Generación de diagramas ladder mediante el uso de redes de Petri difusas

Ladder diagram generation using Fuzzy Petri-Nets

Humberto Gutiérrez R.*

Ángel G. Muñoz**

Fecha de recepción: 30 de mayo de 2013
Fecha de aceptación: 15 de junio de 2013

Resumen

En este documento se describe el proceso metodológico para generar diagramas ladder a partir de redes de Petri difusas. El método, basado en los conceptos del "análisis de la evolución de marcas" [3], ha sido desarrollado con el propósito de poder implementar sistemas de control complejos, expresados mediante redes de Petri difusas, en controladores lógicos programables (*PLC*).

La metodología es implementada en software utilizando como plataforma Matlab®, obteniendo así una herramienta en forma de "caja de herramientas (*toolbox*)", que permite además de simular las redes de Petri difusas, generar el diagrama ladder asociado a dicha red.

Palabras clave: Red de Petri difusas, diagrama ladder, análisis de evolución de marcas, Matlab, caja de herramientas.

* Ingeniero Electrónico, M. Sc., Docente de la Universidad Distrital Francisco José de Caldas Bogotá, Colombia. Correos electrónicos: hgutierrez@udistrital.edu.co - hgutierrez95@yahoo.com

** Ingeniero Electrónico. Estudiante de la Universidad Distrital Francisco José de Caldas Bogotá, Colombia. Correo electrónicos: agmunozp@correo.udistrital.edu.co angamupa@gmail.com

Abstract

In this document a methodological process is proposed, aimed to generate ladder diagrams from fuzzy Petri nets. The method is based on the "marks evolution analysis" concepts, which have been developed with the desire of implementing complex control systems expressed through fuzzy Petri nets, in programmable logical controllers (*PLC*).

Furthermore the method is implemented using Matlab® as platform, getting a toolbox which allows: simulate the fuzzy Petri nets and generate the ladder diagram associated to that net.

Key words: Fuzzy Petri Nets, ladder diagram, marks evolution analysis, Matlab, toolbox.

1. Introducción

En la actualidad muchos procesos industriales son automatizados utilizando controladores lógicos programables PLC (por sus siglas en inglés Programmable Logic Controller) debido a la versatilidad que ellos ofrecen. La norma IEC 61131-3 [5] establece los diferentes lenguajes de programación usados con PLCs, de los cuales sobresale el lenguaje de diagramas ladder LD (siglas de Ladder Diagram) por su simplicidad y similitud a los diagramas de circuitos eléctricos de contactos.

Sin embargo, la simplicidad y facilidad del manejo de diagramas ladder en el diseño de sistemas simples, no aparece en el momento de diseñar sistemas con ciertos eventos específicos tales como la concurrencia, el sincronismo y la secuencialidad. Aunque no es imposible implementar dichos eventos en diagramas ladder, resulta sumamente complicado para el diseñador concebir y ajustar un diseño de forma ágil y sencilla, lo cual se busca en la actualidad con el fin de reducir recursos, especialmente tiempo [2].

Las redes de Petri se han utilizado para el modelamiento de sistemas de eventos discretos, debido a su formalismo matemático y representación gráfica y que por su propia definición, ofrecen una solución sencilla en la descripción de sistemas con eventos de concurrencia, secuencialidad y sincronismo [7]; de esta manera las redes de Petri proveen una solución a la debilidad de los diagramas ladder.

Las redes de Petri se pueden clasificar en dos grupos: redes de Petri de bajo nivel y redes de Petri de alto nivel. Dentro del grupo de las redes de Petri de alto nivel se encuentran las redes de Petri difusas (FPN por sus siglas en inglés Fuzzy Petri-Nets), las cuales incorporan los elementos de la lógica difusa, brindando mayores posibilidades de diseño enfocadas especialmente a sistemas de control.

Con el propósito de poder implementar sistemas complejos de control (fácilmente ca-

racterizables por FPN) en los PLC, se ha desarrollado una metodología que permite llevar una FPN a un LD, basada en el método del análisis de evolución de marcas que ha sido desarrollado para redes de Petri de bajo nivel.

