salidas, parámetros e instalación física.

Basándose en las anteriores clases, se han definido las siguientes clases que implementan funciones y objetos de mayor tamaño. Se recomienda usar estas clases ya probadas largamente antes de rehacer las mismas funciones para un trabajo similar.

Clase PETRINET

Implementa una red de Petri. Esta clase contiene todos los lugares arcos y transiciones que conforman la red, calcula la matriz de incidencia, pre, post, vector de marcado, matriz no controlable, matriz no observable y ensambla el supervisor a la red original por medio de las matrices D_c^- y D_c^+. Adicionalmente retorna todos los eventos y/o acciones presentes en la red, propiedades de las transiciones y nombres de lugares o transiciones. Con respecto a la metodología trabajada en este trabajo, esta clase implementa la transformación H y C para la red de Petri. Esta clase hace uso de diversas clases auxiliares (ver manual) y de las clases menores TRANSICION, LUGAR y ARCO.

Clase SOLVER

Implementa todas las funciones matemáticas de la librería, métodos de solución de restricciones por medio de programación lineal entera (branch and bound), Provee métodos de cálculo de sifones y trampas mediante transformación de la matriz de incidencia, ecuaciones lógicas y reducción recursiva provee métodos de evaluación de admisibilidad de restricciones, provee un método para determinar los invariantes que controlan los lugares de un sifón, convierte de condiciones generales a vectoriales y viceversa. Provee un gran número de funciones auxiliares para cualquier rutina matemática. Esta clase usa varias clases auxiliares y a clase PETRINET, CONDICION, GNERESTRICT y BUCLE.

Clase COMPILADOR

Se encarga de abrir y guardar archivos, esta función comprende abrir el fichero de la red de Petri y cargar el archivo PLC. Sirve de interfaz entre PETRINET y SOLVER para la generación de la red supervisada y se encarga de la generación de código *ladder*. Gran parte de esta clase no es útil para otros proyectos que partan desde este trabajo, se pone como referencia.

Clase CONTROL

Crea toda la interfaz gráfica de la aplicación y se comunica con PIPE. Esta clase se encarga de toda las entradas del usuario y PIPE, haciendo llegar los datos para que la clase compilador los maneje. Esta clase puede no ser útil para proyectos que partan desde este trabajo, a menos que se desee la misma interfaz gráfica. Esta clase esta implementada en CRP, ya que se necesita de una interfaz gráfica, más no es parte de la librería.

5 CASOS DE ESTUDIO

5.1 Introducción

En la presente sección se exponen un caso teórico y otro práctico donde se aplican los conceptos desarrollados en los capítulos anteriores. El primer caso de estudio ilustra el uso del control invariante para la coordinación de un sistema de manufactura flexible y prevención del bloqueo. En el segundo se aplica la metodología a un caso real desarrollado en la planta SED del laboratorio de procesos.

5.2 Celda de Manufactura Flexible

La Figura 5.1 muestra cómo están distribuidas tres máquinas (M1, M2 y M3) y dos robots (R1 y R2), en un sistema de manufactura flexible, para procesar 3 tipos diferentes de piezas de acuerdo a su plan de proceso. La pieza 1 llega a la celda mediante la cinta transportadora I1, es procesada en M1, luego R1 la lleva hasta M2 y finalmente R2 la transporta a M3 donde al finalizar el proceso esta sale por la cinta transportadora O1/O3. La pieza 2 llega por la banda I2, ingresa a M2, posteriormente R1 la transporta a M1 y abandona la celda por la banda O2. Por último la pieza 3 ingresa por I3 donde es transportada por R2 hasta M3 y al finalizar la operación sobre esta, sale por la cinta transportadora O1/O3. Las máquinas pueden trabajar simultáneamente en dos piezas iguales o distintas y cada robot solo puede transportar una pieza a la vez. Para realizar el modelo en PN se considera que todas las transiciones son controlables. Inicialmente, las tres máquinas al igual que los dos robots están disponibles. Las restricciones son:

1. C0 representa la condición $\mu_2 + \mu_3 + \mu_{13} \leq 2$ la cual significa que R1 no debe transportar piezas de tipo 1 si M2 ya está trabajando en dos piezas.
2. C1 representa la restricción $\mu_4 + \mu_5 + \mu_{15} \leq 2$ que implica que R2 no debe transportar piezas de tipo 1 si M3 está trabajando en dos piezas.
3. C2 representa la restricción $\mu_5 + \mu_{14} + \mu_{15} \leq 2$ que significa que R2 no debe transportar piezas de tipo 3 si M3 está trabajando en dos piezas.
4. C3 es el lugar para la condición $\mu_1 + \mu_{11} + \mu_{12} \leq 2$ que significa que R1 no debe transportar piezas de tipo 2 si M1 está trabajando en dos piezas.

P1	M1 trabajando en pieza 1	T1	Llegada de pieza 1 a M1
P2	R1 transportando pieza 1 a M2	T2	Salida de pieza 1 desde M1
P3	M2 trabajando en pieza 1	T3	Llegada de pieza 1 a M2
P4	R2 transportando pieza 1 a M3	T4	Salida de pieza 1 desde M2
P5	M3 trabajando en pieza 1	T5	Llegada de pieza 1 a M3
P6	M1 disponible	T6	Salida de pieza 1 desde M3
P7	R1 disponible	T7	Salida de pieza 2 desde M1
P8	M2 disponible	T8	Llegada de pieza 2 a M1
P9	R2 disponible	T9	Salida de pieza 2 desde M2
P10	M3 disponible	T10	Llegada de pieza 2 a M2
P11	M1 trabajando en pieza 2	T11	Llegada de pieza 3 a R2
P12	R1 transportando pieza 2 a M1	T12	Llegada de pieza 3 a M3
P13	M2 trabajando en pieza 2	T13	Salida de pieza 3 desde M3
P14	R2 transportando pieza 3 a M3		
P15	M3 trabajando en pieza 3		

Tabla 5.1 Descripción de lugares y transiciones FMS

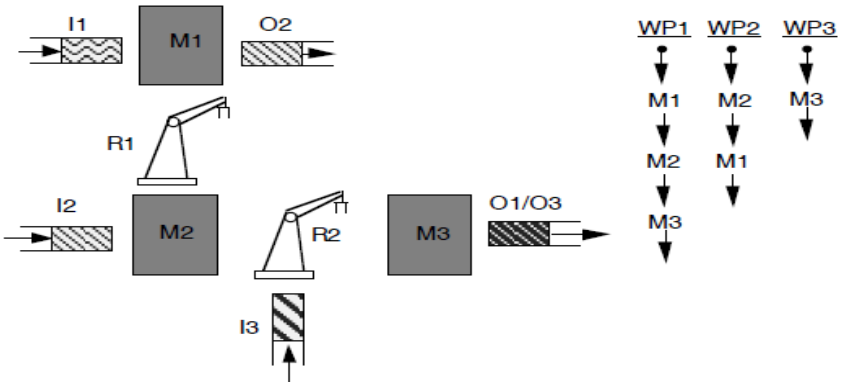


Figura 5.1 Distribución en planta y programación para 3 tipos de piezas [42]

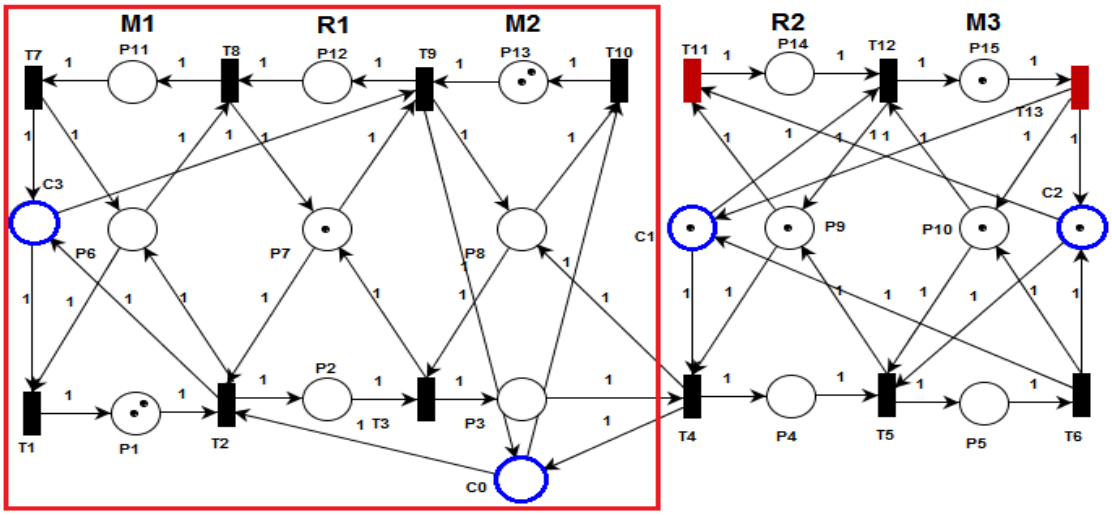


Figura 5.2 Supervisor con bloqueo parcial

Mediante el disparo aleatorio de las transiciones en PIPE se determinó que el supervisor de la Figura 5.2 presenta bloqueo parcial en la parte de la PN enmarcada por el rectángulo.

Nota: En [22] y [23] cuando todas las transiciones se tornan muertas, es decir, no pueden dispararse porque están inhabilitadas, se define bloqueo total o general de la PN. Cuando solo algunas transiciones se tornan muertas se habla de bloqueo parcial, así pues, el supervisor de la Figura 5.2 solo presenta bloqueo parcial en la parte señalada. Aunque la herramienta Space State Analysis de PIPE determinó que la PN no está bloqueada porque no existe bloqueo general de ésta.

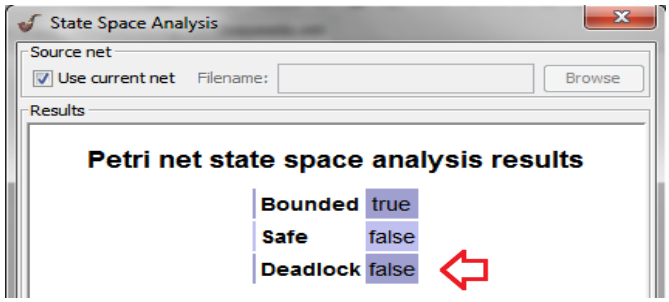


Figura 5.3 Resultado de Space State Analysis para el DEC de la Figura 5.2

Con la herramienta CRP se encontraron 34 sifones mínimos y con PIPE 20, observando los lugares que están involucrados en los sifones mínimos, se tiene que C0, C3, y P11 conforman varios de estos, entre ellos {C0, C3, P11, P12, P2, P3}, observe que estos lugares están dentro de la zona demarcada por el rectángulo en la Figura 5.2, además, CRP determinó que este sifón no está controlado. Ver Figura 5.4. Note que a diferencia de PIPE, CRP además de establecer los sifones de la PN, determina si estos están controlados o no, ya sea por invariante de lugar o trampa, lo que facilita determinar que sifón o sifones se deben controlar.