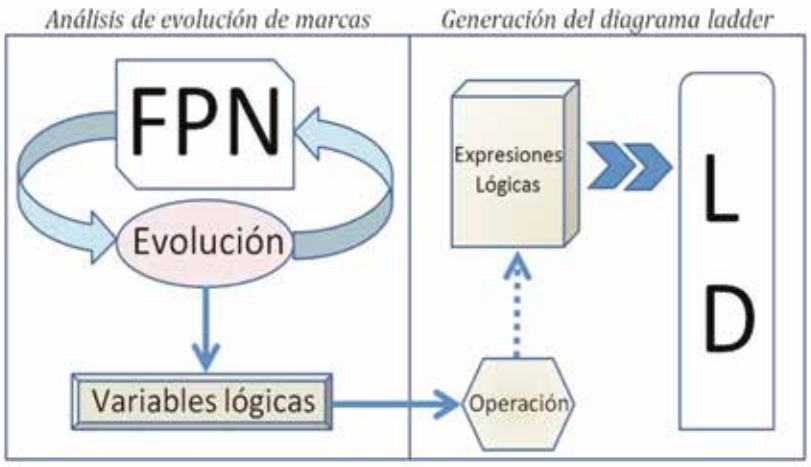
Este documento presenta un resumen del proyecto con los siguientes temas: en la sección 2 se expone la metodología que se ha desarrollado para generar diagramas ladder a partir de redes de Petri difusas, la sección 3 presenta algunos aspectos del diseño de la caja de herramientas que permite simular redes de Petri difusas además de generar diagramas ladder con FPN, finalmente en la sección 4 se tienen las conclusiones.

2. Metodología propuesta

Para realizar el paso de una red de Petri difusa (FPN) a un diagrama ladder (LD), se parte del hecho que se tiene una FPN funcional que representa correctamente el sistema deseado y que es capaz de evolucionar por lo menos una vez, es decir, que en el estado inicial aunque sea una transición se encuentra habilitada.

El método se basa en el análisis de la evolución de las marcas, el cual consiste en observar la FPN antes y después de cada evolución, para poder así comparar y obtener unas variables lógicas, en este caso de carácter difuso, que pueden ser operadas y, por tanto, formar expresiones, las cuales representan un LD. En la figura 1 se ilustra el proceso descrito.

Figura 1. Diagrama del proceso completo



Fuente: elaboración propia.

A continuación se desarrollará la metodología.

2.1. Redes de Petri difusas (FPN)

En esta sección se presentará el concepto de red de Petri difusa, tanto de forma matemática como gráfica con el objetivo de aclarar el concepto y establecer la notación que se usará en adelante. En [6, 10, 9, 1] se puede encontrar mayor información referente a las FPN.

Definición 1: una FPN se define matemáticamente como la séptupla de la forma:

$$N=(P,T,I,O,\alpha,\beta,M_o)$$

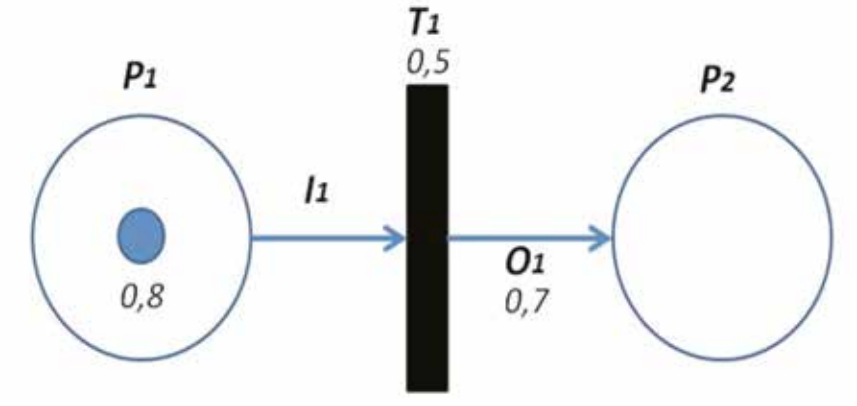
Donde son "lugares", son "transiciones", son arcos de entrada, son "arcos de salida", es una "función de asociación de activación para las transiciones", es una "función de asociación para los arcos de salida" y es el "estado inicial de la marcas de la red".