Para controlar los sifones de la PN, los lugares de control C0, C1, C2 y C3 serán representados en el vector de marca como μ_{16}, μ_{17}, μ_{18} y μ_{19} respectivamente, entonces la restricción a imponer para prevenir el bloqueo es: $\mu_{16} + \mu_{19} + \mu_{11} + \mu_{12} + \mu_2 + \mu_3 \geq 1$, la cual representa C4 en la Figura 5.6.

Una solución similar se obtiene al realizar un análisis del comportamiento de la PN mediante simulación en PIPE, ya que M1 y M2 no deben trabajar en más de dos piezas del mismo tipo (pieza 1, pieza 2) a la vez, si la otra está trabajando en una pieza del otro tipo, lo que se logra imponiendo la restricción $\mu_1 + \mu_{13} \leq 3$. Mediante el disparo aleatorio de las transiciones en PIPE del DEC de la Figura 5.6, se comprobó que la PN está libre de bloqueo. La Figura 5.7 muestra la matriz de incidencia de éste y el vector de marcado

que calculó CRP. En síntesis, con cualquiera de las dos soluciones, el sistema se hace menos permisivo e ineficiente pero se garantiza que este no entre en bloqueo.

Sifón	Estado	Cover Trampa.	Cover Inv.	Mi
{C2,P14,P15,P5,}	Cntrl. por invariante		{P14,}	Sí
{C0,C1,C2,C3,P11,P12,P15,P5,}	No controlado			Sí
{C0,C3,P11,P12,P14,P4,P9,}	No controlado			Sí
{C0,C1,C3,P11,P12,P15,P4,P5,}	Cntrl. por invariante		{P4,}	Sí
{C0,C3,P11,P12,P2,P3,}	No controlado			Sí
{C0,C3,P11,P12,P3,P8,}	No controlado			Sí
{C3,P1,P11,P12,}	No controlado			Sí
{C4,P1,P11,P12,P13,}	No controlado			Sí
{C0,C1,C2,C3,P11,P15,P5,P6,}	No controlado			Sí
{C0,C1,C2,P11,P13,P15,P5,P6,}	No controlado			Sí
{C1,C2,P11,P15,P5,P6,P7,P8,}	Cntrl. por invariante		{P8,}	Sí
{C0,C3,P11,P14,P4,P6,P9,}	Cntrl. por invariante		{C0,C3,P11,P14,P4,...	Sí
{C0,P11,P13,P14,P4,P6,P9,}	Cntrl. por invariante		{C0,P14,P4,P9,}	Sí
{P11,P14,P4,P6,P7,P8,P9,}	Cntrl. por invariante		{P11,P14,P4,P6,P7,P...	Sí
{C0,C1,C3,P11,P15,P4,P5,P6,}	No controlado			Sí
{C0,C1,P11,P13,P15,P4,P5,P6,}	No controlado			Sí
{C1,P11,P15,P4,P5,P6,P7,P8,}	Cntrl. por invariante		{P8,}	Sí
{C0,C3,P11,P2,P3,P6,}	Cntrl. por invariante		{C0,C3,P11,P2,P3,P6,}	Sí
{C0,P11,P13,P2,P3,P6,}	No controlado			Sí
{C0,C3,P11,P3,P6,P8,}	No controlado			Sí
{C0,P11,P13,P3,P6,P8,}	No controlado			Sí
{P1,P11,P6,}	No controlado			Sí
{P11,P2,P6,P7,}	No controlado			Sí
{P11,P3,P6,P7,P8,}	No controlado			Sí
{C0,C1,C2,P11,P12,P13,P15,P5,}	No controlado			Sí
{C1,C2,P11,P12,P15,P5,P7,P8,}	Cntrl. por invariante		{P8,}	Sí
{C0,P11,P12,P13,P14,P4,P9,}	Cntrl. por invariante		{P14,P4,P9,}	Sí
{P11,P12,P14,P4,P7,P8,P9,}	Cntrl. por invariante		{P14,P4,P8,}	Sí
{C0,C1,P11,P12,P13,P15,P4,P5,}	No controlado			Sí
{C1,P11,P12,P15,P4,P5,P7,P8,}	Cntrl. por invariante		{P4,P8,}	Sí
{P10,P15,P5,}	No controlado			Sí
{C0,P12,P13,P2,P3,}	No controlado			Sí
{P12,P2,P7,}	No controlado			Sí
{C0,P12,P13,P3,P8,}	No controlado			Sí

Figura 5.4 Sifones calculados por CRP

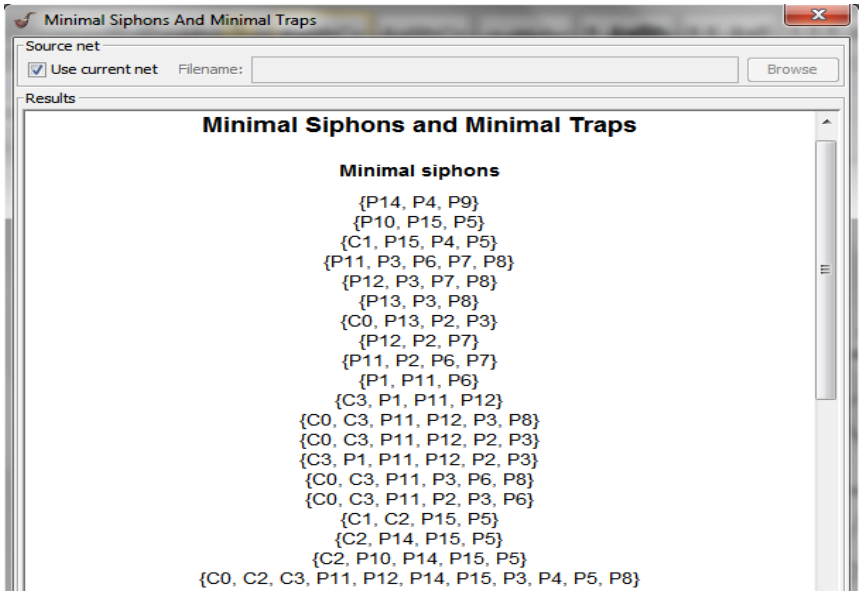


Figura 5.5 Sifones calculados por PIPE

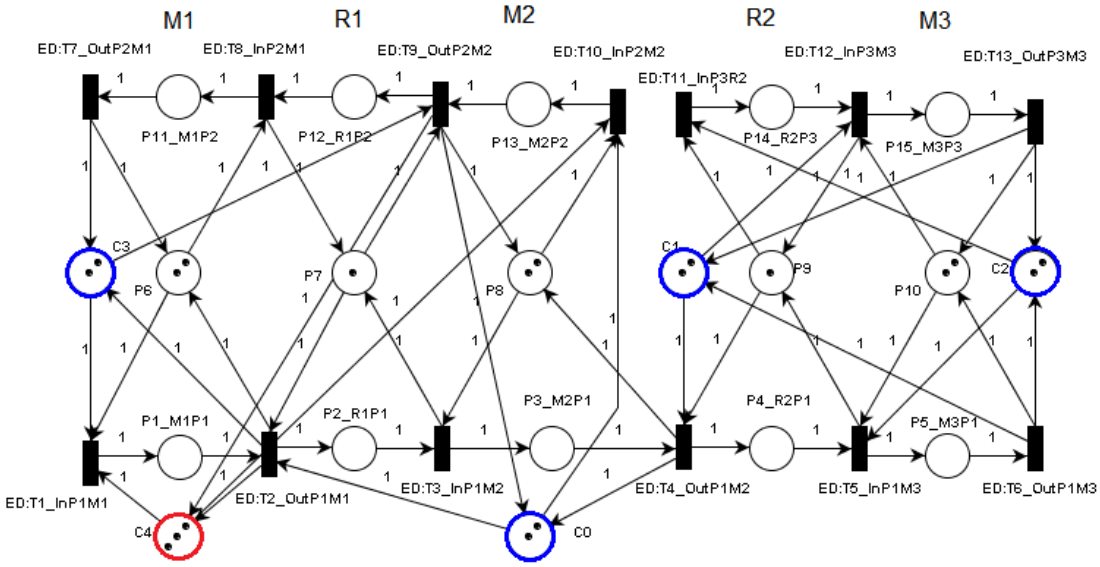


Figura 5.6 Supervisor FMS

Matriz de Incidencia:

	T1/1	T10/2	T11/3	T12/4	T13/5	T2/6	T3/7	T4/8	T5/9	T6/10	T7/11	T8/12	T9/13
C0/1	0	-1	0	0	0	-1	0	1	0	0	0	0	1
C1/2	0	0	0	-1	1	0	0	-1	0	1	0	0	0
C2/3	0	0	-1	0	1	0	0	0	-1	1	0	0	0
C3/4	-1	0	0	0	0	1	0	0	0	0	1	0	-1
C4/5	-1	-1	0	0	0	1	0	0	0	0	0	0	1
P1/6	1	0	0	0	0	-1	0	0	0	0	0	0	0
P10/7	0	0	0	-1	1	0	0	0	-1	1	0	0	0
P11/8	0	0	0	0	0	0	0	0	0	0	-1	1	0
P12/9	0	0	0	0	0	0	0	0	0	0	0	-1	1
P13/10	0	1	0	0	0	0	0	0	0	0	0	0	-1
P14/11	0	0	1	-1	0	0	0	0	0	0	0	0	0
P15/12	0	0	0	1	-1	0	0	0	0	0	0	0	0
P2/13	0	0	0	0	0	1	-1	0	0	0	0	0	0
P3/14	0	0	0	0	0	0	1	-1	0	0	0	0	0
P4/15	0	0	0	0	0	0	0	1	-1	0	0	0	0
P5/16	0	0	0	0	0	0	0	0	1	-1	0	0	0
P6/17	-1	0	0	0	0	1	0	0	0	0	1	-1	0
P7/18	0	0	0	0	0	-1	1	0	0	0	0	1	-1
P8/19	0	-1	0	0	0	0	-1	1	0	0	0	0	1
P9/20	0	0	-1	1	0	0	0	-1	1	0	0	0	0

Vector de marcado:

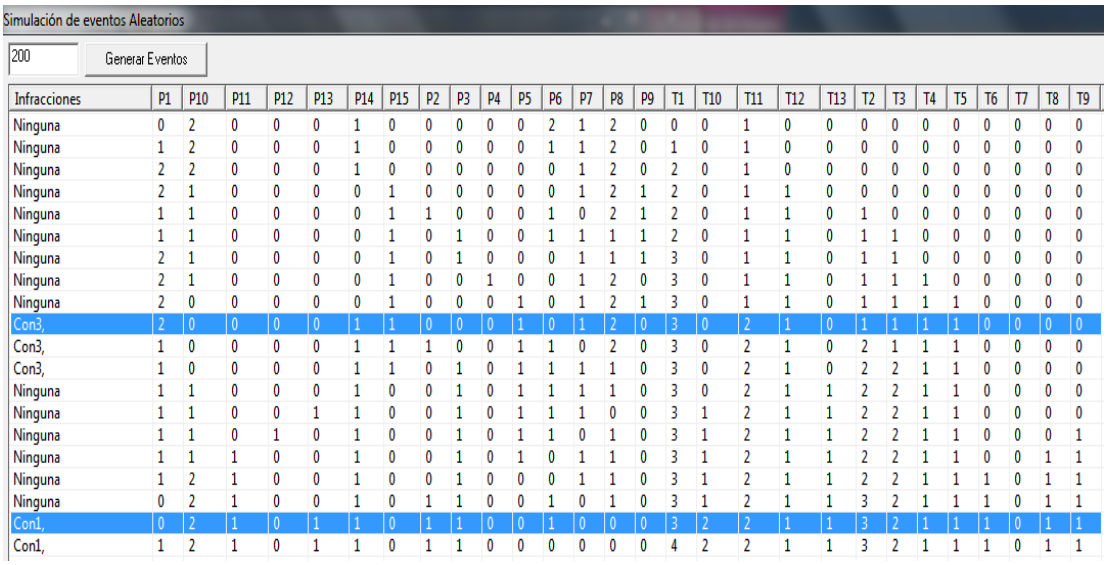
C0	C1	C2	C3	C4	P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9
2	2	2	2	3	0	2	0	0	0	0	0	0	0	0	0	2	1	2	1

Figura 5.7 Matriz de incidencia y vector de marcado calculados por CRP para el DEC de la Figura 5.6

5.2.1 Verificación del supervisor

CRP está dotado de una herramienta denominada “Prueba eventos aleatorios”, que permite generar un número determinado de eventos

aleatorios para determinar qué restricciones son violadas. Para éste caso primero se generan los eventos sin implementar las cuatro primeras restricciones, es decir, sin generar sus respectivos lugares de control, con el objetivo de mostrar los casos en donde se incumplen éstas.

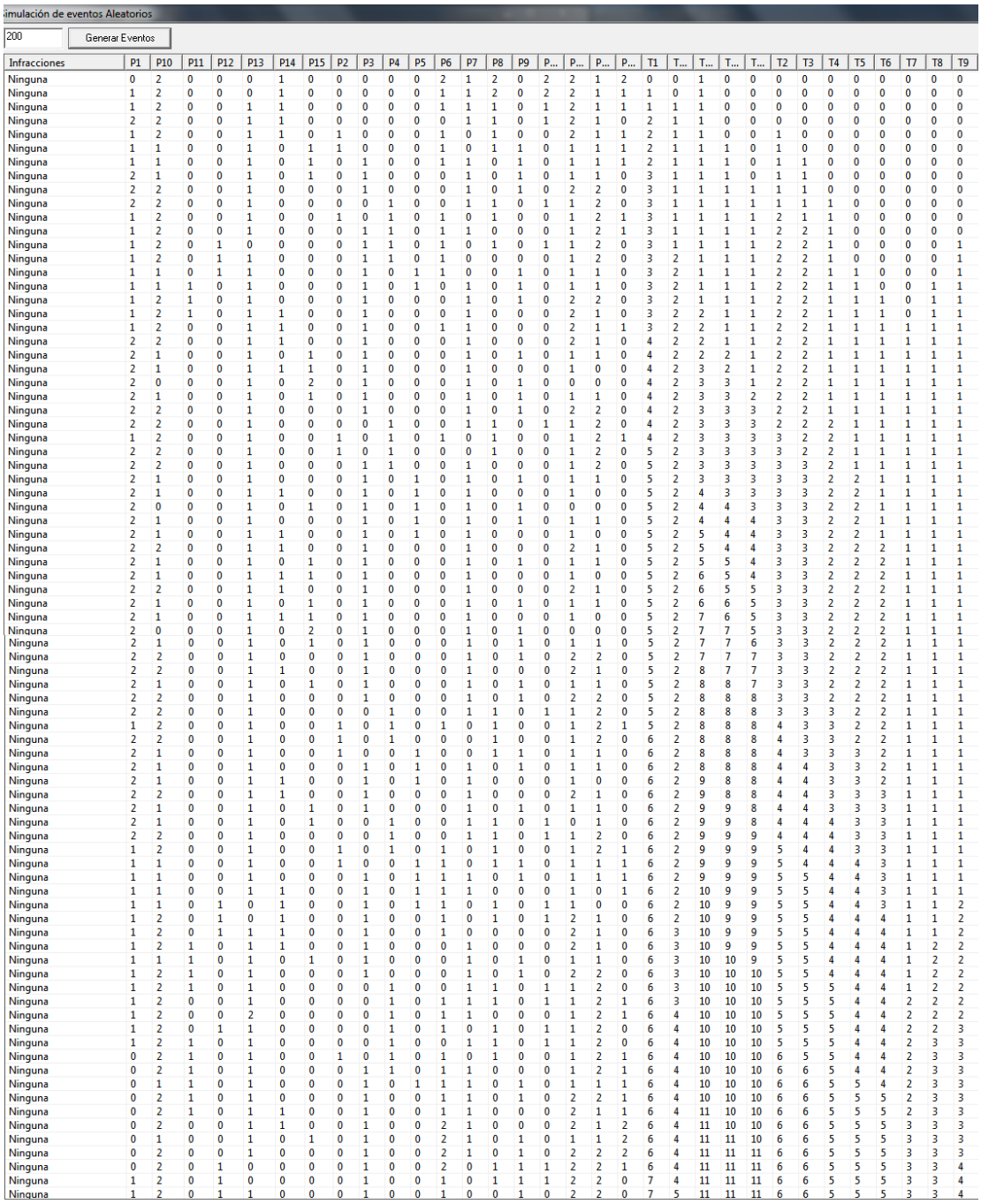


Infracciones	P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9	T1	T10	T11	T12	T13	T2	T3	T4	T5	T6	T7	T8	T9
Ninguna	0	2	0	0	0	1	0	0	0	0	0	2	1	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Ninguna	1	2	0	0	0	1	0	0	0	0	0	1	1	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0
Ninguna	2	2	0	0	0	1	0	0	0	0	0	0	1	2	0	2	0	1	0	0	0	0	0	0	0	0	0	0
Ninguna	2	1	0	0	0	0	1	0	0	0	0	0	1	2	1	2	0	1	1	0	0	0	0	0	0	0	0	0
Ninguna	1	1	0	0	0	0	1	1	0	0	0	1	0	2	1	2	0	1	1	0	1	0	0	0	0	0	0	0
Ninguna	1	1	0	0	0	0	1	0	1	0	0	1	1	1	1	2	0	1	1	0	1	1	0	0	0	0	0	0
Ninguna	2	1	0	0	0	0	1	0	1	0	0	0	1	1	1	3	0	1	1	0	1	1	0	0	0	0	0	0
Ninguna	2	1	0	0	0	0	1	0	0	1	0	0	1	2	0	3	0	1	1	0	1	1	1	0	0	0	0	0
Ninguna	2	0	0	0	0	0	1	0	0	0	1	0	1	2	1	3	0	1	1	0	1	1	1	1	0	0	0	0
Con3,	2	0	0	0	0	1	1	0	0	0	1	0	1	2	0	3	0	2	1	0	1	1	1	1	0	0	0	0
Con3,	1	0	0	0	0	1	1	1	0	0	1	1	0	2	0	3	0	2	1	0	2	1	1	1	0	0	0	0
Con3,	1	0	0	0	0	1	1	0	1	0	1	1	1	1	0	3	0	2	1	0	2	2	1	1	0	0	0	0
Ninguna	1	1	0	0	0	1	0	0	1	0	1	1	1	1	0	3	0	2	1	1	2	2	1	1	0	0	0	0
Ninguna	1	1	0	0	1	1	0	0	1	0	1	1	1	0	0	3	1	2	1	1	2	2	1	1	0	0	0	0
Ninguna	1	1	0	1	0	1	0	0	1	0	1	1	0	1	0	3	1	2	1	1	2	2	1	1	0	0	0	1
Ninguna	1	1	1	0	0	1	0	0	1	0	1	0	1	1	0	3	1	2	1	1	2	2	1	1	0	0	1	1
Ninguna	1	2	1	0	0	1	0	0	1	0	0	0	1	1	0	3	1	2	1	1	2	2	1	1	1	0	1	1
Ninguna	0	2	1	0	0	1	0	1	1	0	0	1	0	1	0	3	1	2	1	1	3	2	1	1	1	0	1	1
Con1,	0	2	1	0	1	1	0	1	1	0	0	1	0	0	0	3	2	2	1	1	3	2	1	1	1	0	1	1
Con1,	1	2	1	0	1	1	0	1	1	0	0	0	0	0	0	4	2	2	1	1	3	2	1	1	1	0	1	1

Figura 5.8 Infracción de las restricciones 1 y 3

La Figura 5.8 muestra solo los primeros disparos ya que por cuestiones de espacio no se muestra la totalidad de éstos, sin embargo, observe los casos resaltados en la Figura donde se incumplen las condiciones 1 ($\mu_2 + \mu_3 + \mu_{13} \leq 2$) y 3 ($\mu_5 + \mu_{14} + \mu_{15} \leq 2$) denominadas Con1 y Con3 respectivamente. Note que en el primer caso la suma pesada de las marcas de los lugares P5, P14 y P15 no debe ser mayor que 2, pero en la Figura 5.8 se observa que si se suman las marcas contenidas por estos lugares en esa marcación de la PN, dicha suma es 3. De igual manera observe que en el segundo caso la suma pesada de las marcas de los lugares P2, P3 y P13 no debe ser mayor que 2, pero en la Figura 5.8 se observa que si se suman las marcas contenidas por estos lugares en esa marcación de la PN, la suma es 3. Debido a que estas condiciones solo están escritas más no implementadas, es decir, no se ha generado aún su respectivo lugar de control.

La Figura 5.9 muestra solo una parte de los 200 eventos aleatorios generados para verificar el cumplimiento de las primeras cuatro restricciones, por cuestiones de espacio no son mostrados todos los estados de la PN. Note que el marcado de la PN no infringe en ningún caso las restricciones impuestas sobre el comportamiento de la PN.