Gráficamente los lugares se representan con círculos, las transiciones con rectángulos, los arcos con flechas: si son arcos de entrada van desde un lugar hacia una transición, si son arcos de salida van desde una transición hacia un lugar; los valores de las funciones de asignación como de las marcas se colocan en el lugar correspondiente al que están asignados. En la figura 2 se muestra un ejemplo de una FPN básica representada gráficamente.

Se dice que una transición se encuentra habilitada si cada uno de los valores de las marcas asignadas a los lugares que entran a dicha transición tienen un valor igual o superior a la función asignada a la transición en cuestión. Si una transición se encuentra habilitada, esta puede ser "disparada", evolucionando así la red.

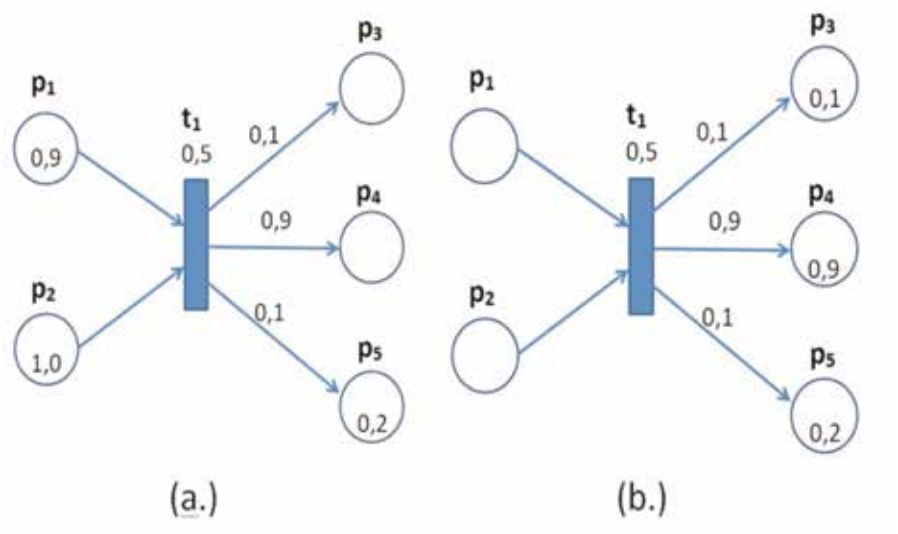
La evolución de una red de Petri se refiere a la actualización de los valores de las marcas, tanto de entrada como de salida de una transición que ha sido disparada. Los valores de las marcas se actualizan teniendo en cuenta la siguiente función:

Figura 2. Representación gráfica de una FPN básica



Fuente: elaboración propia.

Figura 3. Evolución de una FPN (a.) antes y (b.) después de disparada la transición



Fuente: elaboración propia.

$$M'(p_k) = \begin{cases} M(p_k) \vee \left(\mu_{M,t_j} \wedge \beta(t_j, p_k)\right), & \text{si } p_k \notin I(t_j) \\ \mu_{M,t_j} \wedge \beta(t_j, p_k), & \text{si } p_k \in I(t_j) \cap O(t_j) \\ 0, & \text{si } p_k \in I(t_j) \backslash O(t_j) \end{cases}$$

En donde las operaciones $\wedge$ son de tipo difusas y representan la T-norma y T-conorma respectivamente.

En la figura 3 se muestra una FPN antes y después de disparada una transición (transición t_1).

2.2. Análisis de evolución de marcas

Se pueden obtener unas variables lógicas que representan el funcionamiento de una FPN si se analiza su evolución. Esto es posible porque las marcas de la FPN reflejan el estado del sistema; si estas marcas permanecen invariables el estado del sistema, por tanto, también permanecerá invariable. En consecuencia, la evolución de las marcas de una FPN puede ser usada para reflejar el proceso de control del sistema real que está caracterizado por la FPN [3].

Esencialmente se necesitan 4 variables para representar el proceso de una FPN. Estas variables son: P para representar los lugares, T para representar las transiciones, C para representar las condiciones y E para representar los eventos. Como se está analizando la evolución de la red y ésta es dinámica, las variables dependerán de otra variable, llamada "evolución" y representada con el símbolo τ. La variable τ se puede relacionar con el tiempo, pero es llamada "evolución" debido a que ésta incrementa una unidad cada vez que una transición es disparada, es decir, que la red evoluciona.

Por otra parte se denota con el símbolo ψ a una variable de tipo difuso, es decir que pue-

de tomar cualquier valor que se encuentre únicamente en el intervalo (0,1].

Entonces, la asignación de valores a las variables establecidas se realiza de la siguiente manera [3]:

$$P_i(\tau) = \begin{cases} \psi, & \text{si } p_i \text{está marcada con } \psi \text{ en la evolución } \tau \\ 0, & \text{si } p_i \text{no tiene marcas en la evolución } \tau \end{cases}$$

$$T_j(\tau) = \begin{cases} 1, & \text{si } t_j \text{es disparada en la evolución } \tau \\ 0, & \text{si } t_j \text{no es disparada en la evolución } \tau \end{cases}$$

$$C_j(\tau) = \begin{cases} 1, & \text{si } C_j \text{es verdadera en la evolución } \tau \\ 0, & \text{en caso contrario} \end{cases}$$

$$E_j(\tau) = \begin{cases} 1, & \text{si } E_j \text{ocurre en la evolución } \tau \\ 0, & \text{en caso contrario} \end{cases}$$

Como se puede observar, las variables T_j, C_j y E_j, y hacen referencia a la transición *j* y son de tipo booleano, ya que simplemente establecen: si hubo disparo, se cumple la condición de disparo y ocurrió el evento (es decir, la transición fue seleccionada entre todas las transiciones en estado "habilitado" en el tiempo τ). Ninguna de ellas hace referencia al valor de disparo asignado a la transición y que en la red se distingue como α_j. El valor de α_j, únicamente determina si la transición está habilitada y puede ser disparada, por lo que si se dispara, la variable T_j tendrá el valor booleano de 1.

La variable P_i es de tipo difuso, pero ello no le impide poder ser operada con las demás variables. Aunque ellas son booleanas también se pueden considerar difusas, porque los valores '0' y '1' hacen parte del intervalo (0,1].

2.3. Generación del diagrama ladder (LD)

Una vez que se ha realizado el análisis de evolución de marcas y se tienen las variables que describen el proceso de la red, se procede a generar el LD. La etapa de generación del diagrama LD se puede dividir en dos fases: la síntesis de las expresiones y la interpretación de las expresiones.

2.3.1. Síntesis de las expresiones

El proceso de síntesis consiste en obtener las expresiones lógicas a partir de las variables que se obtuvieron. Para deducir las expresiones se aplica la siguiente fórmula:

$$P_i = \left(P_i + \sum_{t_j | t_j \in {}^{\circ}p_i} \left(\prod_{p_i | p_i \in {}^{\circ}t_j} P_i \cdot C_j \cdot E_j\right)\right) \cdot \prod_{t_j | t_j \in {}^{\circ}p_i} \left(\overline{\prod_{p_i | p_i \in {}^{\circ}t_j} P_k \cdot C_j \cdot E_j}\right)$$

La deducción de la ecuación y su explicación se encuentra en el Apéndice A.

2.3.2. Interpretación de las expresiones

En este punto es preciso aclarar que debido a que el lenguaje objetivo es LD, el cual representa un diagrama de contactos y, por tanto, comprende solo dos posibles estados ("encendido" y "apagado"), se debe llevar la expresión a una forma booleana equivalente.

La propuesta consiste en agregar una variable φ, cuyo valor se encuentra en el intervalo (0,1] y actúa como un umbral, el cual lo determina el diseñador de acuerdo a la sensibilidad deseada para el sistema de control. Entonces la nueva variable de tipo booleana para los lugares es:

$$P'_i(\tau) = \begin{cases} 1, & \text{si } P_i(\tau) \geq \varphi \\ 0, & \text{si } P_i(\tau) < \varphi \end{cases}$$

de forma que la fórmula para obtener las expresiones booleanas puede ser utilizada correctamente.