Infracciones	P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9	P...	P...	P...	P...	T1	T...	T...	T...	T...	T2	T3	T4	T5	T6	T7	T8	T9
Ninguna	0	2	0	0	0	1	0	0	0	0	0	2	1	2	0	2	2	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0
Ninguna	1	2	0	0	0	1	0	0	0	0	0	1	1	2	0	2	2	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Ninguna	1	2	0	0	1	1	0	0	0	0	0	1	1	1	0	1	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Ninguna	2	2	0	0	1	1	0	0	0	0	0	0	1	1	0	1	2	1	0	2	1	1	0	0	0	0	0	0	0	0	0	0
Ninguna	1	2	0	0	1	1	0	1	0	0	0	1	0	1	0	0	2	1	1	2	1	1	0	0	1	0	0	0	0	0	0	0
Ninguna	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	1	1	1	2	1	1	1	0	1	0	0	0	0	0	0	0
Ninguna	1	1	0	0	1	0	1	0	0	0	1	1	0	1	0	1	1	1	2	1	1	1	1	0	1	1	0	0	0	0	0	0
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	3	1	1	1	0	1	1	0	0	0	0	0	0
Ninguna	2	2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	3	1	1	1	1	1	1	0	0	0	0	0	0
Ninguna	2	2	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	3	1	1	1	1	1	1	0	0	0	0	0	0
Ninguna	1	2	0	0	1	0	0	1	0	1	0	1	0	1	0	0	1	2	1	3	1	1	1	1	2	1	1	0	0	0	0	0
Ninguna	1	2	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1	2	1	3	1	1	1	1	2	2	1	0	0	0	0	0
Ninguna	1	2	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	2	0	3	1	1	1	1	2	2	1	0	0	0	0	1
Ninguna	1	2	0	1	1	0	0	0	1	1	0	1	0	0	0	0	1	2	0	3	2	1	1	1	2	2	1	0	0	0	0	1
Ninguna	1	1	0	1	1	0	0	0	1	0	1	1	0	0	1	0	1	1	0	3	2	1	1	1	2	2	1	1	0	0	0	1
Ninguna	1	1	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1	0	3	2	1	1	1	2	2	1	1	1	0	1	1
Ninguna	1	2	1	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	3	2	1	1	1	2	2	1	1	1	0	1	1
Ninguna	1	2	1	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	3	2	2	1	1	2	2	1	1	1	0	1	1
Ninguna	1	2	0	0	1	1	0	0	1	0	0	1	1	0	0	0	2	1	1	3	2	2	1	1	2	2	1	1	1	1	1	1
Ninguna	2	2	0	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	4	2	2	1	1	2	2	1	1	1	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	4	2	2	2	1	2	2	1	1	1	1	1	1
Ninguna	2	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	4	2	3	2	1	2	2	1	1	1	1	1	1
Ninguna	2	0	0	0	1	0	2	0	1	0	0	0	1	0	1	0	0	0	0	4	2	3	3	1	2	2	1	1	1	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	4	2	3	3	2	2	2	1	1	1	1	1	1
Ninguna	2	2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	4	2	3	3	3	2	2	1	1	1	1	1	1
Ninguna	2	2	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	4	2	3	3	3	2	2	1	1	1	1	1	1
Ninguna	1	2	0	0	1	0	0	1	0	1	0	1	0	1	0	0	1	2	1	4	2	3	3	3	3	2	2	1	1	1	1	1
Ninguna	2	2	0	0	1	0	0	1	0	1	0	0	1	0	0	0	1	2	0	5	2	3	3	3	3	2	2	1	1	1	1	1
Ninguna	2	2	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	2	0	5	2	3	3	3	3	3	2	1	1	1	1	1
Ninguna	2	1	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	1	0	5	2	3	3	3	3	2	2	1	1	1	1	1
Ninguna	2	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0	5	2	4	3	3	3	3	2	2	1	1	1	1
Ninguna	2	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1	0	5	2	4	4	4	3	3	2	2	1	1	1	1
Ninguna	2	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	1	0	0	5	2	5	4	4	3	3	2	2	1	1	1	1
Ninguna	2	2	0	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	5	2	5	4	4	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	5	2	5	5	4	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	1	1	0	1	0	0	0	1	0	1	0	1	0	0	5	2	6	5	4	3	3	2	2	2	1	1	1
Ninguna	2	2	0	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	5	2	6	5	5	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	5	2	6	6	5	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	5	2	7	6	5	3	3	2	2	2	1	1	1
Ninguna	2	0	0	0	1	0	2	0	1	0	0	0	1	0	1	0	0	0	0	5	2	7	7	5	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	5	2	7	7	6	3	3	2	2	2	1	1	1
Ninguna	2	2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	5	2	7	7	7	3	3	2	2	2	1	1	1
Ninguna	2	2	0	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	5	2	8	7	7	3	3	2	2	2	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	5	2	8	8	7	3	3	2	2	2	1	1	1
Ninguna	2	2	0	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	5	2	8	8	8	3	3	2	2	2	1	1	1
Ninguna	2	2	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	5	2	8	8	8	3	3	3	2	2	1	1	1
Ninguna	1	2	0	0	1	0	0	1	0	1	0	1	0	1	0	0	1	2	1	5	2	8	8	8	4	3	3	2	2	1	1	1
Ninguna	2	2	0	0	1	0	0	1	0	1	0	0	0	1	0	0	1	2	0	6	2	8	8	8	4	3	3	2	2	1	1	1
Ninguna	2	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	1	1	0	6	2	8	8	8	4	3	3	2	2	1	1	1
Ninguna	2	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	0	6	2	8	8	8	4	4	3	3	2	1	1	1
Ninguna	2	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	6	2	9	8	8	4	4	3	3	2	1	1	1
Ninguna	2	2	0	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	6	2	9	8	8	4	4	3	3	3	1	1	1
Ninguna	2	1	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	6	2	9	9	8	4	4	3	3	3	1	1	1
Ninguna	2	1	0	0	1	0	1	0	0	1	0	0	1	1	0	1	0	1	0	6	2	9	9	8	4	4	4	3	3	1	1	1
Ninguna	2	2	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	6	2	9	9	9	4	4	4	3	3	1	1	1
Ninguna	1	2	0	0	1	0	0	1	0	1	0	1	0	1	0	0	1	2	1	6	2	9	9	9	5	4	4	3	3	1	1	1
Ninguna	1	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	1	6	2	9	9	9	5	4	4	4	3	1	1	1
Ninguna	1	1	0	0	1	0	0	0	1	0	1	1	0	1	0	0	1	1	0	6	2	9	9	9	5	5	4	4	3	1	1	1
Ninguna	1	1	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	6	2	10	9	9	5	5	4	4	3	1	1	1
Ninguna	1	1	0	1	0	1	0	0	1	0	1	1	0	1	0	1	1	0	0	6	2	10	9	9	5	5	4	4	3	1	1	2
Ninguna	1	2	0	1	0	1	0	0	1	0	0	1	0	1	0	1	2	1	0	6	2	10	9	9	5	5	4	4	4	1	1	2
Ninguna	1	2	0	1	1	1	0	0	1	0	0	1	0	0	0	0	2	1	0	6	3	10	9	9	5	5	4	4	4	1	1	2
Ninguna	1	2	1	0	1	1	0	0	1	0	0	0	1	0	0	0	2	1	0	6	3	10	9	9	5	5	4	4	4	1	2	2
Ninguna	1	1	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	0	6	3	10	10	9	5	5	4	4	4	1	2	2
Ninguna	1	2	1	0	1	0	0	0	1	0	0	0	1	0	1	0	2	2	0	6	3	10	10	10	5	5	4	4	4	1	2	2
Ninguna	1	2	1	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	6	3	10	10	10	5	5	5	4	4	1	2	2
Ninguna	1	2	0	0	1	0	0	0	1	0	0	1	1	1	0	1	1	2	1	6	3	10	10	10	5	5	5	4	4	1	2	2
Ninguna	1	2	0	0	2	0	0	0	0	1	0	1	1	0	0	0	1	2	1	6	4	10	10	10	5	5	5	4	4	2	2	2
Ninguna	1	2	0	1	1	0	0	0	0	1	0	1	0	1	0	1	1	2	0	6	4	10	10	10	5	5	5	4	4	2	2	3
Ninguna	1	2	1	0	1	0	0	0	0	1	0	0	1	1	0	1	1	2	0	6	4	10	10	10	5	5	5	4	4	2	3	3
Ninguna	0	2	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1	2	1	6	4	10	10	10	6	5	5	4	4	2	3	3
Ninguna	0	2	1	0	1	0	0	0	1	1	0	1	1	0	0	0	1	2	1	6	4	10	10	10	6	6	5	4	4	2	3	3
Ninguna	0	1	1	0	1	0	0	0	1	0	1	1	1	0	1	0	1	1	1	6	4	10	10	10	6	6	5	5	4	2	3	3
Ninguna	0	2	1	0	1	0	0	0	1	0	0	1	1	0	1	0	2	2	1	6	4	10	10	10	6	6	5	5	4	2	3	3
Ninguna	0	2	1	0	1	1	0	0	1	0	0	1	1	0	0	0	2	1	1	6	4	11	10	10	6	6	5	5	5	2	3	3
Ninguna	0	2	0	0	1	1	0	0	1	0	0	2	1	0	0	0	2	1	2	6	4	11	10	10	6	6	5	5	5	3	3	3
Ninguna	0	1	0	0	1	0	1	0	1	0	0	2	1	0	1	0	1	1	2	6	4	11	11	10	6	6	5	5	5	3	3	3
Ninguna	0	2	0	0	1	0	0	0	1	0	0	2	1	0	1	0	2	2	2	6	4	11	11	11	6	6	5	5	5	3	3	3
Ninguna	0	2	0	1	0	0	0	0	1	0	0	2	0	1	1	1	2	2	1	6	4	11	11	11	6	6	5	5	5	3	3	4
Ninguna	1	2	0	1	0	0	0	0	1	0	0	1	0	1	1	1	2	2	0	7	4	11	11	11	6	6	5	5	5	3	3	4
Ninguna	1	2	0	1	1	0	0	0	1	0	0	1	0	0	1	0	2	2	0	7	5	11	11	11	6	6	5	5	5	3	3	4

Figura 5.9 Validación del supervisor

A continuación se muestra una fracción del código *ladder* generado por CRP para la implementación en el PLC Micrologix 1500.

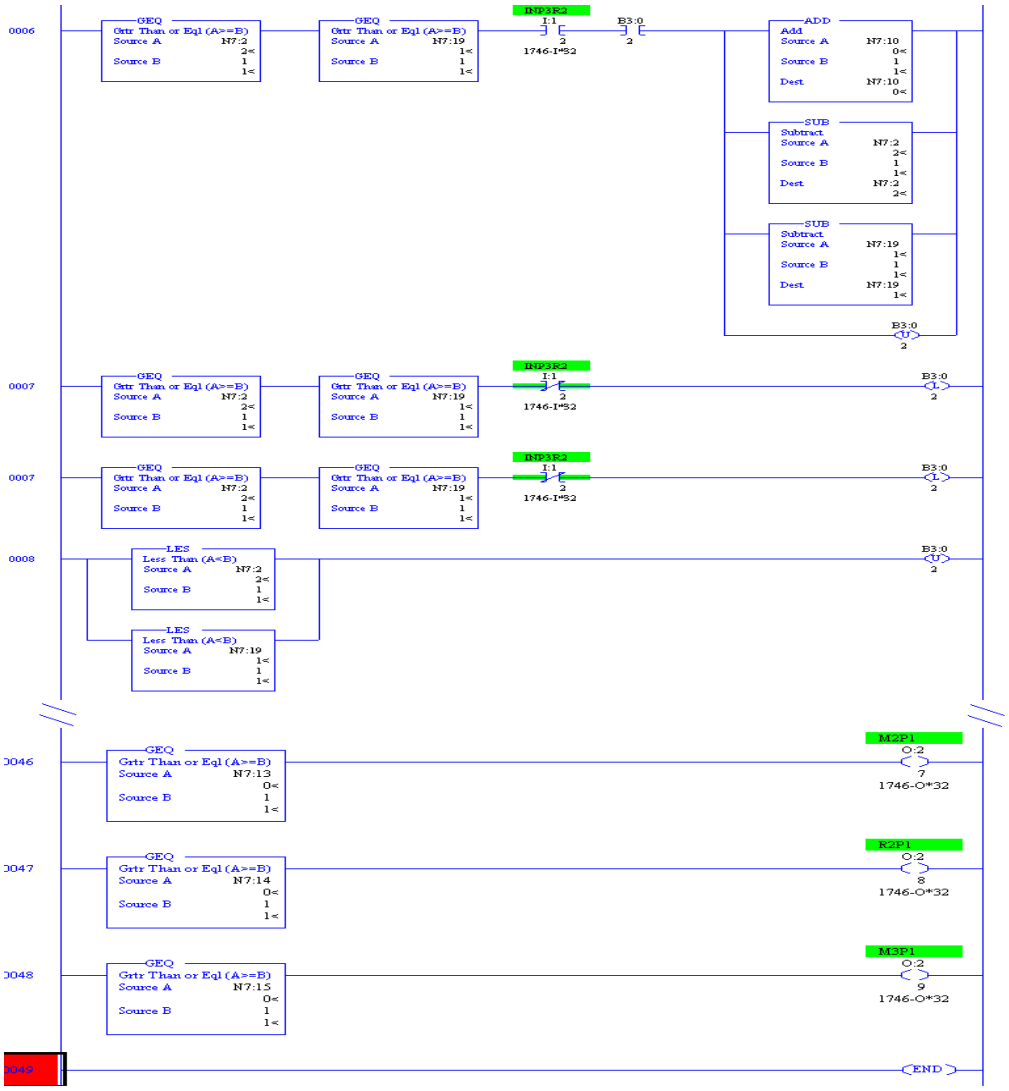


Figura 5.10 Fracción de código *ladder* FMS

5.3 Proceso de Disolución

La Figura 5.11 muestra un esquema de la planta SED del laboratorio de procesos donde se implementará el proceso de disolución, el cual se desarrolla mediante las siguientes fases:

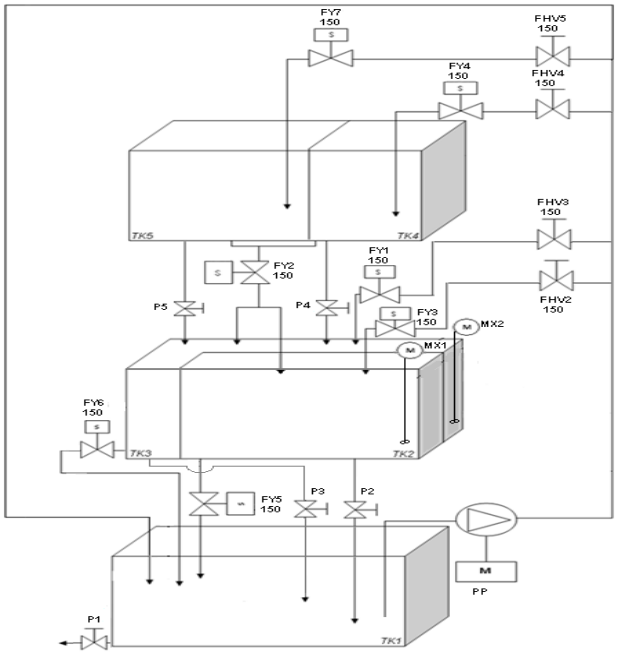


Figura 5.11 Diagrama de flujo proceso de disolución

1. Se deben llenar los tanques cuatro y cinco (TK4 y TK5) con el solvente por medio de la motobomba y las electroválvulas FY6 y FY7 respectivamente, cuando uno de los dos alcance su nivel máximo, estas se deben cerrar y la motobomba se debe apagar.
2. Abrir la electroválvula FY5 que descarga TK4 y TK5 y carga los tanques dos y tres (TK2 y TK3). Cinco segundos después de abierta FY5 la motobomba debe encenderse por tres segundos para verter el disolvente en TK2 y TK3 mediante las electroválvulas FY3 y FY4 respectivamente, transcurrido los tres segundos se debe apagar la motobomba y cerrar las electroválvulas FY3 y FY4.
3. Cuando se llene cualquiera de los tanques, TK2 o TK3, la electroválvula FY5 se debe cerrar y los mezcladores uno y dos (MX1 y MX2) deben agitar la mezcla de TK2 y TK3 por cinco segundos, transcurrido este tiempo MX1 y MX2 deben apagarse.
4. Cuando la mezcla de los tanques TK2 y TK3 esté lista, se debe drenar mediante la apertura de las electroválvulas FY2 y FY1 respectivamente. cuando la totalidad de la mezcla haya sido vaciada, la planta debe quedar lista para producir una nueva cantidad de mezcla.

Todos los tanques de la planta cuentan con sensores de efecto hall que permiten determinar cuándo se ha alcanzado un nivel mínimo o máximo.

Nota: El llenado de los tanques TK4 y TK5 solo se hace cuando hayan alcanzado un nivel mínimo. Dado que la planta cuenta con un tanque (TK1)

el cual sirve como proveedor y deposito del material que fluye en la planta, se supondrá que la cantidad de dicho material está garantizada para llevar a cabo el proceso, sin embargo, se debe evitar que la motobomba siga funcionando si el nivel en éste alcanza un mínimo (en el cual la motobomba no puede succionar). En este caso la motobomba se debe detener por seguridad.

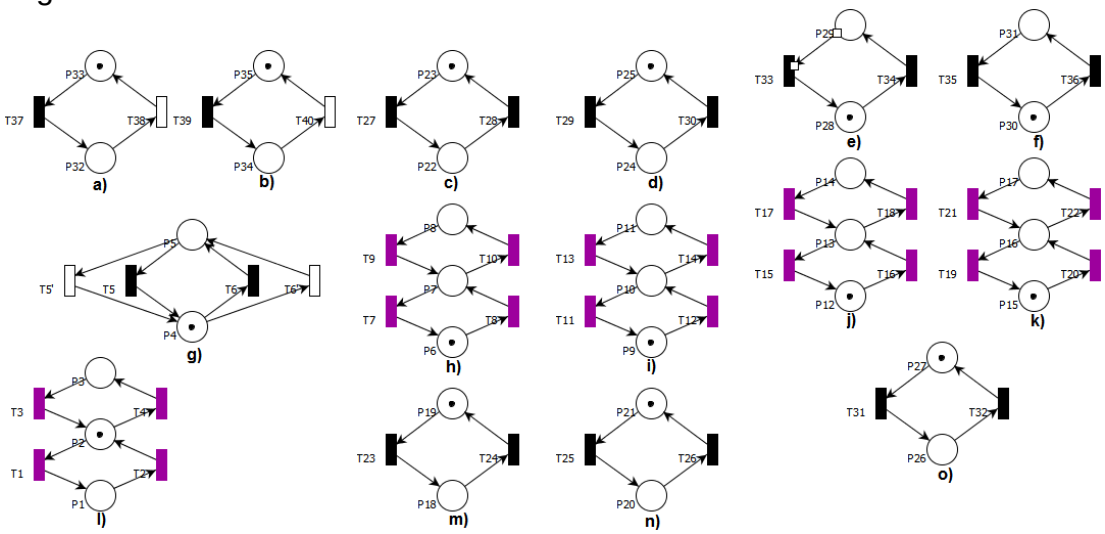


Figura 5.12 Modelo en PN de los componentes de la planta SED

a) Mezclador 1 (MX1)
b) Mezclador 2 (MX2)
c) Electroválvula de carga TK 2 (FY3)
d) Electroválvula de carga TK 3 (FY4)
e) Electroválvula de carga TK 4 (FY6)
f) Electroválvula de carga TK 5 (FY7)
g) Moto – Bomba
h) Tanque 2 (TK2)
i) Tanque 3 (TK3)
j) Tanque 4 (TK4)
k) Tanque 5 (TK5)
l) Tanque 1 (TK1)
m) Electroválvula descarga TK 2 (FY2)
n) Electroválvula descarga TK 3 (FY1)
o) Electroválvula descarga TK 4 y TK5 (FY5)

5.3.1 Restricciones físicas

Con el fin de diseñar un supervisor correcto para el proceso anteriormente descrito, es necesario imponer previamente restricciones físicas que eviten un comportamiento no deseado de la PN, el cual no es lógico que se presente en el sistema real. A continuación se describen dichas restricciones:

1. $-\mu_{22}-\mu_{24}-\mu_{26}-\mu_{28}-\mu_{30}+q_5+q_{5'}\leq 0$: La moto – bomba no se debe encender si las electroválvulas de llenado están cerradas.
2. $\mu_{19}+q_7+q_9\leq 1$: Si FY2 está cerrada, TK2 no puede vaciarse.
3. $\mu_{23}+\mu_{27}+q_8+q_{10}\leq 2$: Si FY3 y FY5 están cerradas, TK2 no puede llenarse.
4. $\mu_{21}+q_{11}+q_{13}\leq 1$: Si FY1 está cerrada, TK3 no puede vaciarse.
5. $\mu_{25}+\mu_{27}+q_{12}+q_{14}\leq 2$: Si FY4 y FY5 están cerradas, TK3 no puede llenarse.
6. $\mu_{29}+q_{16}+q_{18}\leq 1$: Si FY6 está cerrada, TK4 no puede llenarse.
7. $\mu_{31}+q_{20}+q_{22}\leq 1$: Si FY7 está cerrada, TK5 no puede llenarse.
8. $\mu_{27}+q_{15}+q_{19}\leq 1$: Si FY5 está cerrada, TK4 y TK5 no pueden vaciarse.