Es posible observar que para cada uno de los lugares de la FPN existirá una expresión lógica. La construcción del LD consiste entonces en tres pasos:

1. La parte izquierda de la expresión booleana pasa al LD como una salida.
2. Todos los elementos que se encuentran al lado derecho de la expresión pasan al LD como entradas.
3. Una operación + se expresa en el LD de forma paralela y una operación de forma serie.

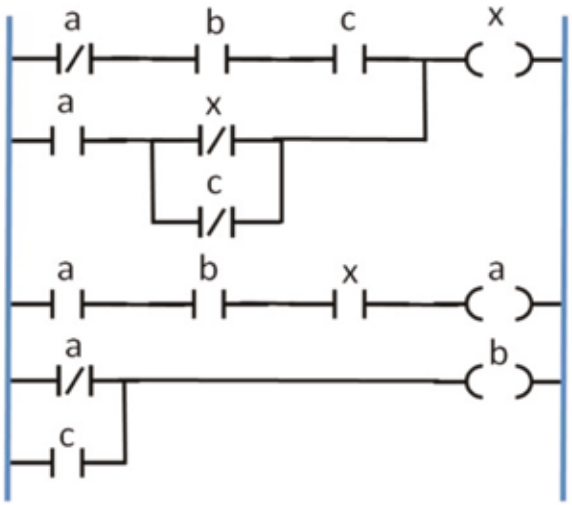
En la figura 4 se puede observar un ejemplo de la construcción del LD partiendo de las expresiones booleanas.

2.4. Otros tipos de LD

Así como las redes de Petri han evolucionado en formas más complejas y diversas, los diagramas ladder tienden a hacer lo mismo. Se ha encontrado que tímidamente se ha explorado el tema de los diagramas ladder de lógica difusa FLLD (por las siglas de Fuzzy Logic Ladder Diagram) [4], en los cuales se aprecian claramente los procesos de *"Fuzzification" y "Defuzzification"*. Otras propuestas han sido desarrolladas y pueden ser encontradas en la literatura [8]. De la misma manera también los fabricantes de PLC han trabajado en el desarrollo de nuevas herramientas para que la programación de los mismos resulte cada vez más amigable al usuario.

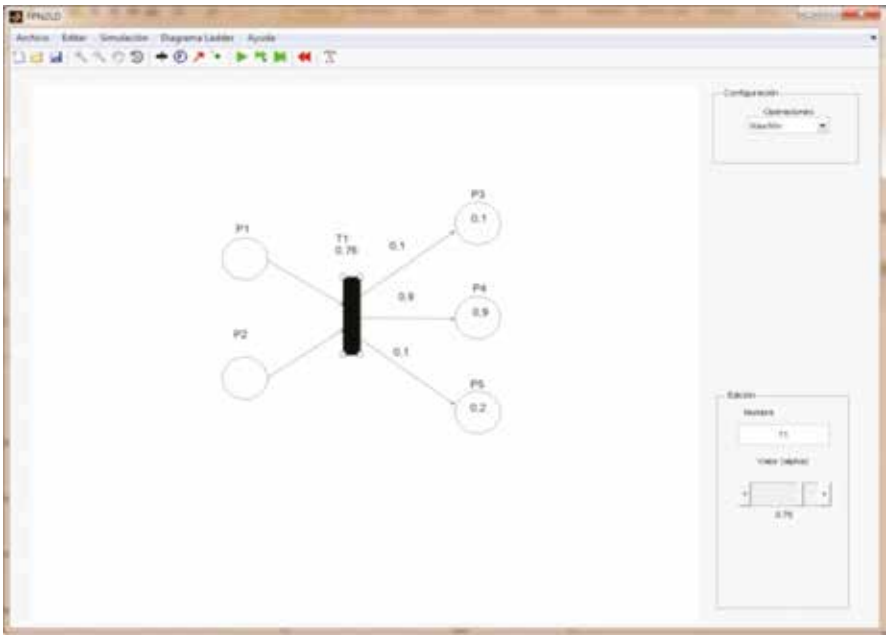
Figura 4. Ejemplo de construcción de LD partiendo de las expresiones booleanas

$$x = (\bar{a} \cdot b \cdot c) + \left(a \cdot (\bar{x} + \bar{c})\right)$$
$$a = (a \cdot b \cdot x)$$
$$b = \bar{a} + c$$



Fuente: elaboración propia.