5.3.2 Restricciones lógicas o de control

9. $-\mu_1-\mu_{14}-\mu_{17}+q_6+q_{34}+q_{36}\leq 0$: Apagar la motobomba y cerrar las electroválvulas FY6 y FY7 si se llenó cualquiera de los tanques TK4 o TK5.
10. $-\mu_{12}+q_5+q_{33}\leq 0$: Abrir FY6 si TK4 alcanza el nivel mínimo.
11. $-\mu_{15}+q_5+q_{35}\leq 0$: Abrir FY7 si TK5 alcanza el nivel mínimo.
12. $-\mu_{12}-\mu_{15}+q_5\leq 0$: Encender moto – bomba si el nivel en TK4 o TK5 es mínimo.
13. $-\mu_{14}-\mu_{17}+q_{31}-v_{26}\leq 0$: Abrir FY5 si TK4 o TK5 se llenó o si se inició una nueva mezcla.
14. $v_{5'}-v_{31}\leq 0$: Arrancar moto – bomba cinco segundos después de abierta FY5.
15. $v_{27}+v_{29}-2v_{5'}\leq 0$: Abrir FY3 y FY4 cuando la moto – bomba haya encendido después de los cinco segundos de cierre de FY5.
16. $v_{28}+v_{30}-2v_{6'}\leq 0$: Cerrar FY3 y FY4 después que la moto – bomba haya encendido por tres segundos.
17. $-\mu_8-\mu_{11}+q_{32}+q_{37}+q_{39}\leq 0$: Cerrar FY5 y encender mezcladores 1 y 2 (MX1 y MX2) cuando se llene TK2 o TK3.
18. $v_{23}+v_{25}-v_{38}-v_{40}\leq 0$: Abrir FY1 y FY2 cuando MX1 y MX2 se hayan apagado.
19. $-\mu_6-\mu_9+2q_{24}+2q_{26}\leq 0$: Cerrar FY1 y FY2 cuando los tanques dos y tres se hayan vaciado.
20. $v_{31}-v_{26}\leq 1$: Arbitro para permitir la apertura de FY5 si se da inicio a otra mezcla.
21. $\mu_8+\mu_{11}+2q_{31}\leq 2$: impedir la apertura de FY5 si TK2 o TK3 están llenos.
22. $v_{37}-v_8\leq 0$: Arbitro para que en cada Batch MX1 solo mezcle una vez.
23. $v_{39}-v_{12}\leq 0$: Arbitro para que en cada Batch MX2 solo mezcle una vez.

24. $v_{6'} - v_{5'} \leq 0$: Arbitro para el apagado de la motobomba con temporizador.
25. $v_6 - v_5 \leq 1$: Arbitro para el apagado de la motobomba sin temporizador.
26. $\mu_{12} + \mu_{15} + q_{31} \leq 2$: Si TK4 y TK5 alcanzan el nivel mínimo, FY5 no debe abrirse si está cerrada.
27. $\mu_1 + q_5 + q_5{}' + q_{33} + q_{35} \leq 1$: Si TK1 alcanza el nivel mínimo, la moto – bomba no debe encenderse si está apagada, FY6 y FY7 no deben abrirse si están cerradas.
28. $\mu_{14} + q_5 + q_{35} \leq 1$: Si TK4 está lleno evitar que la moto – bomba y FY7 enciendan si TK5 está vacío.
29. $\mu_{17} + q_5 + q_{33} \leq 1$: Si TK5 está lleno evitar que la moto – bomba y FY6 enciendan si TK4 está vacío.

La PN del supervisor no es mostrada aquí por cuestiones de espacio.

5.3.3 Verificación del marcado de la PN en código *ladder*

Para verificar el marcado de la PN en el código *ladder* generado por CRP, se puede hacer mediante la comparación del marcado de la PN en PIPE, con la herramienta Incidence & Marking, y el estado de los enteros y bits del PLC que representan el marcado de los lugares de la PN mapeada a código *ladder*. Dado el tamaño y la cantidad de estados de la PN, la verificación del marcado solo se hace para dos estados de ésta.

Antes de iniciar la validación es conveniente definir las *tags* o en que puertos de entrada (I:?) y salida (O:?) del PLC están conectados los sensores y actuadores de la planta SED. Ver Tabla 5.2.

TAG	DESCRIPCIÓN	PUERTO IN/OUT
LLE TK1	Límite inferior tk1	I:0/1
LHE TK1	Límite superior tk1	I:0/2
LLE TK2	Límite inferior tk2	I:0/3
LHE TK2	Límite superior tk2	I:0/4
LLE TK3	Límite inferior tk3	I:0/5
LHE TK3	Límite superior tk3	I:0/6
LLE TK4	Límite inferior tk4	I:0/7
LHE TK4	Límite superior tk4	I:0/8
LHE TK5	Límite superior tk5	I:0/9
LLE TK5	Límite inferior tk5	I:0/10
FY1	Electroválvula de descarga tk 3	O:0/1
FY2	Electroválvula de descarga tk 2	O:0/2

FY3	Electroválvula de carga tk2	O:0/3
FY4	Electroválvula de carga tk3	O:0/4
FY5	Electroválvula descarga tk 4 y 5	O:0/5
FY6	Electroválvula carga tk4	O:0/6
FY7	Electroválvula carga tk5	O:0/7
MX1	Mezclador tk3	O:0/8
MX2	Mezclador tk2	O:0/9
-	Inicio remoto bomba (CM1)	O:0/10
-	Parada remota bomba (CM2)	O:0/11

Tabla 5.2 Tags planta SED

También es necesario definir qué bits (B3:?) y enteros (N7:?) en el PLC representan los lugares de la PN. Ver Tabla 5.3. Esta asignación es realizada automáticamente por CRP.

B3:0/0	P1	B3:1/5	P29	N7:5	PC7
B3:0/1	P10	B3:1/6	P3	N7:6	PC8
B3:0/2	P11	B3:1/7	P30	B3:2/5	PC9
B3:0/3	P12	B3:1/8	P31	B3:2/6	PC10
B3:0/4	P13	B3:1/9	P32	N7:7	PC11
B3:0/5	P14	B3:1/10	P33	N7:8	PC12
B3:0/6	P15	B3:1/11	P34	N7:9	PC13
B3:0/7	P16	B3:1/12	P35	N7:10	PC14
B3:0/8	P17	B3:1/13	P4	N7:11	PC15
B3:0/9	P18	B3:1/14	P5	N7:12	PC16
B3:0/10	P19	B3:1/15	P6	N7:13	PC17
B3:0/11	P2	B3:2/0	P7	N7:14	PC18
B3:0/12	P20	B3:2/1	P8	N7:15	PC19
B3:0/13	P21	B3:2/2	P9	N7:16	PC20
B3:0/14	P22	N7:0	PC0	N7:17	PC21
B3:0/15	P23	N7:1	PC1	N7:18	PC22
B3:1/0	P24	N7:2	PC2	N7:19	PC23
B3:1/1	P25	N7:3	PC3	N7:20	PC24
B3:1/2	P26	N7:4	PC4	N7:21	PC25
B3:1/3	P27	B3:2/3	PC5	B3:2/7	PC26
B3:1/4	P28	B3:2/4	PC6		

Tabla 5.3 Bits y enteros utilizados en el PLC.

Inicialmente los tanques TK2, TK3, TK4 y TK5 están vacíos. TK1 está lleno. La moto – bomba, FY6 y FY7 están operando. En la Figura 5.13 se observa

el estado de la planta, el marcado de la PN y el estado del puerto de entrada, puerto de salida, bits y enteros del PLC.

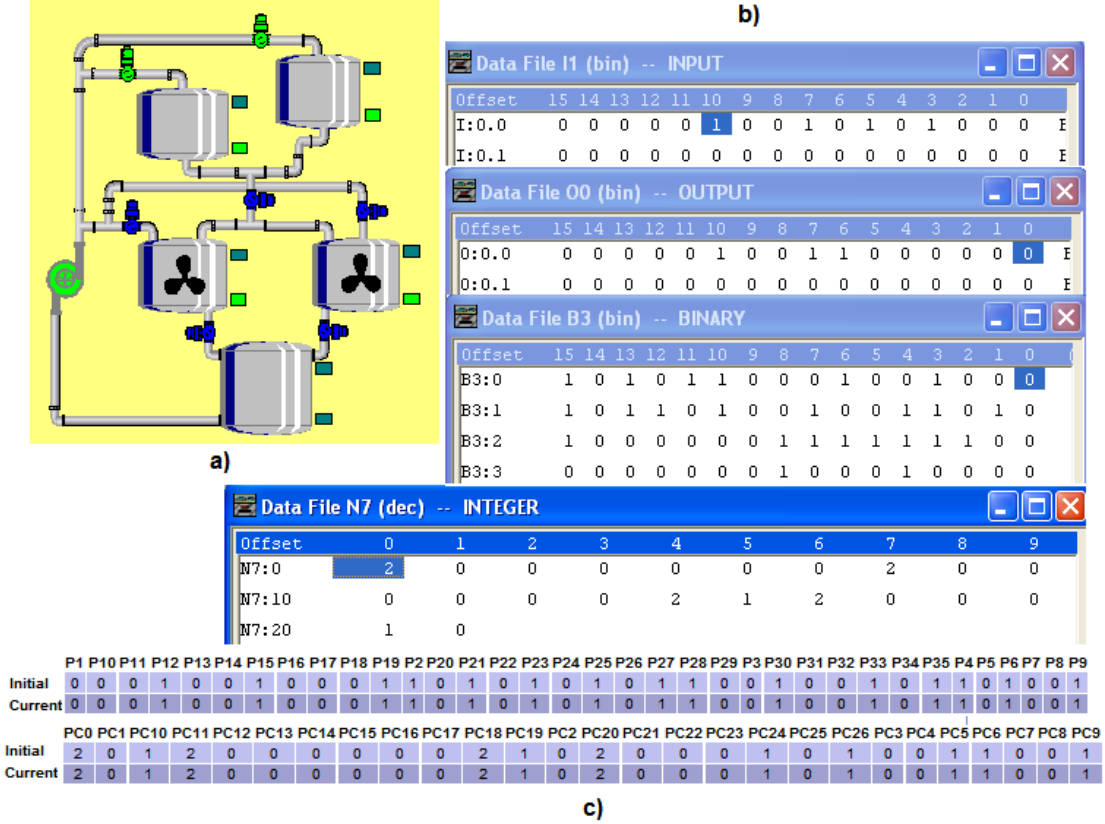


Figura 5.13 a) Estado inicial de la planta. b) Estado inicial del PLC c) Marcado inicial de la PN.

El segundo estado verificado es cuando TK2 se llena, todas las electroválvulas y la moto – bomba están apagadas y los dos mezcladores están encendidos. La Figura 5.14 muestra los estados de la PN, de la planta y el PLC.

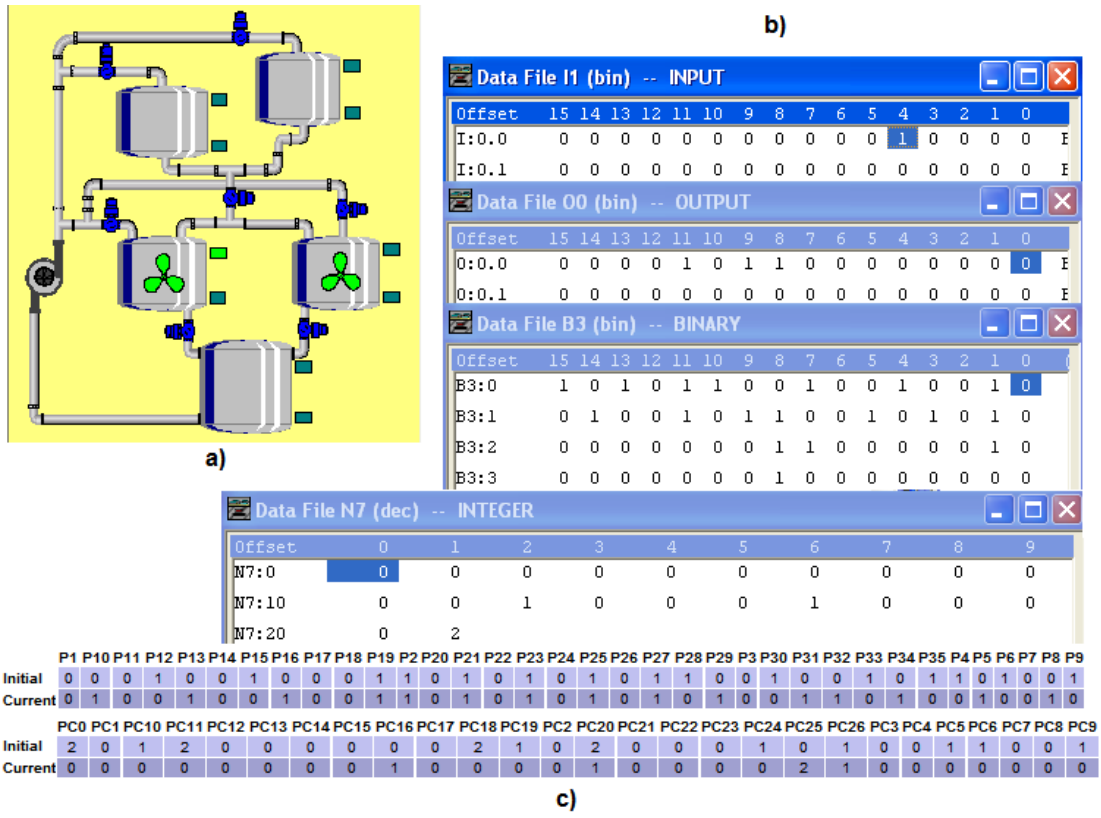


Figura 5.14 a) Estado intermedio de la planta. b) Estado intermedio del PLC. c) Estado intermedio de la PN

6 CONCLUSIONES, TRABAJOS FUTUROS Y LIMITACIONES

6.1 Conclusiones

Gracias a la representación gráfica de las PN's, el modelado de DEDS es mucho más práctico y sencillo que realizarlo con otros formalismos, pues el modelo del sistema en lazo abierto puede ser generado directamente a partir del conocimiento de los estados que puede alcanzar cada componente que afecta la evolución del sistema, sin la necesidad de realizar operaciones previas, permitiendo obtener un modelo más compacto y visualmente más comprensivo, sin dejar de lado el soporte matemático que las respalda y sus propiedades estructurales y dinámicas que ayudan a analizar el comportamiento de los DEDS que representan.

Para la imposición de restricciones físicas y/o de control, también llamadas lógicas en este trabajo, la metodología adoptada permite que estas sean fáciles de percibir puesto que la PN del proceso permite evidenciar con facilidad qué estados (marcados) violan los requerimientos de control y, por lo tanto, se debe impedir que estos sean alcanzados. Además, tanto el formalismo como la metodología adoptadas en este proyecto, para el diseño del supervisor, permiten tener en cuenta aspectos importantes como la naturaleza del sistema (concurrencia, sincronización, no deterministas), eventos no controlables y no observables y problemas de bloqueo.

Por otro lado, a nivel industrial es típico encontrar que ciertas tareas se lleven a cabo por un determinado tiempo, dicho requerimiento se puede satisfacer por medio de las PN's temporizadas, mediante la temporización de las transiciones, con las cuales la metodología adoptada es compatible. Otro requerimiento que es muy usual encontrar en el control de DEDS es el conteo de alguna variable del proceso, ello está contemplado en el método adoptado para el diseño del DEC.

Para el modelado de aplicaciones de primer nivel, donde el supervisor debe organizar como deben interactuar los diferentes componentes del sistema de acuerdo a la ocurrencia de eventos o estado del mismo, en este proyecto se propone modelar cada componente físico de la planta por separado, en lugar de orientar el modelo hacia la secuencia que deba seguir esta, similar al enfoque con autómatas, para posteriormente imponer restricciones físicas y/o lógicas sobre el comportamiento de la misma. Cuando se desee modelar celdas de manufactura, generalmente el modelo del sistema se hace teniendo en cuenta la distribución de la celda o la manera en que fluye el

producto que se fabrica en esta, para posteriormente imponer las restricciones físicas y/o lógicas sobre la misma.

Obedeciendo las anteriores reglas para el modelado de DEDS es posible facilitar la compresión del modelo aun por personas que no estén relacionadas con el tema de supervisores pero que de alguna manera tienen que ver con el proceso que se pretende controlar.

La herramienta resultado de este proyecto permite la construcción sistemática e implementación en PLC del supervisor para DEDS, ya que ésta sirve de puente entre la herramienta PIPE, donde se modela el sistema en lazo abierto, y RsLogix donde se implementa el código *ladder* para su posterior descarga al PLC.

Además, la herramienta desarrollada, bajo la metodología formal expuesta, permite imponer las restricciones sobre el modelo de la planta en lazo abierto de manera sencilla, teniendo en cuenta la naturaleza de los eventos, sin que el usuario se preocupe si estas serán admisibles o no, pues la herramienta permite solucionar las que sean inadmisibles de acuerdo a los métodos vistos en la sección 2.4.3, luego el usuario podrá ver la PN del sistema controlado en PIPE, la cual contendrá los lugares de control conectados automáticamente a la PN de la planta, donde se podrá comprobar mediante simulación el comportamiento del supervisor y observar si este obedece los requerimientos de control o está libre de bloqueos. El proceso de simulación, además de permitir comprobar el comportamiento deseado del supervisor, permite establecer consideraciones adicionales que quizá deban ser tenidas en cuenta para el control de la planta y que no fueron posibles de vislumbrar en primera instancia, permitiendo adicionar nuevas restricciones que contemplen dichas consideraciones, ya que cuando el DEC se diseña directamente en código *ladder*, éstas consideraciones no son fáciles de observar lo que significa una ventaja para la metodología y el formalismo adoptados en este proyecto. Cuando exista bloqueo en la PN es posible encontrar los sifones mínimos e imponer las restricciones necesarias para controlarlos, sin embargo, en ocasiones el tamaño o la topología de la PN hace que el número de sifones aumente, lo que dificulta encontrarlos y determinar cuales se deben controlar. Una vez el supervisor cumpla con todos los requerimientos, es posible asociar a cada lugar y transición las acciones o salidas y eventos o entradas respectivos como se ha indicado en el capítulo 4, lo que permite simular el supervisor mediante la herramienta CRP y verificar por última vez el desempeño de éste de acuerdo al orden lógico en que puedan ocurrir los eventos en el sistema real, permitiendo observar el comportamiento dinámico del DEC. Finalmente la herramienta permite mapear la PN del supervisor a *ladder* mediante la metodología adoptada y posteriormente modificada vista en el capítulo 3. De esta manera

la herramienta permite el diseño sistemático y la posterior implementación del supervisor en PLC.

Todo lo anterior permite el diseño de supervisores de manera formal y sistemática lo que disminuye el tiempo de diseño del mismo, evita soluciones que dependen de estilos personales de programación, garantiza que el supervisor generado esté libre de errores, lo que lo hace más confiable, facilita el mantenimiento o cambio del algoritmo de control pues, el programador ya no deberá modificar directamente el código *ladder* sino la PN del supervisor mediante la imposición de restricciones que permitan al DEC ajustarse a la nueva política de control, aun cuando la actual solución no haya surgido de éste.

El potencial del formalismo de las PN's, de la metodología adoptada para el diseño del supervisor y del método para el mapeo de la PN a *ladder* convergen en la herramienta desarrollada en este proyecto, la cual está desarrollada en lenguaje de programación C++ lo que la hace de libre acceso al igual que la herramienta PIPE que ha sido diseñada en Java, además, esta permite vincular una herramienta para el modelado de DEDS mediante PN's y de libre acceso, con otra comercial muy común en la industria para la programación de PLC's. En la sección de casos de estudio y a lo largo del desarrollo de este proyecto se presentaron varios casos teóricos en donde se puede verificar la metodología seleccionada, además se utilizó la planta SED del laboratorio de procesos para la verificación de la herramienta en un caso real.

6.2 Trabajos Futuros

La herramienta desarrollada permite el diseño de DEC's bajo una metodología formal, dicha metodología puede brindar también soluciones a problemas de control hibrido lo que abre las puertas para que el programa de ingeniería en automática industrial incursione en esta disciplina y se desarrollen trabajos en dicho campo.

La herramienta solo permite la generación automática de código *ladder* para PLC's que son programables mediante RSlogix, pero en la industria es muy común encontrar diferentes tipos de PLC's los cuales podrían ser programados a partir de PN, esto permitiría estandarizar, en gran medida, la forma en que estos se programan para problemas que involucran DEDS.

La metodología seleccionada como ya se vio puede brindar soluciones tanto de primer nivel como de un nivel superior pero a medida que aumenta la complejidad de los sistemas en el nivel superior, se hace más difícil

modelarlos mediante PN's generales, en la literatura este tipo de problemas se soluciona mediante las PN's coloreadas lo que imposibilita la metodología adoptada y por ende la herramienta desarrollada para el diseño de supervisores en este campo, sin embargo, hay trabajos que extienden la metodología adoptada en este trabajo para diseñar supervisores en ese caso.

El formalismo acogido para el modelado y control de DEDS tiene bondades graficas que lo hacen atractivos para pensar en la implementación de SCADAS con PN's.

6.3 Limitaciones

Uno de los objetivos del control supervisor para DEDS es prevenir que el sistema a controlar llegue al bloqueo, en la literatura se han propuesto distintos métodos para encontrar los sifones y trampas, que son los causantes del bloqueo de la PN, pero estos métodos se hacen ineficientes conforme la PN aumenta su complejidad (aumento de estados) ya que se hace más difícil encontrar los sifones y trampas mínimas que contenga la PN.