Figura 5. Interfaz de usuario principal de la aplicación



Fuente: elaboración propia.

3. Desarrollo de la aplicación

Para validar la metodología desarrollada se ha realizado el diseño e implementación de una aplicación en forma de caja de herramientas (*toolbox*) para Matlab®, con la cual el usuario puede diagramar sus diseños de FPN y simularlos, para posteriormente tener la opción de convertirlos en LD. A continuación se presentan las características principales de la aplicación.

3.1. Redes de Petri Difusas

La primera parte de la aplicación consiste en un simulador de FPN, en donde el usuario tiene la posibilidad de diagramar sus diseños y observar la evolución de su red en diferentes modos.

El usuario puede agregar lugares, transiciones, arcos y marcas (algunas veces llamadas *tokens*), asignándole a cada elemento su valor respectivo. En la figura 5 se muestra la interfaz de usuario principal.

Existen 3 modos de simulación, los cuales son: "1 paso", "N pasos", "Hasta el final". El modo de un solo paso realiza una evolución sencilla al presionarlo, para el modo de N pasos el usuario selecciona la cantidad de pasos a ejecutar y, por último, el modo Hasta el final realiza el número de evoluciones necesario hasta que no quede transición en estado habilitado o regrese al estado inicial en una red cíclica. En la figura 6 se realiza un acercamiento detallado a los botones de edición de la red y simulación, donde aparecen, en su orden, los botones de: Transición, Lugar, Arco, Token, 1 Paso, N Pasos, Hasta el final y Regresar al inicio.

Figura 6. Acercamiento a la barra de edición y simulación de la FPN.

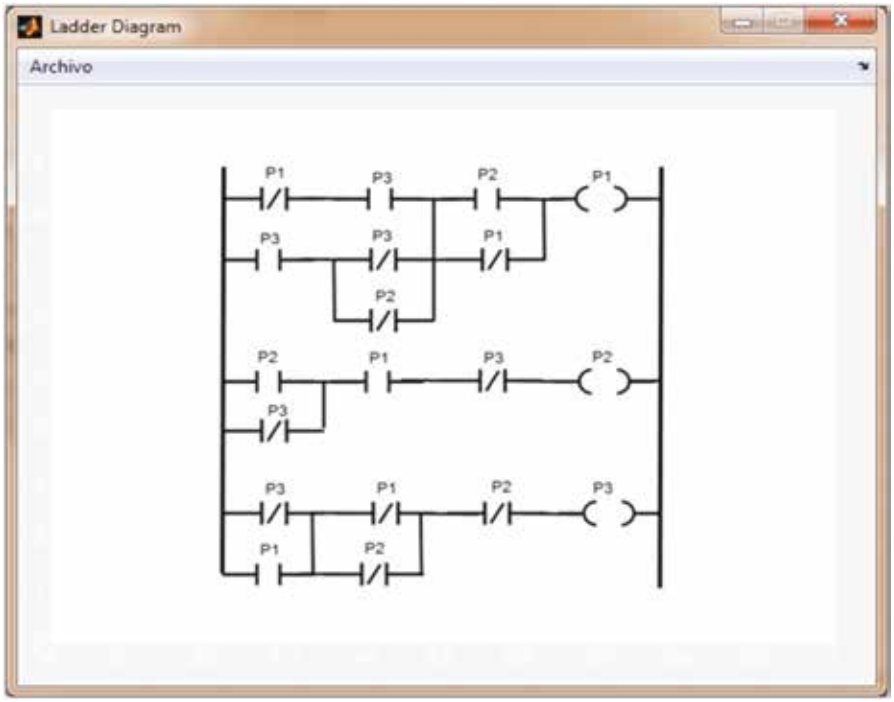
Fuente: elaboración propia.

3.2. FPN2LD

La función principal del programa es generar el LD mediante la FPN. Al presionar el botón mostrado en la figura 6 a la derecha, el programa pregunta el valor deseado de *umbral* y luego ejecuta el algoritmo que se encarga de obtener las variables lógicas y a su vez las expresiones lógicas, con las cuales el LD es generado.

Una ventana con el diagrama ladder generado aparece (figura 7), en esta los nombres de los contactos y las bobinas corresponden a los nombres asignados en la red de Petri difusa.

Figura 7. Ventana emergente con la figura del diagrama ladder generado



Fuente: elaboración propia.

4. Conclusiones

Las redes de Petri representan una solución al problema de modelar sistemas con eventos de concurrencia, sincronismo y secuencialidad, más aún, la extensión a redes de Petri difusas añade las ventajas de la lógica difusa aplicada a los sistemas de control.

Para poder implementar los sistemas diseñados mediante redes de Petri difusas en PLC, es necesario realizar un cambio de lenguaje. Debido a que el lenguaje de programación de PLC que más sobresale sobre los especificados por la Comisión Internacional Electrotécnica (*ICE*) [5] es el de diagramas ladder, se ha desarrollado una metodología que permite generar dichos diagramas a partir de una red de Petri difusa.

La solución propuesta en este documento no es una "Defuzzy-ficación", la cual podría ser implementada en propuestas novedosas como los diagramas ladder de lógica difusa (FLLD) que han venido siendo desarrollados [4], sino una "Umbralización" que permite una mayor flexibilidad al diseñador para ajustar el sistema a las condiciones específicas de la planta a controlar.

Se ha podido implementar el algoritmo de generación de los diagramas y ha arrojado buenos resultados al entregar funciones lógicas implementables en LD. A su vez un simulador de FPN ha sido desarrollado.

Como futuros proyectos relacionados se propone investigar la temática de los FLLD, para poder desarrollar una metodología que permita implementar las funciones lógicas difusas sin tener que umbralizar, realizando así una correcta "defuzzificación". También se debe explorar una metodología para programar los PLC directamente desde las redes de Petri, es decir, sin que exista la necesidad de traducirlas a un lenguaje ya existente, sino consolidando las redes de Petri como un lenguaje autónomo.

5. Referencias

[1] C. G. Looney, *Intelligent Systems, Control and Automation: Science and Engineering*, volume 11, chapter Fuzzy Petri Nets and Application. Springer, 1994.

[2] C. G. MacGown. The evolution of automation in the 21st century. In *Northcon/93. Conference Record*, pages 2-4, October 1993.

[3] G. B. Lee, H. Zandong, J. S. Lee. Automatic generation of ladder diagram with control Petri net. *Journal of Intelligent Manufacturing*, 15:245-252, April 2004.

[4] G. Cieslak, E. Dummermuth, O. J. Struger. Fuzzy logic ladder diagram program for a machine or process controller. (PATENT: US 5285376 A), 1991.

[5] International Electrotechnical Commission. *Programable Controllers, ICE 61131-3*, 2013.

[6] M. Kouzegharm M. A. Badamchizadeh, S. Khanmohammadi. Fuzzy petri nets for human behavior verification and validation. *IJACSA, International Journal of Advanced Computer Science and Applications*, 2(12):106-114, 2011.

[7] M. Zhou, R. Zurawski. *Petri Nets in Flexible and Agile Automation*, chapter Introduction to Petri Nets in Flexible and Agile Automation. Springer, US, 1995.

[8] P.R. Venkateswaran, J. Bhat, S. Meenatchisundaram. Formalism for fuzzy automation petri nets to ladder logic diagrams. *ARPN Journal of Engineering and Applied Sciences*, 4(10):83-92, 2009.

[9] T. Cao, A. C. Sanderson. *Intelligent Task Planning Using Fuzzy Petri Nets*. World Scientific, Farrer Road, Singapore, 1996.

[10] Y. Cao, G. Chen. A fuzzy petri-nets model for computing with words. *IEEE*

Transactions on Fuzzy Systems, 18(3):486-499, June 2010.

6. Apéndices

A. Deducción de la fórmula para la síntesis de las expresiones lógicas

A continuación se realiza la deducción de la fórmula usada para obtener las expresiones lógicas.