Pero no basta con encontrar los sifones mínimos de la PN, además, se deben controlar. No obstante, cuando se busca controlarlos es probable que surjan nuevas situaciones de bloqueo, pues el control de estos se hace añadiendo nuevas restricciones al sistema lo que implica un nuevo lugar de control, y dicho lugar puede generar nuevos sifones en la PN. Por esta razón, tratar de prevenir que la PN del DEC pierda su vivacidad, se puede convertir en una tarea complicada para el diseñador del DEC, más aún, si el sistema es lo bastante complejo, sin embargo, en varios casos dicha situación de bloqueo se puede evitar con la imposición de restricciones idóneas.

La metodología de generación de código *ladder* no se puede considerar óptima, al considerar una gran variedad de posibilidades para la PN, no realiza un uso muy adecuado de los recursos del PLC. Proponer nuevas metodologías más eficientes, que mediante un estudio de la red de Petri disminuyan la cantidad de recursos necesarios para implementarlas en el PLC, es necesario antes de considerar este método de diseño de supervisores factible para un uso masivo.

7 BIBLIOGRAFÍA

[1] P. J. G. Ramadge, W.M. Wonham. “The Control of Discrete Event Systems”. Proceedings of the IEEE, Vol. 77, No. 1. 1989.

[2] Giua A. “Petri Nets as Discrete Event Models for Supervisory Control”. Rensselaer Polytechnic Institute Troy, New York. July, 1992.

[3] Murat Uzam. “Petri-Net-Based Supervisory Control of Discrete Event Systems And Their Ladder Logic Diagram Implementation”. Telford Research Institute, Research and Graduate College. University of Salford. Reino Unido. 1998.

[4] Zhou M.C., DiCesare F. “Petri Net Synthesis for Discrete Event Control of Manufacturing Systems”. Boston, MA, Kluwer Academic Publishers, 234 Págs. 1993.

[5] Desrochers A.A., Al-Jaar R.Y. “Applications of Petri Nets in Manufacturing Systems, Modelling, Control and Performance Analysis”. IEEE Press, New York, ISBN: 0-87942-295-5, 345 pages.1995.

[6] Giua A. “Petri Net Techniques for Supervisory Control of Discrete Event Systems”. Proceedings of 1st International. Workshop on Manufacturing and Petri Nets. Osaka Japan. June, pp. 1 – 21. 1996.

[7] O.R. Boissel, J.C. Kantor. “Optimal Feedback Control Desing for Discrete Event System Using Simulated Annealing”. Department of Chemical Engineering. University of Notre Dame. Notre Dame, Estados Unidos. 1993.

[8] Luis Diego Murillo Soto. “Redes de Petri: Modelado e implementación de algoritmos para autómatas programables”. Tecnología en Marcha. Vol. 21, N° 4, Págs.102-125. 2008.

[9] T. Murata. “Petri nets: Properties, Analysis and application”. Proceedings of the IEEE, Vol. 77, No 4, pp. 541-580. 1989.

[10] J. Moody. “Petri net supervisors for discrete events systems”. Universidad de Notre Dame. Indiana. Estados Unidos. 1998.

[11] K. Lautenbach, H. Ridder. “The Linear Algebra of deadlock avoidance a Petri net aproach”.Universidad de Koblenz –Landau. Instituto de tecnología de software Rheinau 1. 1996.

[12] K. Barkaoui, J. F. Pradat-Peyre. “On Liveness and Controlled Siphons in Petri Nets ”, Conservatoire National des Arts e Métiers. Laboratorio CEDRIC. París, Francia, 1998.

[13] Z. Li, M. Zhou. “Elementary Siphons in Petri nets and deadlock control”. Escuela de ingeniería electro-mecánica, Universidad Xidiana, China. Departamento de electricidad e ingeniería computacional, Instituto de tecnología de New Jersey, Estados Unidos.2004.

[14] L. Zi-Whu, W.An-Rong. “A Petri Net Deadlock Based Prevention Approach for Flexible Manufacturing Systems”. Escuela de ingeniería electro-mecánica, Universidad Xidiana, China. 2003.

[15] M. Yamauchi. “Algorithms for Extracting Minimal Siphons Containing Specified Places in a General Petri Net”. IEICE Trans, Fundamentals, Vol. E82-A No 11, Noviembre 1999.

[16] D. Y. Chao. “Computation of Elementary Siphons in Petri nets for Deadlock Control”. Departamento de Administración y Ciencia de la Información, Universidad Nacional Cheng Chi. Taipei, Taiwan. 2006.

[17] J. Ezpeleta, J. M. Couvreur, M. Silva. “A new technique for finding a generating family of siphons, traps and ST-components applications to colored Petri nets”. Avances en redes de Petri, G. Rozemberg, Vol. 614 de Notas de lectura en ciencia computacional, páginas 126-147. Springer-Verlag, Berlin, Nueva York. 1993.

[18] A. Węgryzn, A. Karatevich, J. Bieganowsky. “Detections of Deadlocks and Traps in Petri Nets by Means of Thelen’s Prime Implicant Method”. Instituto de Ingeniería Computacional y Electronica, Universidad de Zieloena Góra. Zieloena Góra, Polonia. 2004.

[19] R. Cordone, L. Ferrarini, L. Piroddi. “Some results on the computation of minimal siphons in petri nets”. Dipartimento di Tecnologie dell’Informazione, Dipartimento di Elettronica e Informazione, Politecnico di Milano. Proccedings of the 42th IEEE conference on decision control. Maui, Hawái, Estados Unidos. Diciembre 2003.

[20] B. Thelen. “Investigations of algorithms for computer aided logic design of digital circuits”. Universidad de Karl-Shue.Alemania.1988.

[21] J. Bieganowsky, A. Karatevich. “Heuristics for Thelen’s Prime Implicant Method”. Instituto de Ingeniería Computacional y Electronica, Zieloena Góra. Schada e Informatica, Volumen 14. 2005.

[22] M. V. Iordache, J. O. Moody, P. J. Antsalkis. “A Method for Deadlock Prevention in Discrete Event Systems Using Petri Nets”. Grupo ISIS, Universidad de Notre Dame. 1999.

[23] M. V. Iordache, J. O. Moody, P. J. Antsalkis. “Automated Synthesis of Liveness Enforcing Supervisors Using Petri Nets”.Grupo ISIS, Universidad de Notre Dame. 2000.

[24] Ramadge P.J. and Wonham W.M. "Modular Feedback Logic for Discrete Event Systems", SIAM Journal on Control and Optimization, Vol. 25, No. 5, Sept., pp.1202 – 1218. 1987.

[25] Ramadge P.J. and Wonham W.M. "Supervisory Control of a Class of Discrete Event Processes". SIAM Journal on Control and Optimization, Vol. 25, No. 1, January, pp. 206-230. 1987.

[26] Murat Uzam and Anthony H. Jones. “A New Petri-Net-Based Synthesis Technique for Supervisory Control of Discrete Event Systems”. Turk J Elec. Engin, VOL.10, N°.1 .2002.

[27] Giua A., DiCesare F., Silva M. "Petri Net Supervisors for Generalized Mutual Exclusion Constraints". Proc. of 12th IFAC World Congress (Sidney, Australia).July.pp. I: 267-270. 1993.

[28] A. Giua, F. DiCesare, M. Silva. “Generalized Mutual Exclusion Constraints on Nets with Uncontrollable Transitions”. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 974-979. Chicago. 1992.

[29] K. Yamalidou, J. O. Moody, M. Lemmon and P.J. Antsaklis. “Feedback Control of Petri Nets Based on Place Invariants”. Proceedings of the IEEE Conference on Decision and Control. Vol. 3, pp. 3104 – 3109. Lake Buena Vista, FL, Diciembre.1994.

[30] M.V. Iordache. “Methods for the Supervisory Control of Concurrent Systems Based on Petri Net Abstractions”. Universidad de Notre Dame. Indiana. Estados Unidos. 2003.

[31] John O. Moody and Panos J. Antsaklis. “Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions”. Universidad de Notre Dame. Febrero, 1999.

[32] M. V. Iordache, P. J. Antsalkis. “Supervision Based on Place Invariants: A Survey”. Discrete Event Dyn Syst, Vol. 16, Springer Science + Bussiness Media, páginas 451-492.2006.

[33] F. Basile, P. Chiacchio, A. Giua. “On the Choice of Suboptimal Monitors for Supervisory Control of Petri Nets”. Dip. di Informatica e Sistemistica, Università de gli Studi di Napoli Federico II, Italia. Dip. di Ingegneria Elettrica ed Elettronica, Università di Cagliari, Italia.1998.

[34] E. A. Lima, C. E. T. D´orea. “An Algorithm for Supervisory Control of Discrete Event Systems Via Place Invariants”. Universidade Federal do Bahia, Escola Politécnica. Universidade do Estado da Bahia. Brazil. 2002.

[35] M. V. Iordache, P.J. Antsalkis. “Synthesis of Supervisors Enforcing General Linear Vector Constraints in Petri Nets”. The 2002 American Control Conference, Mayo 8 – 10, Anchorage, Alaska. pp. 154 – 159. 2002.

[36] Thomas Mertke, Georg Frey. “Formal Verification of PLC – Programs Generated from Signal Interpreted Petri Nets”. Proceedings of the. IEEE Systems, Man and Cybernetic conference. 2001.

[37] Marian V. Iordache, Panos J. Antsaklis. “Admissible Decentralized Control of Petri Nets” Proceedings of the American Control Conference. Denver, Colorado June 4-6, 2003.

[38] Manual de la aplicación Petri-LLD disponible en: http://sourceforge.net/apps/mediawiki/petrilld/index.php?title=Main_Page. 2011.

[39] G. Frey, M. Minas. “Editing, Visualizing and Implementing Signal Interpreted Petri Nets”. Lehrstuhl für Automatisierungstechnik, Universität Kaiserslautern. Lehrstuhl für Programmiersprachen, Universität Erlangen-Nürnberg.Alemania.

[40] G. Frey, M. Minas. “Visual PLC-Programming using Signal Interpreted Petri Nets”. Lehrstuhl für Automatisierungstechnik, Universität Kaiserslautern. Lehrstuhl für Programmiersprachen, Universität Erlangen-Nürnberg.Alemania.

[41] M. Uzam, A. H. Jones, N. Ajlouni. “Conversion of Petri Nets Controllers for Manufacturing Systems into Ladder Logic Diagrams”. Instituto de investigación para diseño, manufactura y marketing. Universidad de Salford.1996.

[42] J. Ezpeleta, F. García-Valles, and J.M. Colom. “A Class of Well Structured Petri Nets for Flexible Manufacturing Systems”. Departamento de Informática e Ingeniera de Sistemas, Centro Politécnico Superior de Ingenieros, María de Luna 3. Zaragoza, España. 1998.