Debido a que se propone una umbralización, la fórmula es de carácter booleano y es la misma que se obtiene de las redes de Petri de bajo nivel [3]. Sin embargo se realiza una aproximación a la fórmula difusa con el objetivo de ilustrar el método.

A.1 Forma booleana

Cuando se tienen las variables lógicas todas de tipo booleano como lo son $P_i(\tau)$, $T_j(\tau)$, $C_j(\tau)$ y $E_j(\tau)$ es posible expresar el disparo de la transición en términos de las otras tres variables:

$$T_j(\tau) = \prod_{p_i | p_i \in {}^\circ t_j} P_i \cdot C_j \cdot E_j \quad (1)$$

$$P_i(\tau + \Delta\tau) = \left(P_i(\tau) + \sum_{t_j | t_j \in {}^\circ p_i} T_j(\tau) \right) \cdot \prod_{t_j | t_j \in {}^\circ p_i} \overline{T_j(\tau)} \quad (2)$$

Remplazando (1) en (2) se tiene que:

$$P_i(\tau + \Delta\tau) = \left(P_i(\tau) + \sum_{t_j | t_j \in {}^\circ p_i} \prod_{p_i | p_i \in {}^\circ t_j} P_i(\tau) \cdot C_j(\tau) \cdot E_j(\tau) \right) \cdot \prod_{t_j | t_j \in {}^\circ p_i} \overline{\prod_{p_i | p_i \in {}^\circ t_j} P_i(\tau) \cdot C_j(\tau) \cdot E_j(\tau)} \quad (3)$$

Como en un LD se sabe que la salida se activa en el momento siguiente a que toda la línea está energizada, es posible retirar la variable de tiempo de la expresión (3) teniendo en cuenta que la parte izquierda de la ecuación corresponde a la salida y la parte derecha corresponde a la entrada:

$$P_i = \left(P_i + \sum_{t_j | t_j \in {}^\circ p_i} \prod_{p_i | p_i \in {}^\circ t_j} P_i \cdot C_j \cdot E_j \right) \cdot \prod_{t_j | t_j \in {}^\circ p_i} \overline{\prod_{p_i | p_i \in {}^\circ t_j} P_i \cdot C_j \cdot E_j}$$

A.2 Forma difusa

La fórmula difusa resulta más sencilla que la booleana, por la simple razón de que se usa la misma función de cálculo del valor del lugar en la evolución siguiente. Solo hay que tener cuidado al determinar cuál de los tres casos corresponde al lugar en cuestión.

La definición de la variable transición está dada por:

$$T_j(\tau) = C_j(\tau) \wedge E_j(\tau)$$

Dado que:

$$C_j(\tau) = \begin{cases} 1, & \text{si } \mu_{M,t_j} \geq \alpha_{t_j} \\ 0, & \text{si } \mu_{M,t_j} < \alpha_{t_j} \end{cases}$$

Con

$$\mu_{M,t_j} = \wedge_{p_i | p_i \in {}^\circ t_j} P_i(\tau)$$

Por lo tanto la función difusa para obtener las expresiones es:

$$P_i(\tau + \Delta\tau) = \begin{cases} P_i(\tau) \vee \left(\mu_{M,t_j} \wedge \beta(t_j, p_i) \right), & \text{si } p_i \notin I(t_j) \\ \mu_{M,t_j} \wedge \beta(t_j, p_i), & \text{si } p_i \in I(t_j) \cap O(t_j) \\ 0, & \text{si } i \in I(t_j) \backslash O(t_j) \end{cases}$$

Eliminando la variable de tiempo τ por la misma razón que en la forma booleana se tiene entonces:

$$P_i = \begin{cases} P_i \vee \left(\mu_{M,t_j} \wedge \beta(t_j, p_i) \right), & \text{si } p_i \notin I(t_j) \\ \mu_{M,t_j} \wedge \beta(t_j, p_i), & \text{si } p_i \in I(t_j) \cap O(t_j) \\ 0, & \text{si } i \in I(t_j) \backslash O(t_j) \end{cases}$$